A Glimpse into Quantum-enhanced Machine Learning Solutions

Massimiliano Pronesti, Federico Tiblias, Giulio Corallo

April 8, 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 going 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 going Quantum ?

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?

Why going Quantum ?

For 2 people

2 Total combination.

Why going Quantum ?

For 2 people

2 Total combination.

For 5 people

120 Total combination.

Why going Quantum ?

For 2 people

2 Total combination.

For 5 people

120 Total combination.

For 10 people

Over 3 Million of total combination!!!

Why going Quantum ?

For 2 people

2 Total combination.

For 5 people

120 Total combination.

For 10 people

Over 3 Million of total combination!!!

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

Why going Quantum ?

For 2 people

2 Total combination.

For 5 people

120 Total combination.

For 10 people

Over 3 Million of total combination!!!

- 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 is Quantum faster?

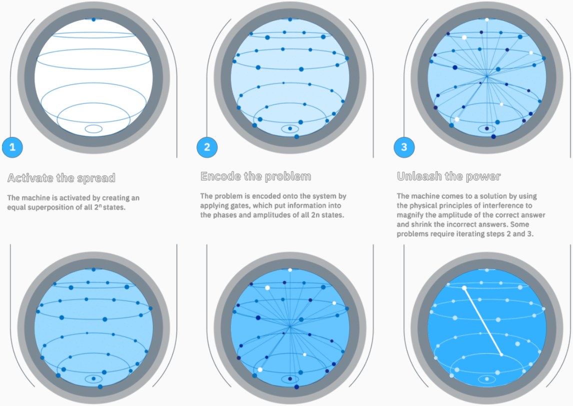


Figure 1: Quantum Supremacy

- Quantum computers can create vast multidimensional spaces in which to represent these very large problems. Classical supercomputers cannot do this.

Why is Quantum faster?

- 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.

Quantum Computer

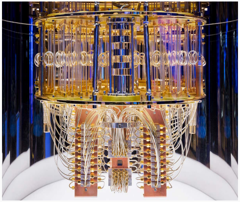


Figure 2: IBM Quantum Computer

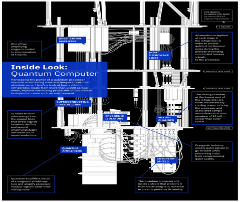


Figure 3: Inside Look

A quantum computer is a physical realisation of a quantum Turing machine supported by quantum mechanical processes which are modelled as physical qubits and abstracted as logical qubits.

How is a Quantum computer programmed?

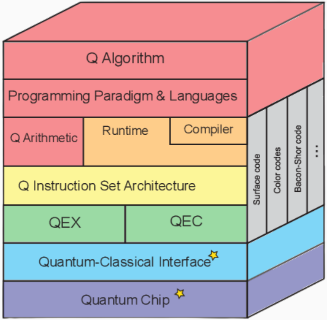


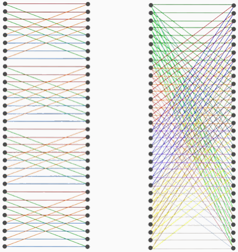
Figure 4: Quantum Computer Architecture

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

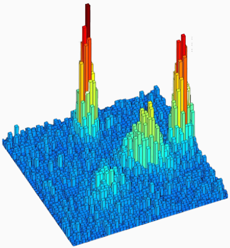
How is a Quantum computer programmed?

- Quantum Circuits are constructed from Quantum Registers.
- Quantum Register is a type of circuit construction from logical qubits.
- Logical Qubits can create different permutations and combinations of physical qubit manifestations.

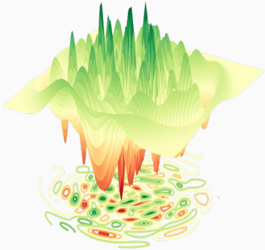
What are Quantum Computers good at ?



(a) Linear Algebra



(b) Sampling



(c) Optimization

A Growing Interest in the field

cose da dire:

- tanti framework (qiskit, pennylane, cirq, tensorflow quantum, ...)
- tanti paperi (con plot della crescita 2012-2021)
- tutti (Microsoft, Google, IBM, NVidia ...) hanno/vogliono un computer quantistico
- tanta ricerca nel settore Quantum Programming Languages

Quantum Computing Foundations

Bra-ket notation

Useful for representing quantum systems

$$\text{Bra} = \text{Row} \qquad\qquad \text{Ket} = \text{Column}$$

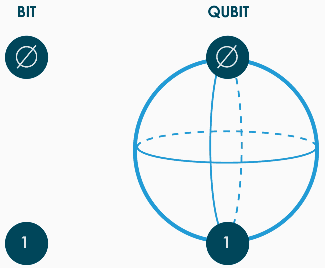
$$\langle A| = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots \end{bmatrix} \qquad\qquad |B\rangle = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \end{bmatrix}$$

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, Schrodinger'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.

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

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

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 samples, m: number of features)

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 samples, m: number of features)

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 samples, m: number of features)

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

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

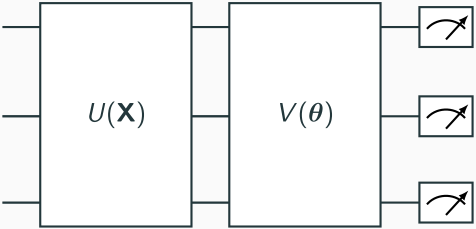


Figure 5: A generic Variational Quantum Circuit with 3 inputs

Architecture

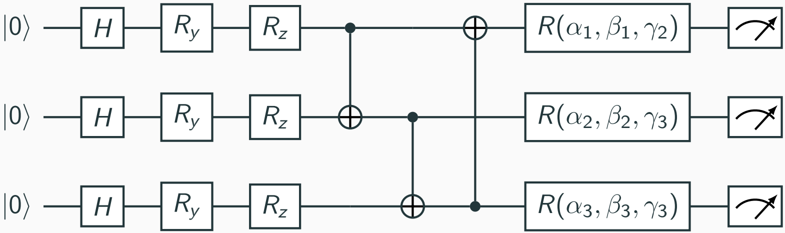


Figure 6: Single-layer Quantum LSTM cell

Architecture

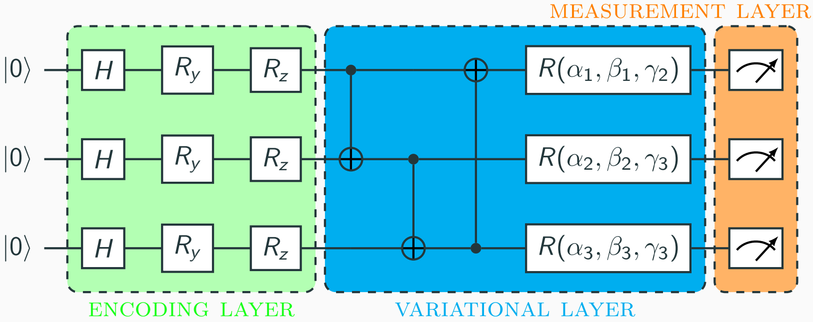


Figure 6: single-layer Quantum LSTM cell

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

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

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

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

- 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 rotatio angles $\{\alpha_i, \beta_i, \gamma_i\}$ are not fixed in advance, rather they should be updated after every iteration via iterative optimization based on a gradient descent method.

Measurement Layer

Write me!

Optimization Procedure

Write me!

References

Some references to showcase [allowframebreaks] [4, 2, 5, 1, 3]

Conclusion

Summary

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

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

Thanks for you attention!

Any Question ?

P. Erdős.
A selection of problems and results in combinatorics.
In *Recent trends in combinatorics (Matrahaza, 1995)*, pages 1–6.
Cambridge Univ. Press, Cambridge, 1995.

R. Graham, D. Knuth, and O. Patashnik.
Concrete mathematics.
Addison-Wesley, Reading, MA, 1989.

G. D. Greenwade.
The Comprehensive Tex Archive Network (CTAN).
TUGBoat, 14(3):342–351, 1993.

D. Knuth.
Two notes on notation.
Amer. Math. Monthly, 99:403–422, 1992.

H. Simpson.
Proof of the Riemann Hypothesis.
preprint (2003), available at
`http://www.math.drofnats.edu/riemann.ps`, 2003.