PROCEEDINGS OF SCIENCE

A quantum analytical Adam descent through parameter shift rule using Qibo

Matteo Robbiati,[a,b] **Stavros Efthymiou,**[c] **Andrea Pasquale**[a,c] **and Stefano Carrazza**[a,b,c,*]

[a] *TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Sezione di Milano Via Celoria 16, 20133, Milan, Italy.*

[b] *CERN, Theoretical Physics Department, CH-1211 Geneva 23, Switzerland.*

[c] *Quantum Research Centre, Technology Innovation Institute, Abu Dhabi, UAE.*

E-mail: matteo.robbiati@unimi.it, stavros.efthymiou@tii.ae, andrea.pasquale@unimi.it, stefano.carrazza@cern.ch

In this proceedings we present quantum machine learning optimization experiments using stochastic gradient descent with the parameter shift rule algorithm. We first describe the gradient evaluation algorithm and its optimization procedure implemented using the Qibo framework. After numerically testing the implementation using quantum simulation on classical hardware, we perform successfully a full quantum hardware optimization exercise using a single superconducting qubit chip controlled by Qibo. We show results for a quantum regression model by comparing simulation to real hardware optimization.

41st International Conference on High Energy physics - ICHEP2022
6-13 July, 2022
Bologna, Italy

*Speaker

PoS(ICHEP2022)206

1. Introduction

In this work we use `Qibo` [1–3], a full-stack open-source framework for quantum simulation, control and calibration, to perform a gradient-based optimization on a one qubit Quantum Process Unit (QPU). Specifically, we implement an Adam optimizer [4], a stochastic gradient descent method. Classical machine learning strategies make use of the Back-Propagation algorithm [5] which requires to know the value of the target function in the middle of the propagation in order to estimate the errors made in the predictions. On the other hand, this cannot be done when the model is a Variational Quantum Circuit (VQC) because it would be necessary to perform a measurement in

the middle of the propagation in order to know the value of the function, causing the system to collapse to one of the accessible states and the consequent loss of the information accumulated up to that moment.

VQC requires a method for evaluating gradients which should be deployable on the quantum hardware available in the NISQ era [6]. A method for evaluating gradients of a quantum circuit was proposed in 2018 and it is known as *Parameter Shift Rule* (PSR) [7, 8]. In the next sections we first summarize the PSR algorithm and then we show results for a VQC regression optimization example where all results presented in this manuscript have been obtained using the latest backends available in `Qibo` (v0.1.8): `qibojit` [2], for simulation; `qibolab`[1] for hardware control; and `qibocal`[2] for calibration, characterization and validation.

2. The Parameter Shift Rule

Let us consider a circuit $\mathcal{U}(\boldsymbol{\theta})$ which, when applied to the initial state $|0\rangle$, returns the final state $\mathcal{U}(\boldsymbol{\theta})\,|0\rangle = |q_f\rangle$. Let also B be an observable we involve into the estimation of an output variable y[3]. Finally, let us consider the unitary operator

$$\mathcal{G} = \exp\big[-i\mu G\big], \tag{1}$$

in which the variational parameter $\mu \in \boldsymbol{\theta}$ appears and which has at most two eigenvalues $\pm r$. We are interested in evaluating $\partial_\mu f$, where f is defined as follows:

$$f(\boldsymbol{\theta}) \equiv \langle 0|\mathcal{U}^\dagger(\boldsymbol{\theta})\, B\, \mathcal{U}(\boldsymbol{\theta})|0\rangle\,. \tag{2}$$

It can be shown that, if μ appears in a single gate in the form (1), the target estimation can be written as follows:

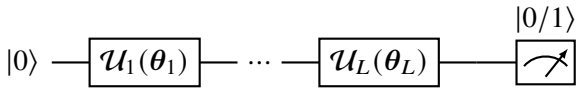
$$\partial_\mu f = r\big[f(\mu^+) - f(\mu^-)\big], \tag{3}$$

with $\mu^\pm = \mu \pm s$ and $s = \pi/4r$. We use the case presented in [7], which requires the Hermitian G picked from the set of rotation generators $\frac{1}{2}\{\sigma_x, \sigma_y, \sigma_z\}$. In this case we can use the remarkable values $s = \pi/2$ and $r = 1/2$.

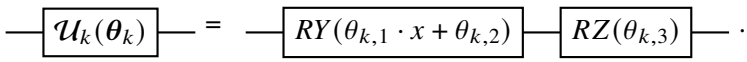
[1] https://github.com/qiboteam/qibolab
[2] https://github.com/qiboteam/qibocal
[3] In this case we use $\langle B\rangle \equiv \text{Prob}(|0\rangle) - \text{Prob}(|1\rangle)$ as estimator of y.

2.1 Evaluating gradients in a re-uploading strategy

The circuit we build up follows the idea presented in 2019 by Adrián Pérez-Salinas *et al.* [9], also called as *re-uploading strategy*. It allows to use a sequence of parametrized gates applied to a single qubit structure as model. Specifically, we summarize below the model used:



with an arbitrary number of layers $\mathcal{U}_k(\boldsymbol{\theta}_k)$, defined as follows:



Each layer involves two parametric rotation gates. In the first rotation, we impose that the angle is constructed by a combination of two variational parameters and x, which is an input data. By repeating this procedure multiple times, we obtain the effect of the data re-uploading strategy. As explained in the previous section, the PSR can be used for evaluating the derivative of a circuit with respect to a variational parameter which appears at the exponent in a gate like (1). Considering our ansatz, when the feature is involved into the parameter's definition, the PSR takes into account the entire angle of rotation θ', which we calculate as $\theta' = \theta \cdot x$. Thus, we need to correct the formula dividing the shift parameter s by x when evaluating $f(\pm\mu)$ and, once the two f's are obtained, we must recombine the values in the following way:

$$\partial_\mu f = r\left[f(\mu^+) - f(\mu^-)\right] \cdot x. \tag{4}$$

The effect of this correction can be seen in Figure 1, where five coloured lines are shown, each of which corresponds to a randomly selected value of θ with which we initialised a circuit consisting of a rotation around the X axis and for which the angle of rotation is calculated as $\theta' = \theta \cdot x$. We calculate the derivative with respect to θ of the term:

$$\hat{y} = \left(\langle B\rangle_x - c\right)^2, \tag{5}$$

where $\langle B\rangle_x$ is obtained through the difference of the probabilities of occurrence of the states $|0\rangle$ and $|1\rangle$ after N_{shots} executions of the circuit and c is a constant for which we choose the arbitrary value $c = 0.2$. Finally, we plotted the differences obtained by calculating the derivatives with the PSR and the `GradientTape()` method of `TensorFlow`. We carry out this procedure with and without the correction outlined above.

2.2 Loss function's gradient

Our aim in this draft is to use PSR for gradient descent optimization. Since we choose a *Mean Squared Error* loss function J_{mse}, we need two different estimators for calculating its derivative with respect to a μ as explained in [10]. In fact, once we have chosen an input variable x_j and defined the associated circuit via re-uploading strategy, we can execute it N_{shots} times and calculate the following contribution to the loss function which we call J_j. At this point the derivative of J_j with respect to μ becomes:

$$\partial_\mu J_j(\boldsymbol{\theta}) = 2\left(\langle B\rangle_{\boldsymbol{\theta},x_j} - y\right)\partial_\mu \langle B\rangle_{\boldsymbol{\theta},x_j}, \tag{6}$$

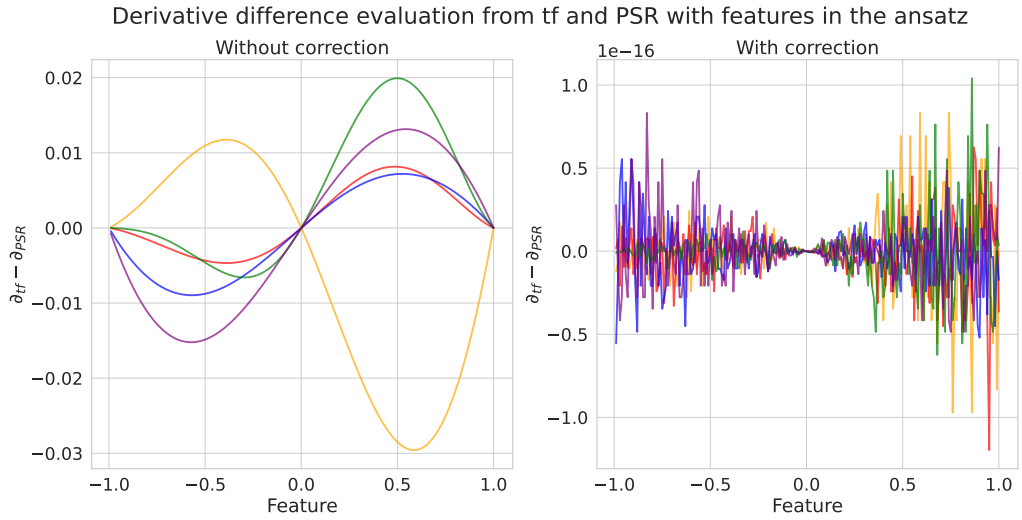


Figure 1: Difference between the derivative with respect to θ evaluated with the `TensorFlow` module `GradientTape()` called ∂_{tf} and our PSR implementation called ∂_{PSR}. On the left, the value is calculated without making any change to the first version of the PSR. On the right, we can see the effect of the correction, which breaks down errors to the order of 10^{-16}. Different colors correspond to different rotation's combinations.

where $\langle B \rangle_{\boldsymbol{\theta}, x_j}$ is the estimator of y, calculated as difference of the probabilities of occurrence of the two fundamental states after performing the measurements on $|q_f\rangle$ and $\partial_\mu \langle B \rangle_{\boldsymbol{\theta}, x_j}$ can be evaluated using the PSR. With the subscript x_j we are emphasising that the circuit under consideration has a structure that depends explicitly on the variable x_j. This procedure must be carried out for each of the p variational parameters involved in the definition of the model. Of course, N_{data} can be used for the training, involving a computational cost equal to $O(3pN_{data}N_{shots})$ for each optimization step. This aspect makes PSR a very computationally heavy technique. Nevertheless, it remains one of the few possible solutions for successfully performing gradient descent on quantum hardware. Before moving on to the description of the results obtained, let us summarise the algorithm we use for evaluating the derivative of J_{mse} in the following block of pseudo-code.

Optimization with parameter-shift rule

initialize $\partial J_{mse} = \vec{0}$
for `this_feature` and `this_label` **in** range N_{data}**:**
    define `this_circuit(this_feature)`
    estimate $\langle B \rangle$ with N_{shots} of `this_circuit`
    for μ **in range** p**:**
        with (3) evaluate $\partial_\mu \langle B \rangle$
        use $\langle B \rangle$ and (3) for calculate : `this_dloss` through (6)
        $\partial_\mu J_{mse}$ `+= this_dloss` .

N_{data}	N_{shots}	η	N_{epochs}	ε_J
25	1024	0.1	100	$5 \cdot 10^{-3}$

Table 1: Hyper-parameters used to train the qubit.

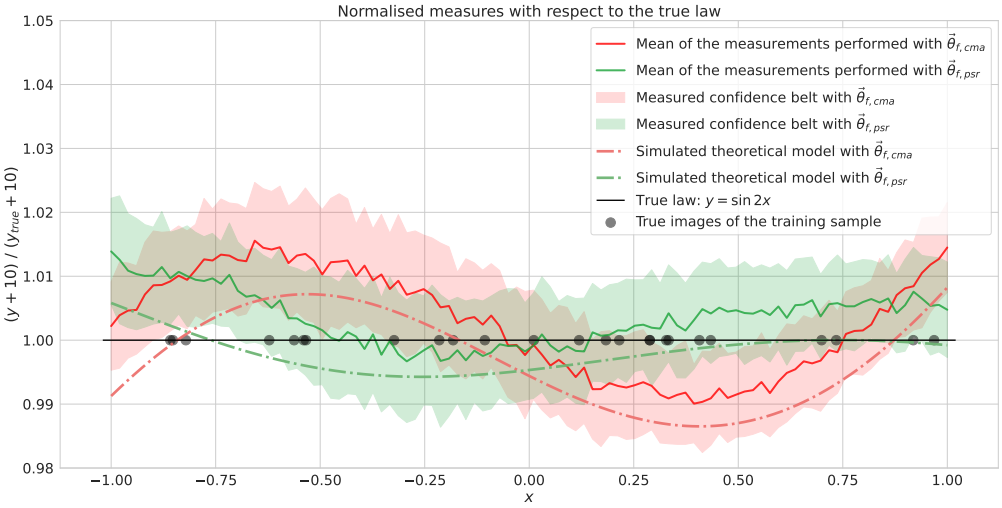


Figure 2: Results obtained by performing the optimization on the QPU and normalized with respect to the true law. The choice of adding $k = 10$ to each result is entirely arbitrary and serves to prevent the values under consideration from diverging for x tending to zero. The black line is the true law $\sin 2x$ and the black points are the true images of the training set. Colors green and red respectively refers to our PSR and to the CMA optimization. The dashed lines are the prediction's sets purposed through simulated theoretical models. The continuum lines and the confidence belts are drawn using one hundred predictions for each of one hundred points equally distributed into the interval $[-1, 1]$.

3. Deployment on quantum hardware

We can execute the code implementation presented above directly on the QPU by choosing the appropriate `Qibo` backend. In this way we can perform the fit of $y = \sin 2x$ with $x \in [-1, 1]$ directly on quantum hardware. For this exercise we use the single qubit platform located in the Quantum Research Centre (QRC) of the Technology Innovation Insitute (TII) in Abu Dhabi[4]. The selected optimization hyper-parameters are reported in Table 1 where η and ε_J are respectively the Adam's learning rate and a threshold value for J we impose for stopping the optimization.

At first, we fix the initial parameters $\boldsymbol{\theta}_0$ and the training set. At this point we perform the optimization on the qubit using `qibolab`, stopping it once reached the ε_J value, obtaining the $\boldsymbol{\theta}_{best}$. We use the trained model to make statistics about the predictions; we pick one hundred points equally distributed in the $[-1, 1]$ range and for each point x_j we evaluate one hundred times the prediction. Finally, we calculate mean and standard deviation from the mean for each j. We use them for drawing the predictions as continuous line and the confidence bands in Figure 2. Secondly, we use $\boldsymbol{\theta}_{best}$ obtained above for simulating a quantum circuit on a classical hardware and for getting the *simulated theoretical* predictions. This passage is crucial to highlight the difference between the

[4] https://www.tii.ae/quantum

noisy quantum hardware and the classical simulation. The simulated predictions are presented as dashed lines in Figure 2. To give consistency to the analysis, we compare the results obtained using this optimizer with those obtained using a genetic algorithm (CMA-ES). We also decide to present the results normalized with respect to the true law, which is represented as black line in Figure 2. We first note that the simulated theoretical model is compatible with the measurements recorded by the qubit, within the compatibility range we have defined. This is the case for both optimization methods. Furthermore, looking at the theoretical law, we see that it falls within the confidence belt relative to our training for a good part of the domain. The model proposed by the qubit, in this range, is compatible with the target law: we have successfully performed a gradient descent on a QPU with one qubit.

We conclude the discussion by highlighting that Qibo has achieved the level of completion required to obtain a successful application through the open-source availability of modules and backends for simulation, control and calibration.

Acknowledgements MR and SC are supported by CERN QTI program. SC is supported by the European Research Council under the European Union's Horizon 2020 research and innovation Programme (grant agreement number 740006). The Qibo project is supported by QRC.

References

[1] Stavros Efthymiou, Sergi Ramos-Calderer, Carlos Bravo-Prieto, Adrián Pérez-Salinas, Diego García-Martín, Artur Garcia-Saez, José Ignacio Latorre, and Stefano Carrazza. Qibo: a framework for quantum simulation with hardware acceleration. *Quantum Science and Technology*, 7(1):015018, dec 2021.

[2] Stavros Efthymiou, Marco Lazzarin, Andrea Pasquale, and Stefano Carrazza. Quantum simulation with just-in-time compilation. *Quantum*, 6:814, sep 2022.

[3] Stefano Carrazza, Stavros Efthymiou, Marco Lazzarin, and Andrea Pasquale. An open-source modular framework for quantum computing, 2022.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[5] Ronald J. Williams David E. Rumelhart, Geoffrey E. Hinton. Learning representations by back-propagating errors. *Nature*, 1986.

[6] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, aug 2018.

[7] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Physical Review A*, 98(3), sep 2018.

[8] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3), mar 2019.

[9] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, feb 2020.

[10] Ryan Sweke, Frederik Wilde, Johannes Meyer, Maria Schuld, Paul K. Faehrmann, Barthé lémy Meynard-Piganeau, and Jens Eisert. Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum*, 4:314, aug 2020.