A Glimpse into Quantum-enhanced Machine Learning Solutions

Massimiliano Pronesti, Federico Tiblias, Giulio Corallo

April 27, 2022

Amadeus Knowledge Sharing Session

Who are we?

Giulio Corallo

Massimiliano Pronesti

Federico Tiblias

Three Computer Engineering and Data Science students from PoliTO, mainly interested in Machine Learning and High-performance Computing, who recently got fascinated by Quantum Computing.

Outline

Motivation

Why Quantum?

Until now, we've relied on supercomputers to solve most problems. These are very large classical computers, often with thousands of classical CPU and GPU cores. However, supercomputers are not very good at solving certain types of problems, which seem easy at first glance.

Why Quantum? A simple example

Imagine you want to seat 10 fussy people at a dinner party, where there is only one optimal seating plan out of all the different possible combinations. How many different combinations would you have to explore to find the optimal?

Can you guess how many combinations[1] ?

[1] Example from IBM's website
https://www.ibm.com/quantum-computing/what-is-quantum-computing/

Why Quantum? A simple example

For 2 people

2 Total combinations.

Why Quantum? A simple example

For 2 people

2 Total combinations.

For 5 people

120 Total combinations.

Why Quantum? A simple example

For 2 people

2 Total combinations.

For 5 people

120 Total combinations.

For 10 people

Over 3 Million of total combinations!!!

Why Quantum? A simple example

For 2 people

2 Total combinations.

For 5 people

120 Total combinations.

For 10 people

Over 3 Million of total combinations!!!

- Supercomputers don't have the working memory to hold the myriad combinations of real world problems.

Why Quantum? A simple example

For 2 people

2 Total combinations.

For 5 people

120 Total combinations.

For 10 people

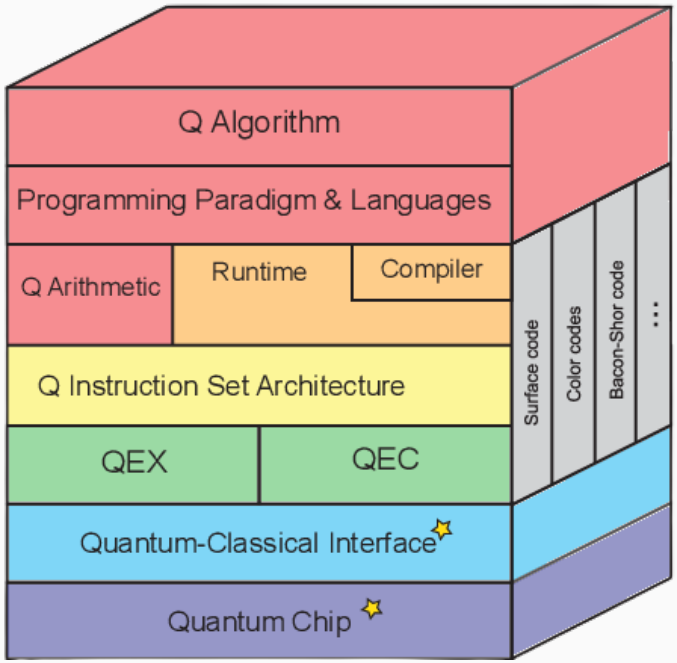
Over 3 Million of total combinations!!!

- Supercomputers don't have the working memory to hold the myriad combinations of real world problems.
- Supercomputers have to analyze each combination one after another, which can take a long time.

Why Quantum?

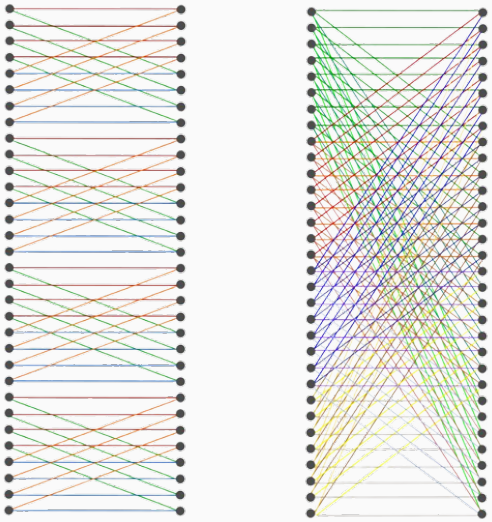
- Quantum computers represent problems in vast multidimensional spaces. Classical computers are more limited.
- Referring to the "10 fussy people at a dinner party" problem, with 22 qubit we can represent $2^{22} = 4194304$ states.
- The computation may be carried out on all those numbers in a single parallel computation. This built-in parallelism is the key to the power of quantum computers.

How is a Quantum computer programmed?

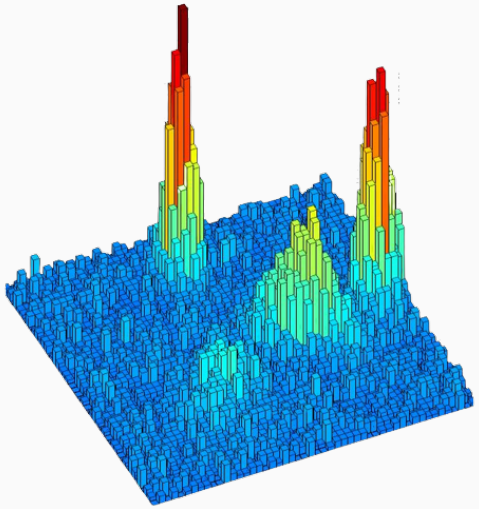


There is an interface between quantum mechanical processes and classical computer processes. Through this interface, input data from a classical computing device can be fed to quantum circuits.

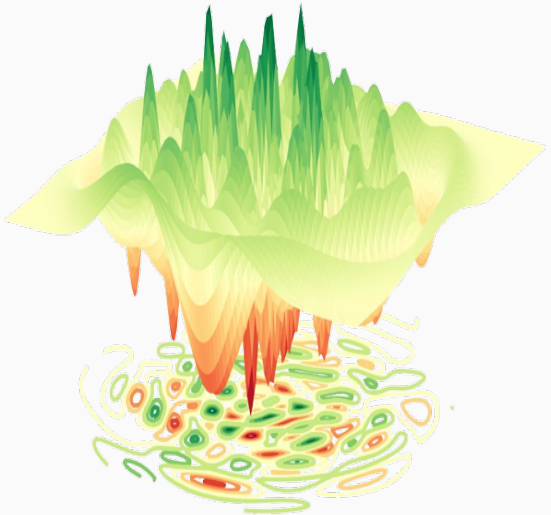
What are Quantum Computers good at?



Linear Algebra



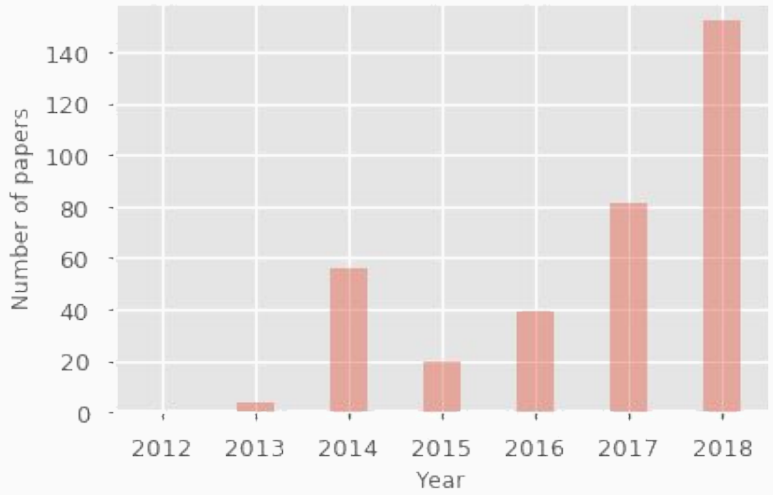
Sampling



Optimization

A Growing Interest in the field

- Many frameworks from big techs (IBM's Qiskit, PennyLane, Google's TensorFlow Quantum, Microsoft Q#, ...)
- A significant growth in research



- All the Big Techs (Microsoft, Google, IBM, Nvidia ...) have/want a quantum computer
- Growing interest in the field of Quantum Programming Languages

Quantum Computing Foundations

Bra-ket notation

Useful for representing quantum systems

Bra = Row

$$\langle A| = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots \end{bmatrix}$$

Ket = Column

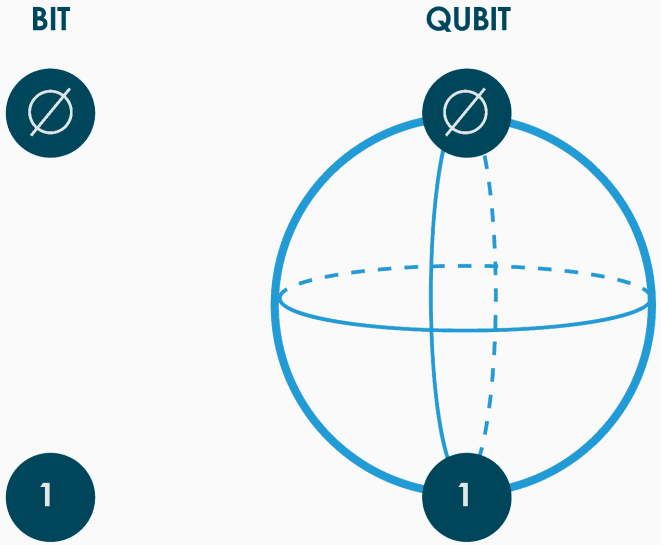
$$|B\rangle = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \end{bmatrix}$$

Properties & operations:

- Scalar product $= \langle A|B\rangle = [a_0, a_1, a_2, \cdots] \cdot [b_0, b_1, b_2, \cdots]^T$
- Norm $= \langle A|A\rangle = |A|^2$

What is a qubit?

- Building block for quantum computers
- 2-state quantum system (photon, electron, Schrödinger's cat, ...)
- 0, 1, both at the same time (superposition)
- Can be manipulated (quantum circuits)
- Can form more complex quantum systems (multi-quibit systems)
- Can be observed causing its collapse (measurement)



What is a qubit?

- What does it mean to be 0 and 1 simultaneously?
  It's a matter of probability during measurement
- Bra-ket qubit representation:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad |\psi\rangle = \begin{bmatrix} \psi_0 \\ \psi_1 \end{bmatrix}$$

- Probability measurement:

$$P(|\psi\rangle = 0) = |\psi_0|^2$$

$$P(|\psi\rangle = 1) = |\psi_1|^2$$

- Probability distribution: $\langle\psi|\psi\rangle = |\psi|^2 = 1 \ \forall \ \psi$

Multi-qubit system and entanglement

How can we represent 2 or more qubits with bra-kets?

With tensor products and longer vectors

$$|AB\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{bmatrix}$$

By stacking n qubits we can represent 2^n infinite precision numbers

$$P(A = 0, B = 1) = |a_0 b_1|^2$$

Multi-qubit system and entanglement

How can we represent state $|00\rangle + |11\rangle$ with a tensor product?

We can't as there is no set of values for a_0, a_1, b_0, b_1 that allows it.

$$|AB\rangle = \begin{bmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{bmatrix}$$

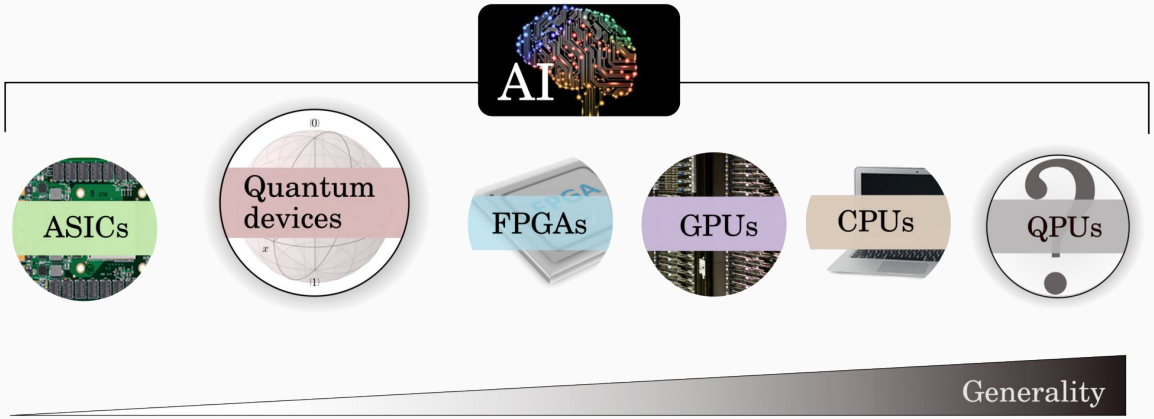
However, we can still create this state with quantum circuits. The resulting state is said to be entangled as measuring one qubit immediately tells us the state of the other.

$$|00\rangle + |11\rangle = \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{bmatrix}$$

Quantum Machine Learning

How can ML benefit from Quantum Computers?

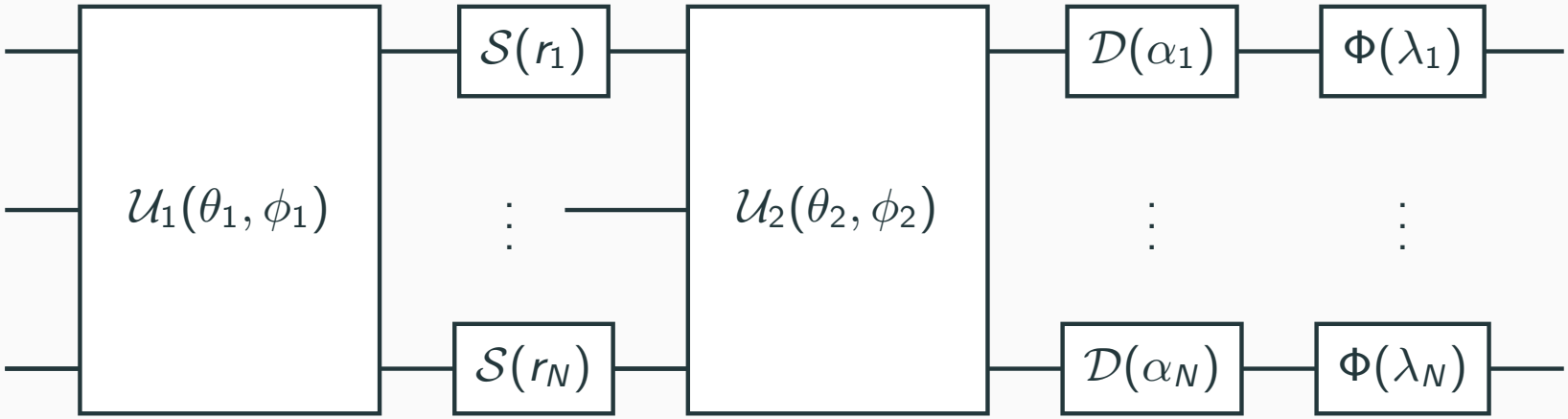
- AI/ML already uses (massively) special-purpose processors: GPUs, TPUs, ASICs
- Quantum computers (QPUs) could be used as special-purpose AI accelerators
- May enable training of previously intractable models



How can ML benefit from Quantum Computers?

Quantum computing can also lead to new machine learning models
Examples currently being studied are:

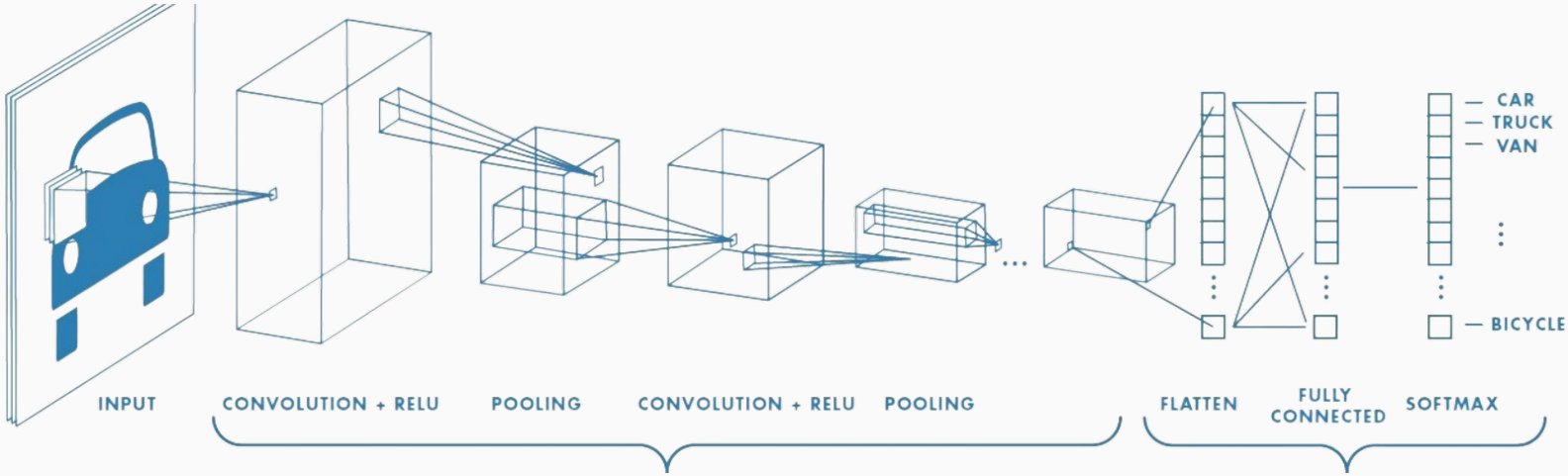
- Tensor Networks
- Kernel methods
- Boltzmann machines
- Variational Quantum Circuits
- Quantum Neural Networks



A lesson from Deep Learning's success

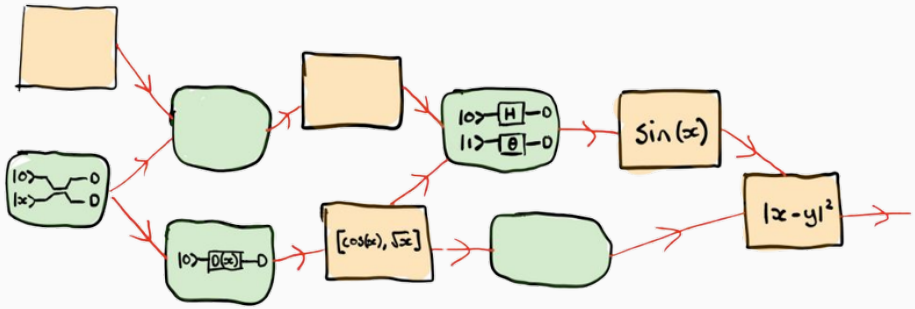
Why is Deep Learning Successful?

- hardware advancements(GPUs, recently TPUs)
- Workhorse algorithms (backprop, stochastic GD, ...)
- specialized, user-friendly, maintained, high-perfomance software (PyTorch, Tensorflow, Keras)



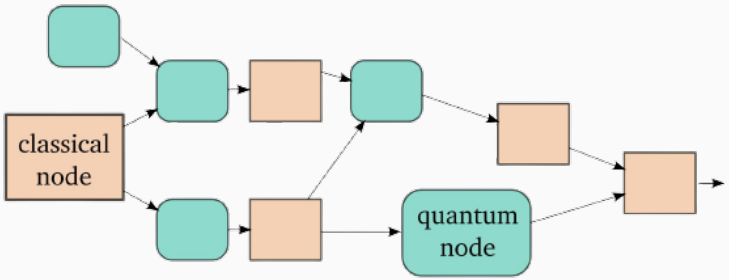
What can we leverage?

- hardware advancements(GPUs, TPUs + QPUs)
- workhorse algorithms
  (quantum-aware backpropagation, stochastic GD, ...)



- specialized, user-friendly, high-performance software (Pennylane, Qiskit, Tensorflow Quantum, ...)

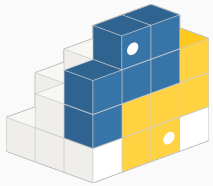
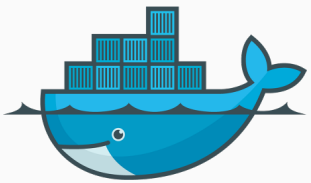
- hybrid computation



Qlearnkit

What is Qlearnkit?

- a Python library for Quantum Machine Learning, built on top of Qiskit and (optionally) Pennylane
- developed at EURECOM for MALIS and QUANTIS
- implements some famous Machine Learning and Deep Learning algorithms and models
- open-source and available on Github
- installable from PyPi or provided in a Docker container

QSVM - Basic idea

Many implementations are possible. The general structure is common:

1. Encode samples in quantum format
2. Compute a kernel matrix of distances using quantum circuits
3. Use matrix to solve SVM problem and obtain a solution

Steps 2 and 3 can benefit from a quantum speedup. A fully quantum solution has $O(\log mn)$ time complexity

(n: number of features, m: number of samples)

An implementation

1. Encode samples in quantum format

$$\mathcal{U}_{\Phi(\mathrm{x})} = \prod_{d} U_{\Phi(\mathrm{x})} H^{\otimes n}, \;\; U_{\Phi(\mathrm{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\mathrm{x}) \prod_{k \in S} P_i \right),$$

   Equivalent to mapping in a higher dimensional space.

2. Compute a kernel matrix of distances using quantum circuits

$$K_{ij} = \langle f(\vec{x_i}), f(\vec{x_j}) \rangle = \left| \langle \phi^\dagger(\vec{x_j}) | \phi(\vec{x_i}) \rangle \right|^2 = |\langle 0^n | \mathcal{U}^\dagger_{\Phi(\mathrm{x_i})} \mathcal{U}_{\Phi(\mathrm{x_j})} | 0^n \rangle|^2$$

   Equivalent to computing a scalar product.

(n: number of features, m: number of samples)

An implementation

3. Use this matrix to solve SVM problem and obtain a solution
   LS-SVM Formulation:

$$\begin{bmatrix} 0 & 1_N^T \\ 1_N & K + \gamma^{-1} I_N \end{bmatrix} \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ Y \end{bmatrix}$$

   α: SVM parameters

   β: bias term

   Predict as

$$f(\cdot) = K(\cdot, X) \cdot \alpha + \beta$$

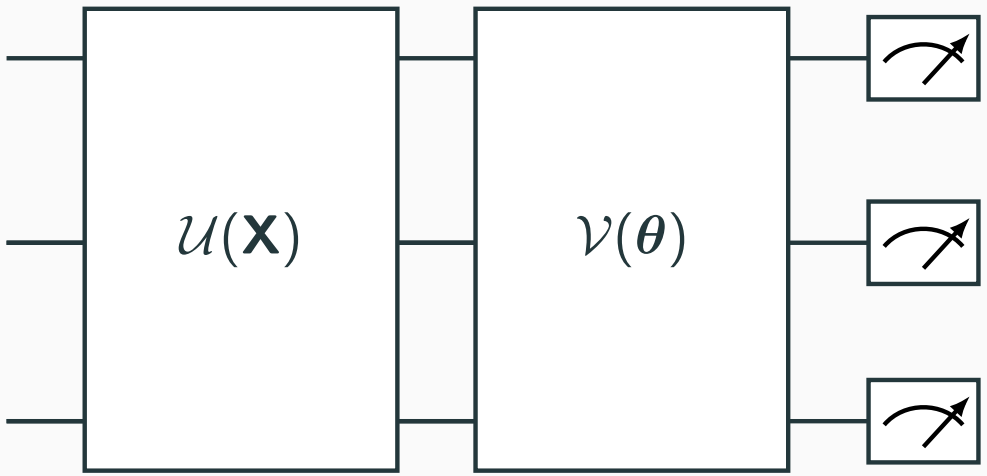
   Note: $K(\cdot, X)$ is done again on a quantum computer

(n: number of features, m: number of samples)

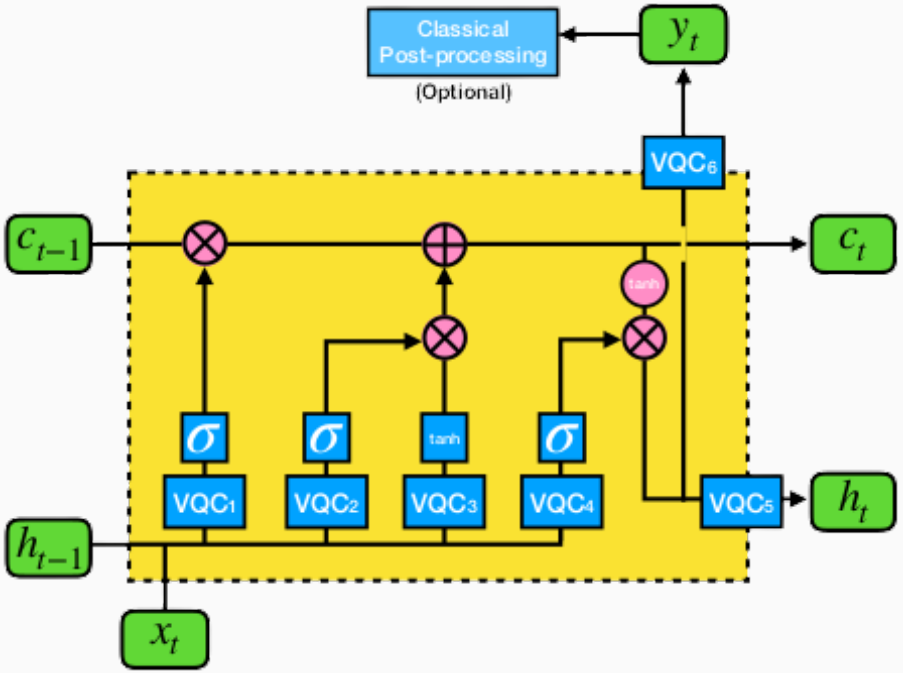
QLSTM - Basic idea

The Quantum Cell is a Variational Quantum Circuit (VQC):

- Encoding Layer for data preparation $\mathcal{U}(X)$
- Variational Layer $\mathcal{V}(\theta)$, with θ the learnable parameters
- Measurement Layer, to retrieve a classical bit string



QLSTM - Formulation



$$\mathbf{v}_t = [\mathbf{x}_t, \mathbf{h}_{t-1}]$$

$$\mathbf{f}_t = \sigma(\mathrm{VQC}_1(\mathbf{v}_t))$$

$$\mathbf{i}_t = \sigma(\mathrm{VQC}_2(\mathbf{v}_t))$$

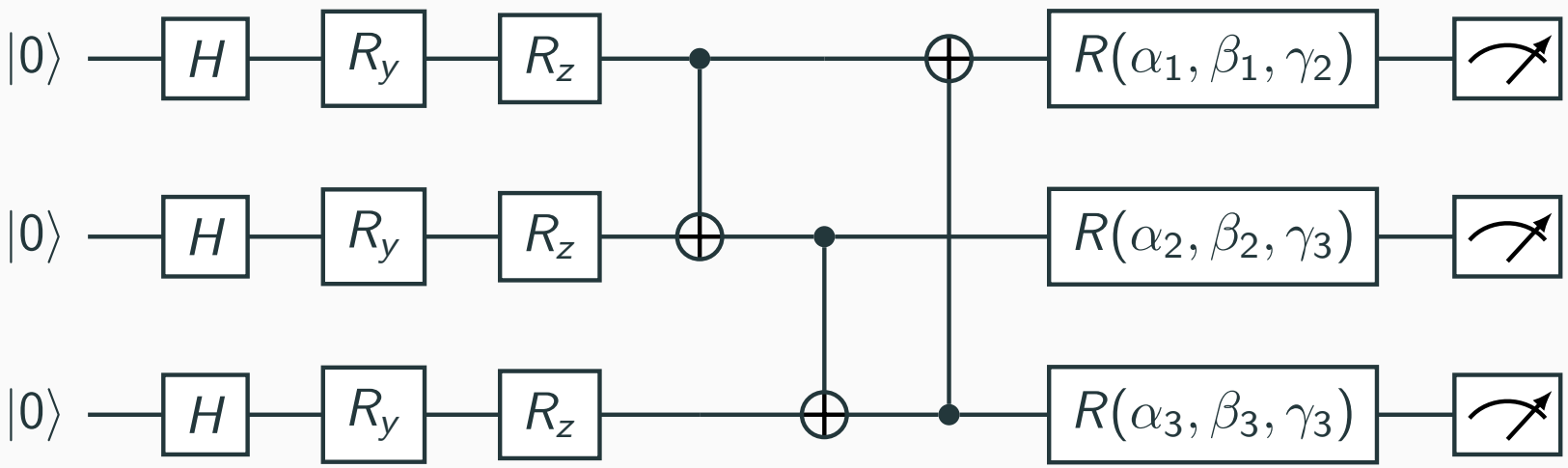
$$\tilde{\mathbf{C}}_t = \tanh(\mathrm{VQC}_3(\mathbf{v}_t))$$

$$\mathbf{c}_t = \mathbf{f}_t \mathbf{c}_{t-1} + \mathbf{i}_t \tilde{\mathbf{C}}_t$$

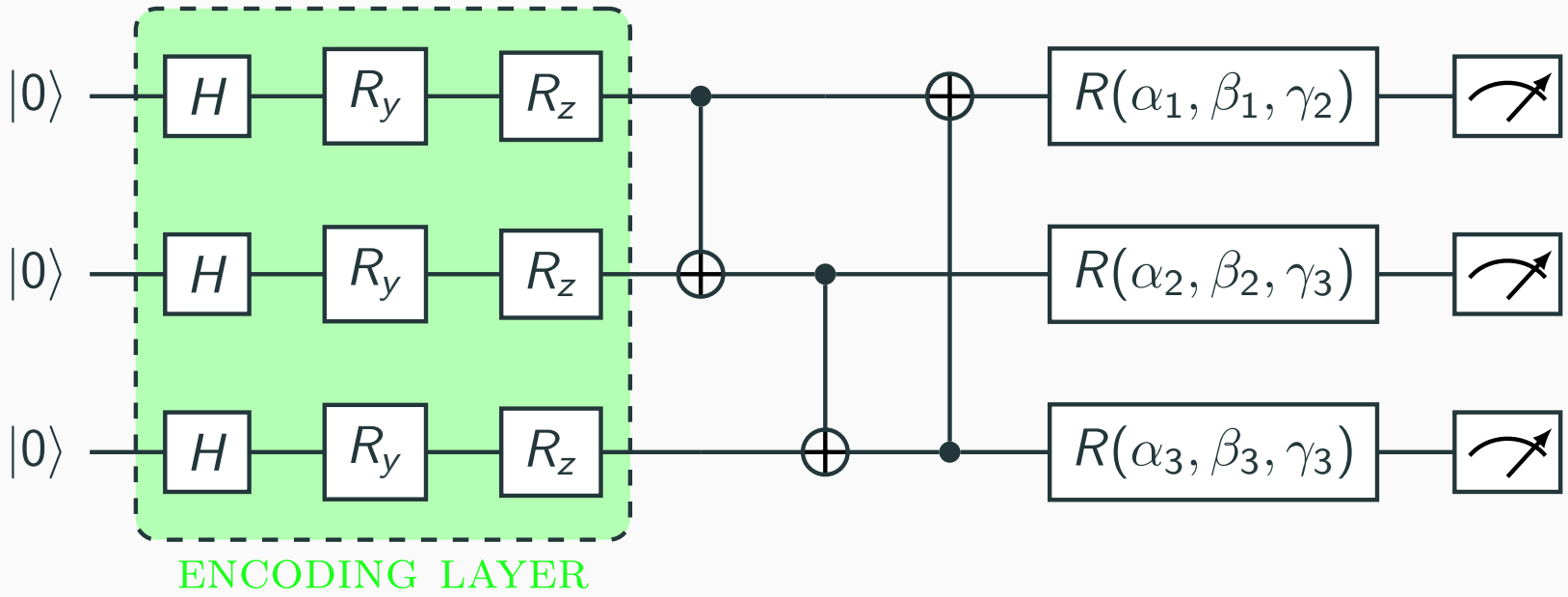
$$\mathbf{o}_t = \sigma(\mathrm{VQC}_4(\mathbf{v}_t))$$

$$\mathbf{h}_t = \mathrm{VQC}_5(\mathbf{o}_t \cdot \tanh(\mathbf{c}_t))$$

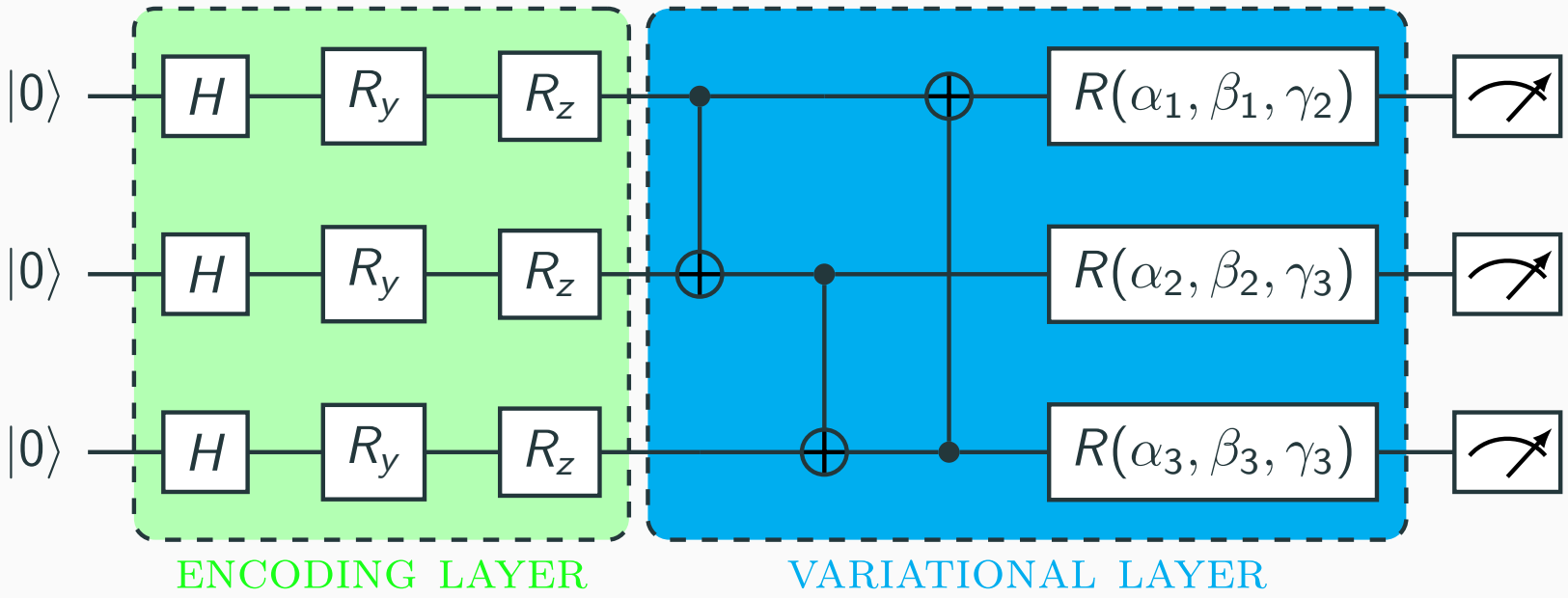
Architecture



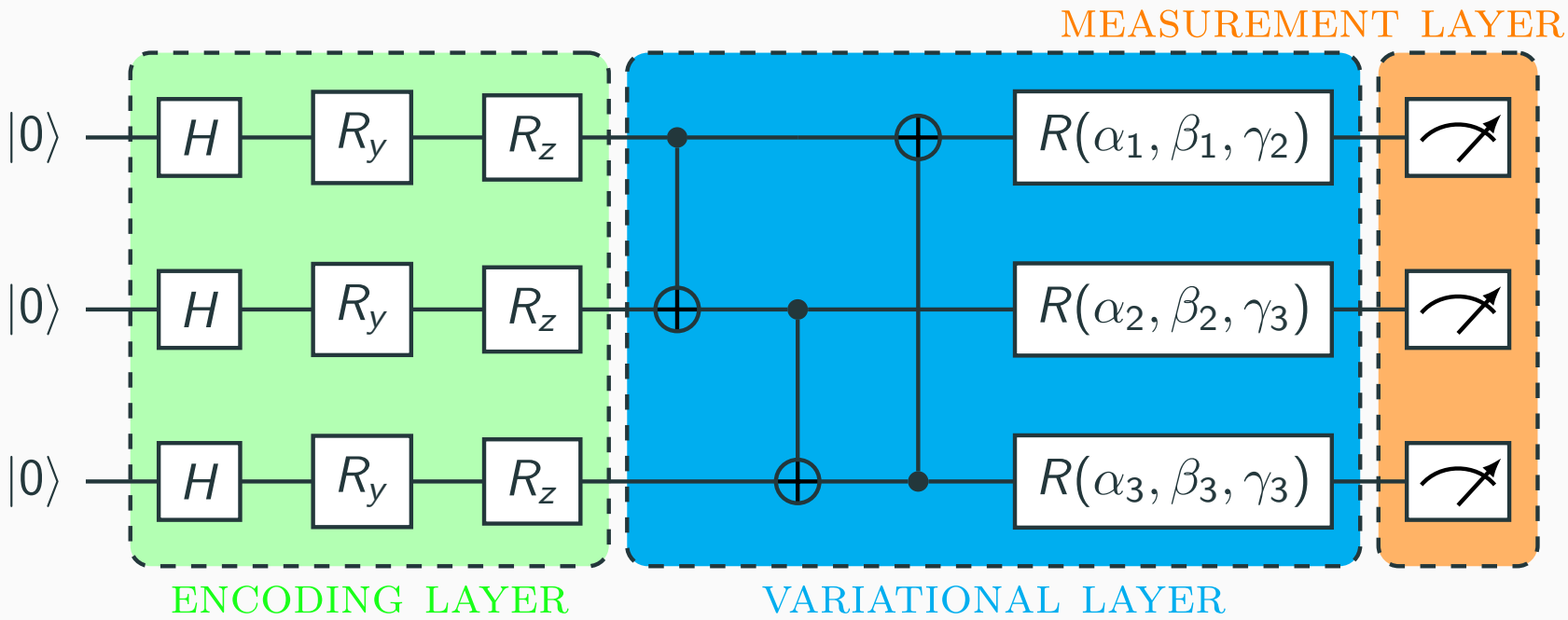
Architecture



Architecture



Architecture



Encoding Layer - Data preparation

The classical input vector is encoded into rotation angles to "guide" the single-qubit rotations as follows:

Encoding Layer - Data preparation

The classical input vector is encoded into rotation angles to "guide" the single-qubit rotations as follows:

- apply Hadamard to the initial state $|0\rangle \otimes ... \otimes |0\rangle$ to transform it into an unbiased state

$$(H|0\rangle)^{\otimes N} = \frac{1}{\sqrt{2^N}}(|0\rangle \otimes ... \otimes |0\rangle + ... + |1\rangle \otimes ... \otimes |1\rangle) = \frac{1}{\sqrt{2^N}} \sum_{i=0}^{2^N-1} |i\rangle$$

Encoding Layer - Data preparation

The classical input vector is encoded into rotation angles to "guide" the single-qubit rotations as follows:

- apply Hadamard to the initial state $|0\rangle \otimes ... \otimes |0\rangle$ to transform it into an unbiased state

$$(H|0\rangle)^{\otimes N} = \frac{1}{\sqrt{2^N}}(|0\rangle \otimes ... \otimes |0\rangle + ... + |1\rangle \otimes ... \otimes |1\rangle) = \frac{1}{\sqrt{2^N}} \sum_{i=0}^{2^N-1} |i\rangle$$

- generate 2N rotation angles from the N-dimensional input vector by taking

$$\theta_{i,1} = tan^{-1}(x_i),\ \theta_{i,2} = tan^{-1}(x_i^2)$$

where $\theta_{i,1}$ and $\theta_{i,2}$ are respectively the rotation angles around the y and z axis.

Variational Layer

- The encoded classical data (now quantum state) will go through an entangling layer of CNOTs and single-qubit rotation gates
- The 3 rotation angles $\{\alpha_i, \beta_i, \gamma_i\}$ are not fixed in advance
- They should be updated after every iteration via iterative optimization based on a gradient descent method.

Measurement Layer

- The end of every VQC block is a quantum measurement layer.
- With quantum simulation software such as PennyLane and IBM Qiskit, it can be calculated numerically on a classical computer, whereas with real quantum computers, such values are statistically estimated through repeated measurements, which should be in theory close to the value obtained from simulation in the zero-noise limit.
- The returned result is a fixed-length vector to be further processed on a classical computer

Optimization Procedure

The proposed optimization procedure is based on the parameter-shift method.
For example, given the expectation of an observable $\hat{B}$

$$f(x;\theta_i) = \langle 0|U_0^\dagger(x)U_i^\dagger(\theta_i)\ \hat{B}\ U_i(\theta_i)U_0(x)|0\rangle = \langle x|U_i^\dagger(\theta_i)\ \hat{B}\ U_i(\theta_i)|x\rangle$$

where

- x is the input value
- $U_0(x)$ is the state preparation routine to encode x
- $U_i(\theta_i)$ is the single-qubit rotation generated by the Pauli operators

Then the gradient of f w.r.t. θ_i is nothing but

$$\nabla_{\theta_i} f(x;\theta_i) = \frac{1}{2}\left[f\left(x;\theta_i + \frac{\pi}{2}\right) - f\left(x;\theta_i - \frac{\pi}{2}\right)\right]$$

Demo Time!

Conclusion

Summary - Takeaways

- Quantum computing offers interesting perspectives in several fields (Machine Learning, Cryptography, Quantum Internet, Business, ...)
- It's too early to put anything in production, but there's a lot of research in the field
- Quantum computing will be a parallel form of computing with classical computing — not a replacement
- Every business should get prepared in advance for it and its potentially outbreaking results

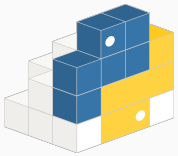
Summary

All the material used for this presentation is available at the following link:

`https://github.com/mspronesti/talk-qml-amadeus`

Summary

`https://github.com/mspronesti/qlearnkit`



```
$ pip install qlearnkit
```

`https://mspronesti.github.io/qlearnkit`

Thanks for your attention!

Questions ? ☺

References i


V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Száva, and N. Killoran.
Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2018.


S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang.
Quantum long short-term memory, 2020.


M. S. A. et al.
Qiskit: An open-source framework for quantum computing, 2021.

References ii


V. Havlicek, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta.
Supervised learning with quantum enhanced feature spaces, Jun 2018.


F. Phillipson.
Quantum machine learning: Benefits and practical examples.
2020.


P. Rebentrost, M. Mohseni, and S. Lloyd.
Quantum support vector machine for big data classification.
July 2013.


J. Suykens and J. Vandewalle.
Least squares support vector machine classifiers.
Neural Processing Letters, 9:293–300, 06 1999.

References iii


Y. Zhang and Q. Ni.
Recent advances in quantum machine learning.
2020.