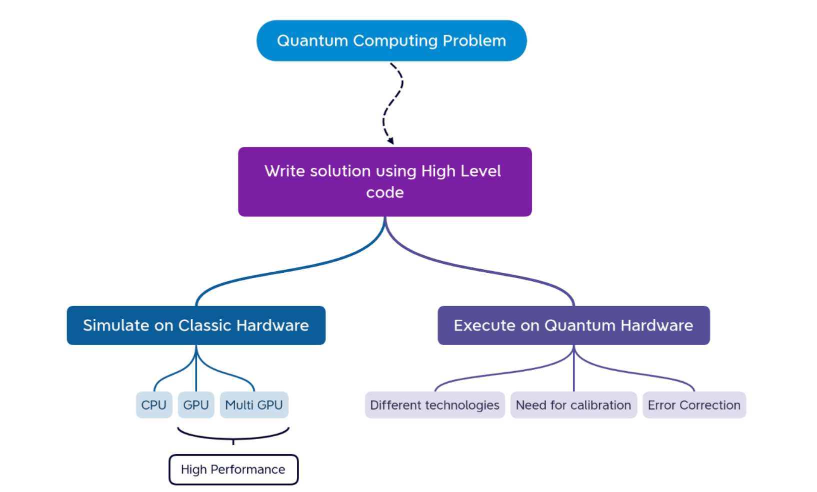
Quantum Computing & Quantum Machine Learning tutorials

Andrea Pasquale, Matteo Robbiati, Alessandro Candido

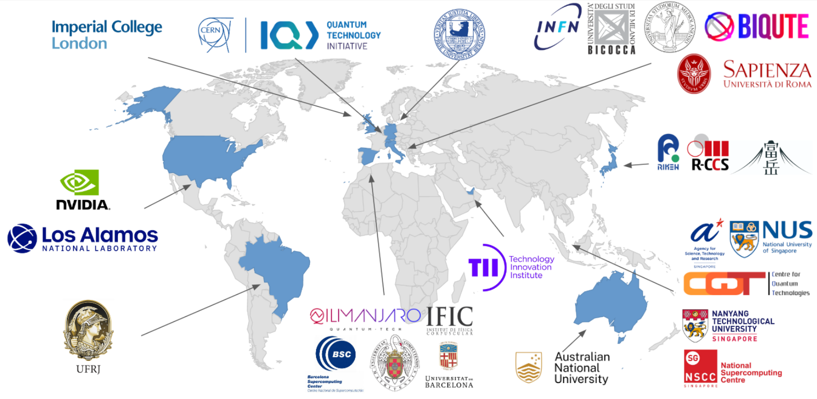
July, 2023

Quantum Computing using `Qibo`

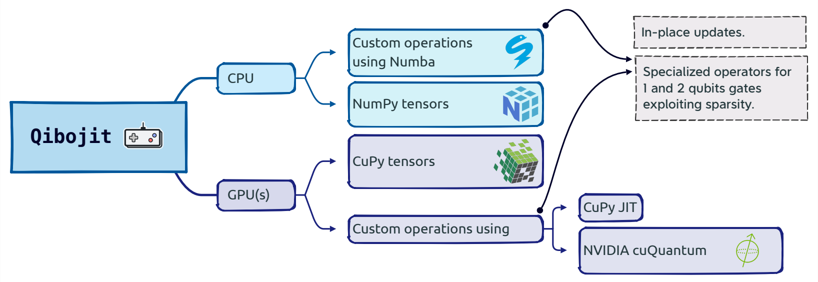
The aim of `Qibo`



Contributors at July 2023



High performance quantum simulation



Quantum simulation with just-in-time compilation, 2022

Quantum Machine Learning

Variational Quantum Circuits (VQCs)

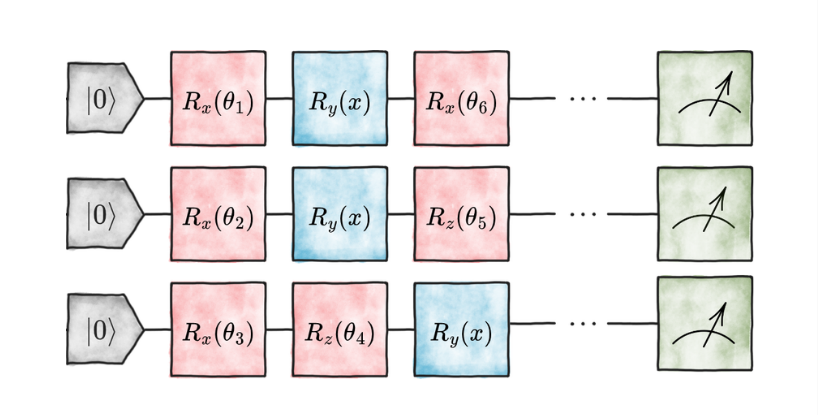


Figure 1: `AKA` a circuit with some parameters, which we call $\mathcal{U}(\boldsymbol{\theta})$.

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.

Quantum Machine Learning - operating on qubits

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.

VQC execution
$\mathcal{M}$, $\mathcal{S}$, $\mathcal{E}$
$\mathcal{U}(\boldsymbol{\theta})\,|q_i\rangle \rightarrow |q_f\rangle$

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.

VQC execution
$\mathcal{M}$, $\mathcal{S}$, $\mathcal{E}$
$\mathcal{U}(\boldsymbol{\theta})\,|q_i\rangle \rightarrow |q_f\rangle$

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

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.

VQC execution

$\mathcal{M}, \mathcal{S}, \mathcal{E}$

$\mathcal{U}(\theta)|q_i\rangle \rightarrow |q_f\rangle$

Expected values

$y_{est} \equiv \langle q_f|B|q_f\rangle$

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
$y_{est} \equiv \langle q_f | B | q_f \rangle$

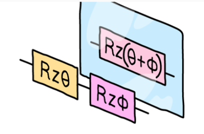
Quantum Computation
$\mathcal{Q}$: qubits;
$\mathcal{S}$: superposition;
$\mathcal{E}$: entanglement.

VQC execution
$\mathcal{M}$, $\mathcal{S}$, $\mathcal{E}$
$\mathcal{U}(\boldsymbol{\theta}) |q_i\rangle \rightarrow |q_f\rangle$

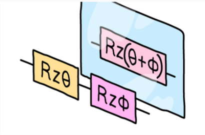
Why QML?

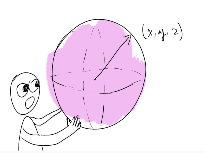
Why QML?

💵 **shallow models** thanks to superposition and entanglement;



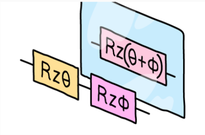
- **shallow models** thanks to superposition and entanglement;
- map problems into Hilbert's spaces loads to high **expressivity**;

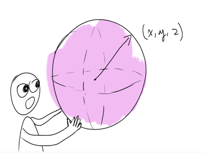


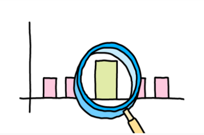


Why QML?

- **shallow models** thanks to superposition and entanglement;
- map problems into Hilbert's spaces loads to high **expressivity**;
- exploit QC sub-routines to **speed-up** classical algorithms (e.g. using Grover);

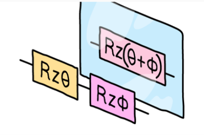


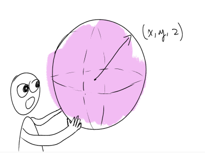


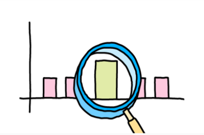
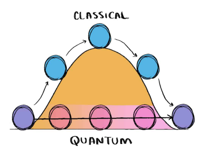


Why QML?

- **shallow models** thanks to superposition and entanglement;
- map problems into Hilbert's spaces loads to high **expressivity**;
- exploit QC sub-routines to **speed-up** classical algorithms (e.g. using Grover);
- physical advantages when dealing with **combinatorial optimization** (quantum annealing).





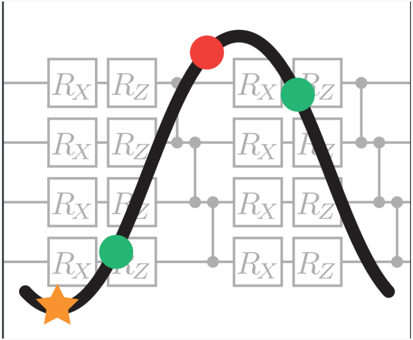
Rational

Using Variational Quantum Circuits we define a Variational Quantum Computer!

Rational

Using Variational Quantum Circuits we define a Variational Quantum Computer!

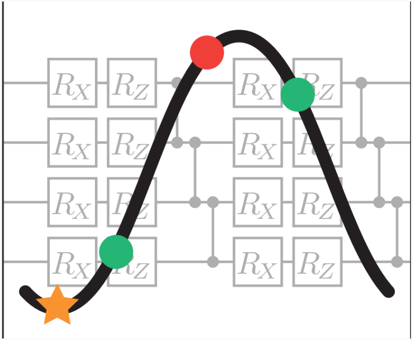
1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;



Rational

Using Variational Quantum Circuits we define a Variational Quantum Computer!

1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;
2. executing $\mathcal{U}(\boldsymbol{\theta})$ we use a variational quantum state to reach the solution;



Rational

Using Variational Quantum Circuits we define a Variational Quantum Computer!

1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;
2. executing $\mathcal{U}(\boldsymbol{\theta})$ we use a variational quantum state to reach the solution;
3. **Solovay-Kitaev theorem**: the number of gates needed by $\mathcal{U}$ to represent V with precision δ is $\mathcal{O}(\log^c \delta^{-1})$, where $c < 4$.