Full-stack Quantum Machine Learnig

Matteo Robbiati

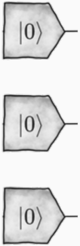
28 September 2023

A snapshot of Quantum Computing

Quantum Computing and circuits notation

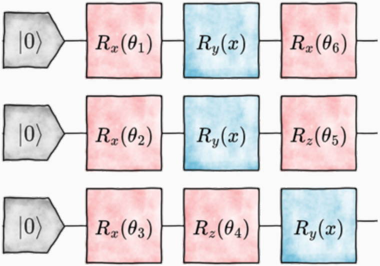
Quantum Computing and circuits notation

- Classical bits are replaced by **qubits**: $|q\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$;



Quantum Computing and circuits notation

- Classical bits are replaced by **qubits**: $|q\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$;
- we modify the qubits state by applying unitaries, which we call **gates**;

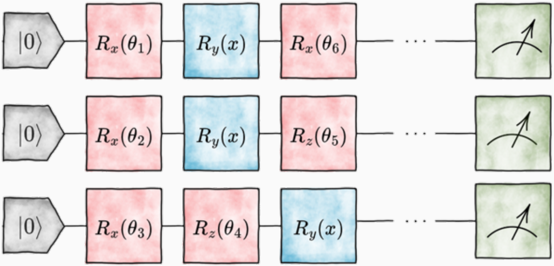


Quantum Computing and circuits notation

- Classical bits are replaced by **qubits**: $|q\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$;
- we modify the qubits state by applying unitaries, which we call **gates**;
- we extract information by calculating expected values:

$$\langle q_i | \mathcal{C}^\dagger(\boldsymbol{\theta}) \hat{O}\, \mathcal{C}(\boldsymbol{\theta}) | q_i \rangle \,,$$

with $\mathcal{C}(\boldsymbol{\theta})$ parametric circuit, $|q_i\rangle$ initial qubit's state and $\hat{O}$ arbitrary observable.



Quantum Machine Learning

Quantum Machine Learning - doing ML using QC

Machine Learning
$\mathcal{M}$: model;
$\mathcal{O}$: optimizer;
$\mathcal{J}$: loss function.
(x, y): data

Quantum Computation
$\mathcal{Q}$: qubits;
$\mathcal{S}$: superposition;
$\mathcal{E}$: entanglement.

Machine Learning

$\mathcal{M}$: model;

$\mathcal{O}$: optimizer;

$\mathcal{J}$: loss function.

(x, y): data

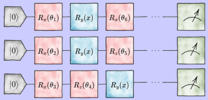
Quantum Computation

$\mathcal{Q}$: qubits;

$\mathcal{S}$: superposition;

$\mathcal{E}$: entanglement.

Circuit execution



Machine Learning

$\mathcal{M}$: model;

$\mathcal{O}$: optimizer;

$\mathcal{J}$: loss function.

(x, y): data

Expected values

$y_{est} \equiv \langle q_f | \hat{O} | q_f \rangle$

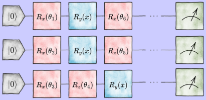
Quantum Computation

$\mathcal{Q}$: qubits;

$\mathcal{S}$: superposition;

$\mathcal{E}$: entanglement.

Circuit execution



Machine Learning

$\mathcal{M}$: model;

$\mathcal{O}$: optimizer;

$\mathcal{J}$: loss function.

(x, y): data

Expected values

$y_{est} \equiv \boxed{\langle q_f | \hat{O} | q_f \rangle}$

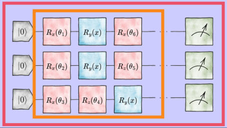
Quantum Computation

$\mathcal{Q}$: qubits;

$\mathcal{S}$: superposition;

$\mathcal{E}$: entanglement.

Circuit execution



Quantum Machine Learning!

Optimizer $\mathcal{O}$
Hybrid-strategy

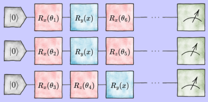
Machine Learning
$\mathcal{M}$: model;
$\mathcal{O}$: optimizer;
$\mathcal{J}$: loss function.
(x, y): data

loss function $\mathcal{J}$
$\mathcal{J}(y_{meas}, y_{est})$

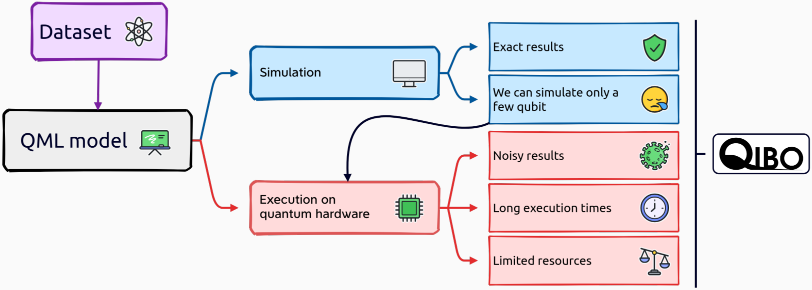
Expected values
$y_{est} \equiv \langle q_f | \hat{O} | q_f \rangle$

Quantum Computation
$\mathcal{Q}$: qubits;
$\mathcal{S}$: superposition;
$\mathcal{E}$: entanglement.

Circuit execution



Full-stack QML



Some results

High level API: Qibo

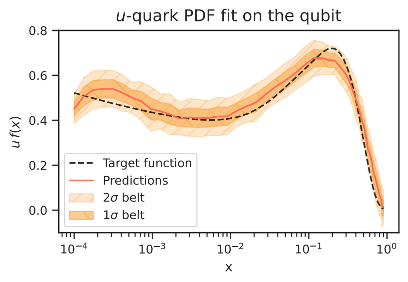
- </> define prototypes;
- </> implement training loop;
- </> simulate training.

Calibration: Qibocal

- calibrate qubits;
- generate platform configuration;

Execution: Qibolab

- allocate calibrated platform;
- compile and transpile circuits;
- execute the model and return results.



Parameter	Value
N_{data}	50
N_{shots}	500
MSE	50
Electronics	Xilinx ZCU216
Training time	2h

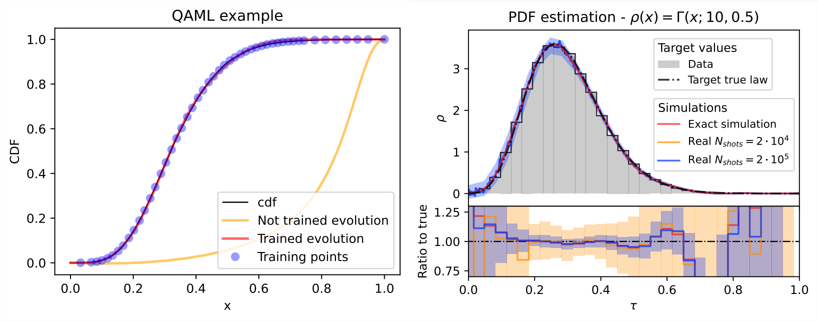
Determining Probability Density Functions (PDF) by fitting the corresponding Cumulative Density Function (CDF) using an adiabatic QML ansatz.

Algorithm's summary:

1. we optimize the parameters $\boldsymbol{\theta}$ of the following adiabatic evolution:

$$H_{\rm ad}(\tau;\boldsymbol{\theta}) = [1 - s(\tau;\boldsymbol{\theta})]\hat{X} + s(\tau;\boldsymbol{\theta})\hat{Z} \tag{1}$$

   in order to approximate some target CDF values with $\hat{F}(x_k \equiv \tau) = \langle\psi(\tau)|\hat{Z}|\psi(\tau)\rangle$;
2. we derivate from $H_{\rm ad}$ a circuit $\mathcal{C}(\tau;\boldsymbol{\theta})$ whose action on the GS of $\hat{X}$ returns $|\psi(\tau)\rangle$;
3. the circuit at step 2. can be used to calculate the CDF;
4. we compute the PDF by derivating $\mathcal{C}$ with respect to τ using the Parameter Shift Rule.

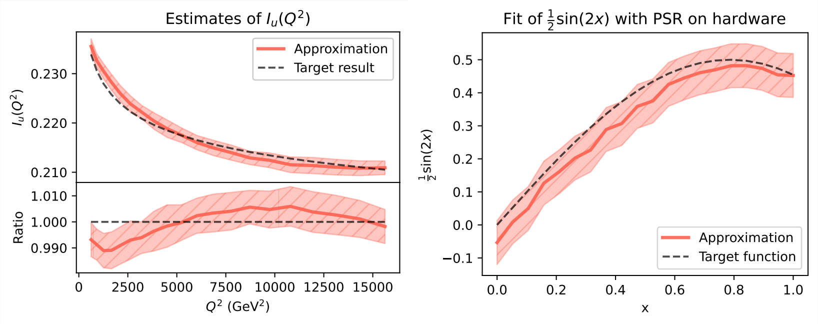


- Use Variational Quantum Circuits to calculate multi-dimensional integrals of the form:

$$I(\alpha) = \int_{x_a}^{x_b} g(\alpha; \boldsymbol{x}) \mathrm{d}^n \boldsymbol{x}. \quad (2)$$

- Algorithm's summary:

1. inspired by arXiv:2211.02834, we train the derivative of a VQC with respect to the integral variables $\boldsymbol{x}$ to approximate the integrand $g(\boldsymbol{x})$;
2. the derivatives are computed using the Parameter Shift Rule and this allows the same circuit $\mathcal{C}$ to be used for approximating any integrand marginalisation and the primitive!
3. thanks to 2., it's much more convenient to compute Eq. (2) when varying α.



Real time error mitigation in QML trainings   Coming soon!

Cleaning up the parameters space with a real time error mitigation strategy in order to overcome Noise-Induced Barren Plateaus (NIBP) when training a QML model.

Algorithm's summary:

1. we mitigate all the expected values E through Clifford Data Regression (CDR):

$$E_{\text{mit}} = \alpha_{\text{cdr}} E_{\text{noisy}} + \beta_{\text{cdr}}; \tag{3}$$

2. reduced CDR computational cost by updating $(\alpha, \beta)_{\text{cdr}}$ periodically during the training;
3. the mitigation removes the bounds and accelerate the training process.