Brief Dissertation on Quantum Enhanced Reinforcement Learning

Simone Caldarella

DISI, University of Trento, Povo (TN), 38123 Italy .

simone.caldarella@studenti.unitn.it

Abstract

Reinforcement Learning (RL) has emerged as a powerful framework for training intelligent agents to make optimal decisions in complex environments. In recent years, there has been a growing interest in enhancing classical Machine Learning algorithms through the use of Quantum Computers, leveraging the unique properties of quantum systems to speed up classical algorithms. More recent works focused on the more specific task of exploiting quantum speed up to improve the learning process in RL setting. Among the other, The work [1] by Saggio et al., have shown that the introduction of an agent that possesses the ability to interact with its environment using both classical and quantum-mechanical methods can accelerate the learning process of the agent. In this dissertation I propose a brief introduction to both Reinforcement Learning and Quantum Computing, followed by the description of my implementation of the work accomplished by Saggio et al. ([1]). I then show a comparison between my results and those presented in the paper and I conclude discussing my hopes and doubts related to the proposed framework.

Keywords: Quantum Machine Learning, Reinforcement Learning, Quantum Mechanics

1 Introduction

Reinforcement learning (RL) is a branch of machine learning that focuses on how an agent can learn to make sequential decisions in an environment to maximize a notion of cumulative reward. RL is inspired by the concept of learning through interactions and feedbacks, inspired by how humans and animals learn from their experiences.

In RL setting, an agent interacts with an environment in a series of discrete time steps. At each time step, the agent observes the current state of the environment and takes an action based on its learned policy. The environment then transitions to a new state, and the agent receives a reward that reflect the desirability of the transition. The objective of the agent is to learn a policy-a mapping from states to actions—that maximizes the cumulative reward it receives over time. The agent's goal is to explore and learn through its interactions with the environment in order to make informed decisions that lead to higher rewards over time.

Reinforcement learning has been successfully applied to a wide range of domains, including robotics, game playing, finance, and autonomous systems. Notable RL algorithms include Q-learning, SARSA, and deep reinforcement learning methods such as Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO). Recently a training framework called Reinforcement Learning from Human Feedback, which mix the use of PPO and Human Labels to fine-tune Large Language Models, achieved fame due to the incredible performance that the fine-tuned model, i.e. GPT-3.5 obtained [2].

Overall, reinforcement learning offers a powerful framework for training intelligent agents to learn through trial and error, enabling them to make sequential decisions and optimize their behavior to achieve specific goals in complex and dynamic environments.

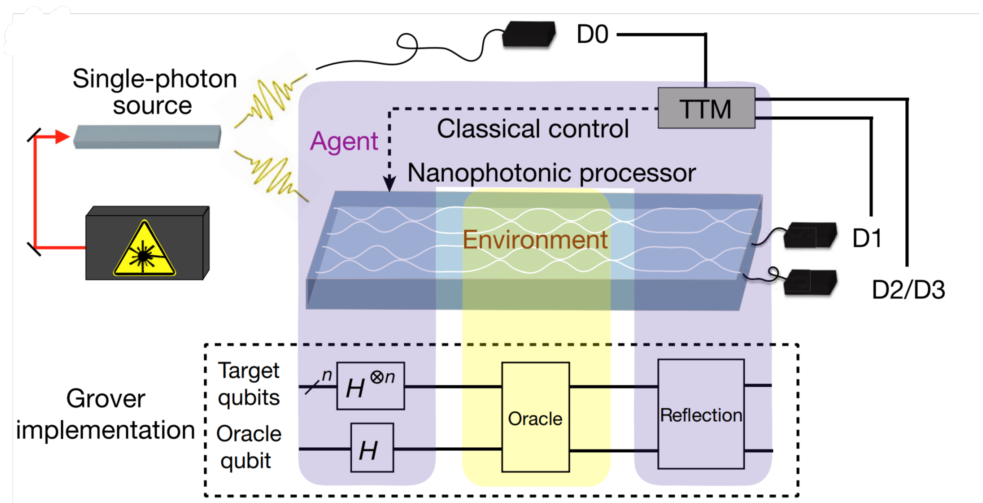


Fig. 1: Newly proposed framework of a quantum enhanced reinforcement learning agent.

Quantum computing is an emerging field that harnesses the principles of quantum mechanics to process and manipulate information. While classical computers use bits to represent and manipulate data as either 0 or 1, quantum computers use quantum bits, or qubits, which can exist in a superposition of both 0 and 1 states simultaneously.

This unique property of qubits enables quantum computers to perform certain calculations much more efficiently than classical computers. Quantum computers leverage quantum phenomena such as superposition and entanglement to perform parallel computations and solve problems that are intractable for classical computers.

Superposition allows qubits to represent multiple states simultaneously, expanding the computational capacity exponentially as more qubits are added. This enables quantum computers to perform complex computations in parallel, potentially speeding up tasks such as factorizing large numbers or searching vast databases.

Entanglement, on the other hand, establishes a correlation between qubits, even when physically separated. Manipulating one qubit can instantly affect the state of another qubit through entanglement. This property is crucial for performing quantum operations collectively and can facilitate more powerful algorithms and communication protocols.

Quantum computing is still in its early stages, and building practical, large-scale quantum computers remains a significant technical challenge. Quantum systems are extremely delicate and susceptible to errors caused by environmental interference, or noise, which can disrupt the fragile quantum states. Researchers are actively working on developing error-correcting codes and fault-tolerant architectures to mitigate these challenges and build robust quantum computers.

In recent years, there has been growing interest in exploiting quantum computing advantages to speed-up RL algorithms. Combining reinforcement learning, which focuses on decision-making and optimization, with the unique capabilities of quantum computing could unlock new avenues for solving complex problems and enhancing the efficiency of learning algorithms. In this new field, one of the most promising work is the research accomplished by Saggio et al. [1], which provides an experimental implementation of a mixed framework that aims to make advantage of quantum properties to parallelize and speed-up the initial exploration phase of a quantum-aware RL agent in a mixed classical-quantum environment. A teaser of the setting, that will be explained in the section 4, is provided in fig. 1.

The goals of this dissertation involves 1) providing a necessary and sufficient background on the topics by proposing a narrow introduction to the fields of Reinforcement 2 and Quantum Computing 3 and 2) proposing a Python implementation of the experiment proposed in [1], including comments and insights on the practical use of theoretical tool employed.

2 Reinforcement Learning

Reinforcement learning is in its essence the process of learning what to act at a certain moment as to maximize a numerical reward signal. The learner is not aware about which actions to take, but instead must discover which actions yield the most reward by a trials and errors tests. Moreover, differently from machine learning, the main goal in RL is not to solely learn a stateless model of the best action from a given state by looking at many labeled examples, but to learn a stateful (and time variant) decision making model from which the agent knows which action should be performed at a certain time to increase the reward, given all the previous actions (with discounted importance over time), i.e. the story of the agent, and the expected next actions.

The learning process strictly relies on the interaction with the environment, which provide the rewards and the search space. Philosophically speaking we could think to the environment as a co-star of the agent in the learning process, which would be impossible without its *weak supervision*. An idea of the continuous (but time discrete) interaction of an agent with an environment is provided in fig. 2, in which the classical RL loop is depicted, along with a psichological interpretation of an agent (dog) that learn through a supervision given by the environment (human).

In this first section I formalize the setting of reinforcement learning introducing the famous decision making framework named Markov Decision Process and propose a brief discussion about the relation between exploration and exploitation.

For the reader who wants to delve deeper in this intricate world I cannot avoid to suggest the famous book "Introduction to Reinforcement Learning" [3] by Sutton and Barto.

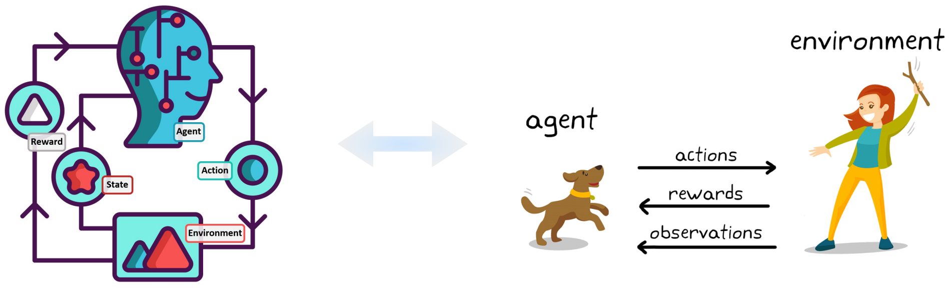


Fig. 2: Exemplification of the RL loop, which consists in an interaction of an agent with the environment, accomplished through an action performed by the agent, followed by an update of the current agent state and a reward given by the environment.

2.1 Markov Decision Process

A Markov decision process (MDP) is a mathematical framework for modeling stochastic decision-making in dynamic systems. MDPs are used when the outcomes of decisions are uncertain or controlled by a decision maker who makes sequential choices over time. By taking in account the current state and environment of the system, MDPs aims to evaluate the optimal actions for the decision maker to take.

MDPs have been in existence since the early 1950s, named after the Russian mathematician Andrey Markov, who made significant contributions to stochastic processes. Initially, MDPs were used to address challenges related to inventory management, queuing optimization, and routing. Presently, MDPs find applications in various fields, including dynamic programming, robotics, automatic control, economics, and manufacturing.

In the field of artificial intelligence (AI), MDPs are employed to model sequential decision-making scenarios with probabilistic dynamics. They enable the design of intelligent machines or agents that operate in environments where actions can yield uncertain outcomes. This make MDPs perfectly suitable for Reinforcement Learning setting.

Let's now formalize the MDP framework. First of all, a MDP requires the following components to be defined:

- $\mathbf{S} \rightarrow$ set of states that define an environment
- $\mathbf{A} \rightarrow$ set of actions an agent can perform
- $\mathbf{P} \rightarrow$ set of transition probabilities each state to the next one
- $\mathbf{R} \rightarrow$ set of rewards, one for each state

Without any further elements there could not be any learning. To achieve so two other concepts are required: a *policy* and a *value function*.

A policy is mappingo from a state to an action that is expected to maximize the reward. Intuitively, the policy

$$\Pi : \mathbf{S} \rightarrow \mathbf{A}$$

can be seen as the optimal directions, learned by the agent, used to efficiently move in the environment.

A policy is learnt through the maximization of a value function

$$V(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t\right]$$

which models the reward return at each specific state. The above formulation, peculiar because of the expected sum of discounted future rewards, can be expanded as a sum of the reward of the current state and the discounted reward value of the next state. This new formulation lead to the famous value function called Bellman equation [4]:

$$V^*(s) = \max_a \left[R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s')\right]$$

which embeds in an elegant and recursive way the heart of the optimisation problem. The optimal value function can be found through an iterative optimization process such as temporal-difference learning or dynamic programming.

Going back to the policy, it is now possible to define the optimal policy as:

$$\Pi^*(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} V^(s') \right]$$

Since the policy it is consequence of the current state, usually it is updated iteratively and continuatively over learning.

2.2 Exploration vs Exploitation

To have a clear idea on the definition and consequences of this trade-off let's introduce the buffet metaphor.

Imagine you're at a buffet with an incredible assortment of dishes. On one hand, you could decide to *play it safe* and stick to the dishes you already know and love, such as pizza. This is exploitation, because you're exploiting your *existing knowledge* and preferences to enjoy the food you're already familiar with.

On the other hand, you could choose to be adventurous and try out new dishes that you've never tasted before. Maybe there's a mysterious-looking dish with an exotic name, or a chef's special creation that sounds intriguing. This is exploration, because you're venturing into the unknown, taking risks, and opening yourself up to new experiences and flavors.

Now, here's the trade-off: if you focus too much on exploration and constantly try new dishes, you might end up wasting your time and stomach space on something you don't enjoy. You may come across some disappointments or even encounter the dreaded "food regret" when you realize you should have stuck to what you already knew you liked. This is the risk of exploration.

On the other hand, if you exclusively exploit your existing knowledge and preferences, you'll miss out on the potential delights and surprises that lie in the uncharted territory of the buffet. You might get stuck in a culinary rut, always ordering the same dishes and never discovering new flavors that could become your new favorites. This is the opportunity cost of exploitation.

So, the trade-off between exploration and exploitation is like standing at the buffet, trying to strike the right balance between sticking with the dishes you know and love and venturing into the unknown to discover new culinary treasures.

Going back to our theoretical study, find the balance between exploration and exploitation is the goal in here, in order to find the best solution in an optimal amount of time. In a world with an infinite time, exploitation would be quite useless since you could have the possibility to explore all the solutions. However we live in a world where time is finite and expensive, making necessary to reduce as much as needed-but not avoiding-the exploration phase. Indeed the aim of combining Quantum Computing with Reinforcement Learning is to make the exploration phase cheaper time-wise, leading to a fast convergence of the agent to an optimal subset of solutions.

3 Quantum Computing

A Quantum Computer exploit quantum mechanical phenomena and uses specialized hardware. It operates at small scales where matter exhibits both particle and wave properties and unlike classical computers, quantum computers can perform certain calculations exponentially faster. Quantum Computing is an intersectional branch that combine together computer science and quantum physics in order to study and develop algorithms that can be run by a quantum computer.

In this section I introduce the Qubit, a building block of quantum computing, and the concept of Entanglement. Then I provide a brief introduction to quantum circuits, which are the implementation of quantum algorithms, and present a famous quantum algorithm called "Grover's Algorithm" [5], which is at the core of Experimental Quantum Speed-Up in Reinforcement Learning agents proposed in [1].

N.B Bra-Ket notations, developed by the great P.A.M. Dirac, is assumed to be known by the reader, so it will not be covered in this dissertation. For those who are not familiar with it I can suggest to refer to the original book [6] in which it was introduced.

3.1 Qubit

The basic unit of information in quantum computing is the qubit, which is the quantum counterpart of the bit.

Qubits can exist in *superposition*, being in multiple states simultaneously, when non-observed. To easily understand the idea of a qubit we can take a look at the fig. 3, in which both classical and quantum bit are compared. While a bit can have a value of either 0 or 1, qubit value is given by the superposition of its possible states (basis states), i.e. $|0\rangle$ and $|1\rangle$ in the fig. 3.

It is importante to underline that being in multiple states simultaneously doesn't imply the possibility to observe the continuous space in which the qubit live nor the superposed state, but just to perform the quantum operations on those state. This means that our algorithm can act on both state $|0\rangle$ and $|1\rangle$, but the output of the algorithm that we can observe will only be one of the two states.

Let's take a loot to the main advantages within this quantum framework:

- given that a qubit can represent simultaneosuly 2 states, with N qubits we can represent simultaneously 2^N states. In classical computing, despite the representation power of N bit is 2^N as well, we can make use of only 1 state/combination of possible bit values at a time. To act on all the 2^N states we would need $N * 2^N$ bits, which means an exponential grows in space complexity w.r.t the qubit representation;
- since the inputs and the outputs lives in a classical world, it is "easy" to integrate quantum algorithms within a classical framework in order to speed up certain computations that would represent a bottleneck in the classical system, in term of time complexity, leaving unaltered the rest of the system.

Talking about observations, I anticipated that the process of observing a qubit, and thus a state vector, doesn't allows to observe the entire superposed, but instead only

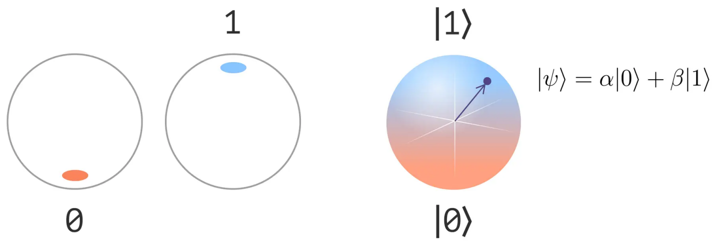


Fig. 3: Comparison between discrete states in a classical Bit (left) and the Bloch Sphere representing the Qubit as a superposition of its basis states (right).

one of the basis states in which the original state reside. It is said that the original state wave collapse into one of its observable states and this process can be theoretically studied through the prediction of the probability to observe one or another state:

$$given\ |\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$\mathcal{P}(|0\rangle) = |\langle\psi|0\rangle|^2$$

and

$$\mathcal{P}(|1\rangle) = |\langle\psi|1\rangle|^2$$

Without adding too many details now, the coefficients α and β in the superposition represent the squared probability of observing the related state of the state vector.

3.2 Entanglement

The concept of Entanglement, often misunderstood, refers to the situations where the properties of two or more particles (qubit in our quantum computing setting) become intrinsically correlated, allowing for interesting corollary behaviours. This correlation implies that when one of the entangled particle is observed all the particles state collapse, regarless their relative distance. This phenomenon, often referred to as ”spooky action at a distance,” challenges our classical intuition but has been experimentally verified. Entanglement plays a crucial role in quantum computing and communication, as it allows for the creation of qubits with enhanced computational power and secure transmission of information through quantum cryptography. Understanding and harnessing entanglement is a key focus in the field of quantum physics and has far-reaching implications for technology and our understanding of the nature of reality.

To better understood the pros and cons of this crucial property, here is given an example of an entangled systems:

$$|\psi_{entangled}\rangle = \frac{1}{\sqrt(2)}(|0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle)$$

First, we can notice that we have a superposition of two states, however this states are just a subset of all the states that can be formed through the complete tensor product of two qubits, in which the state $|psi\rangle$ lies. Having a "full" tensor product would imply to have a state vector in the form

$$|\psi_{separable}\rangle = |qubit_1\rangle \otimes |qubit_2\rangle = \frac{1}{2}(|0\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle$$

To understand the difference between the $|\psi_{entangled}\rangle$ and $|\psi_{separable}\rangle$ we can simply ask ourselves: what happens if we measure the first qubit?

Let's assume that the observation of $|qubit_1\rangle$ lead to the state $|0\rangle$. Having observed the first qubit of $|\psi_{entangled}\rangle$ in the state $|0\rangle$ means that we have no doubts regarding the state of $|qubit_2\rangle$, which will be in the state $|1\rangle$ with a probability of $\mathcal{P}(|qubit_2\rangle) = 1$. This implies that the collapse of the state of the first qubit led to the collapse of the state of the second qubit, hence the collapse of the system, without the need to observing the second qubit itself.

On the other hand, assuming the same collapse of $|qubit_1\rangle$ in $|\psi_{entangled}\rangle$, the results are quiet different. Specifically we don't have any certainty about the state of the second qubit, which could evenly be $|0\rangle$ as well as $|1\rangle$, and thus we require a further observation on the second qubit to know its state. This means that the first observation didn't make the second qubit state to collapse. This second type of mixed state, whose behaviour is somewhat opposed to the entangled ones, is referred to as *separable*.

From a an algorithmic perspective, one of the main use of entanglement is to introduce conditions in your program flow. We can indeed think at $|qubit_1\rangle$ state in $|\psi_{entangled}\rangle$ as a condition A that, if verified, should implies a condition B. The verification process of the condition A is simply the observation of the $|qubit_1\rangle$, while the condition B is the resulted collapsed state of the second qubit. In programming the above statement is written as:

```
if state(qubit_1) == 1:
    state(qubit_2) = 0
else
    state(qubit_2) = 1
```

3.3 Quantum Circuit

A quantum circuit is a computational model for manipulating qubits through a series of quantum gates, which are the building blocks of quantum computation. These gates are analogous to the classical logic gates used in traditional computing but operate on quantum bits.

Quantum circuits (generic architecture in fig. 4) are constructed by connecting quantum gates in a specific sequence, with each gate representing a transformation on the qubits. These gates can perform various operations, such as rotations, flips, and entanglement operations, which manipulate the quantum state of the qubits. By carefully designing and controlling the sequence of gates, a quantum circuit can implement complex algorithms and solve problems that are difficult or even intractable for classical computers.

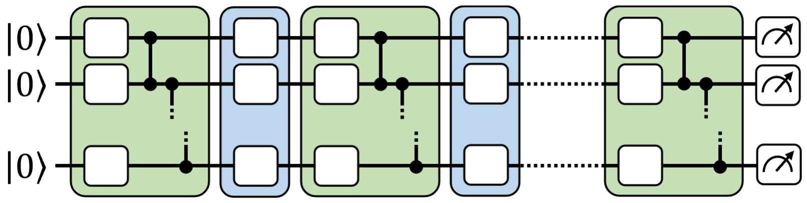


Fig. 4: A generic quantum circuit composed of a set of qubits in the $|0\rangle$ state (left), a set of operations performed through unitaries (center) and an observation process (right).

The set of operations that a quantum gate can perform is the subset of quantum operations that are also *Unitary*, which is consequence of their implementation achieved via the action of a Hamiltonian for a specific time, which leads to a unitary time evolution. A Unitary operator $\mathcal{U} : \mathcal{H} \to \mathcal{H}$ is an operator for which it stands $\mathcal{U}^*\mathcal{U} = \mathcal{U}\mathcal{U}^* = \mathcal{I}$, which translates into the fact that such operators have to conserve properties between elements of the space.

Being Unitary operators, linearity and invertibility are guaranteed, differently from classical computing gates, and allowing for different assumptions w.r.t classical approaches.

In this section, I will not provide a review of the quantum gates currently used, but an extensive list can be found in the summary [7] proposed by Qiskit team.

To execute a quantum circuit, it must be implemented on a physical quantum computer or simulated using a quantum simulator. Physical quantum computers employ different technologies, such as superconducting circuits or trapped ions, to realize and manipulate qubits. The operations performed by the quantum gates are translated into physical manipulations on these qubits, utilizing quantum effects to achieve the desired computations.

However, quantum circuits are also susceptible to various sources of noise and errors, which can cause the fragile quantum information to decohere and lose its quantum properties. Therefore, error correction techniques and quantum error correction codes are crucial to protect the information and maintain the integrity of the computations.

3.4 Grover's Algorithm

The Grover's algorithm [5] is a famous quantum algorithm developed by Lov Grover in 1996, which provides a powerful tool for searching through an unsorted database, offering a significant speed-up over classical algorithms for this specific task.

The fundamental problem that the Grover's algorithm addresses is the following: given an unsorted database of N items, how can we efficiently find a specific item with only a few queries to the database?

In a classical setting, the best-known algorithm (i.e. linear search) requires, in the worst-case, $N-1$ queries to find the desired item. Grover's algorithm requires instead $\sqrt{N}$ queries to find an item, leading to a quadratic speedup compared to the classical case. The core of the algorithm is the technique called *amplitude amplification*, which enhances the probability of finding the desired item in each iteration. In essence, the Grover's algorithm leverages the principles of superposition and interference in quantum computing to rapidly mvoing towards the solution.

The implications of the Grover's algorithm extend beyond database searching. It has applications in areas such as cryptography, optimization, and machine learning. Furthermore, the algorithm has played a crucial role in highlighting the power of quantum computing and has sparked advancements in the development of quantum algorithms for various computational problems.

Formalising what said before, Grover's algorithm is an iterative algorithm that aims to solve the task of finding an input x_0 in a search space with n bits, such that a given classical function $f(x)$ evaluates x_0 equal to 1. This means that our problem can be stated as follows:

$$\mathbf{Grover}(\{0,1\}^n) = \{x_0 | f(x_0) = 1\}$$

$$with\ f(x) : \{0,1\}^n \rightarrow \{0,1\}$$

Let's now delve into the algorithm by understanding what an iteration (depicted in fig. [5]) does:

1. it is given have an arbitrary state vector in the 2-dim Hilbert space spanned by two orthonormal vectors $\rightarrow\ |\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, and we want to maximize the probability of observing the state $|1\rangle$, i.e. $\mathcal{P}(|1\rangle) = |\langle\psi|1\rangle|^2 = |\beta|^2$
   - $|0\rangle$ and $|1\rangle$ are just semplifications/proxies that can imply a superposition of more state each one
2. the state vector $|\psi\rangle$ is then forwarded to an *Oracle* which perform a phase flip of the state to find (1 in this example) $\rightarrow\ Oracle(|\psi\rangle) = |\psi_f\rangle = \alpha|0\rangle - \beta|1\rangle$
   - the use of an *Oracle* is somewhat controversial in terms of quantum speed-up since theoretical implementations of Grover's Algorithms don't take in account *Oracle*'s complexity
   - the phase flip, which is performed through a Unitary Operator usually implemented through a controlled-Z gate, can be seen as a reflection over the $|0\rangle$ state, i.e. the set of components that are orthogonal to the searched one
3. the so called *diffusion*, is the heart of the Grover's algorithm. It consists on a *reflection* of $|\psi_f\rangle$ w.r.t. $|\psi\rangle$, which is accomplish through the reflection operator $\rightarrow\ R_{|\psi\rangle} = 2|\psi\rangle\langle\psi| - \mathcal{I}$ applied to $|\psi_f\rangle$, leading to $|\psi_{fr}\rangle$

To easily understand what state ψ_{fr} is, we have to change the representation of the basis' coefficients α and β as follows:

$$\alpha = cos(\theta)\ \beta = sin(\theta)$$

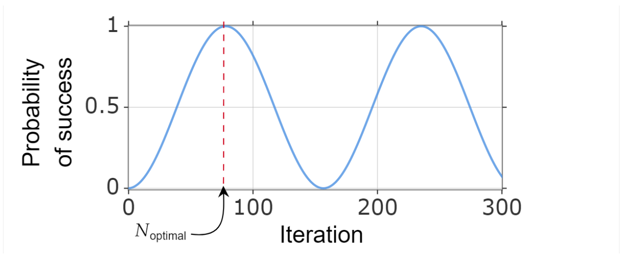


Fig. 5: Oscillating behaviour of Grover's Algorithm

where θ is the angle between the statevector and the first component versor, i.e. $\theta = arccos(\langle\psi|1\rangle)$.

This representation help us to better visualize the geometric interpretation of what happened with the first Grover iteration (shown in fig. 6, with $|\psi\rangle = |all\rangle$ $|0\rangle = |bad\rangle$ $|1\rangle = |good\rangle$, taken from [8]). The two unitaries ($Oracle$ and Reflection) acted as a rotation matrix, rotating the initial $|\psi\rangle$ in the unitary circumference in the space of $|0\rangle$ and $|1\rangle$. With the first phase flip (reflection w.r.t. $|0\rangle$), $|\psi\rangle$ is rotated by an angle equal to -2θ, while with the reflection over $|psi\rangle$, during the diffusion step, $|psi_f\rangle$ is rotated in the opposite direction of an angle equal to 4θ. After this iteration, the probability to observe the state $|1\rangle$ become $\mathcal{P}(|1\rangle) = |\langle\psi_{fr}|1\rangle|^2 = sin^2(3\,\theta) = sin^2(3\,arccos(\alpha))$.

Finally, it is now possible to extend what has been said so far studying the behaviour of the algorithm over a variable number k of iterations. Since each iteration rotate the state vector by an angle equal to 3 times the initial one, we can easily compute the probability of observing the $|1\rangle$ as $\mathcal{P}(|1\rangle; K) = sin^2((2k+1)\,\theta) = sin^2((2k+1)\,arcsin(\beta))$.

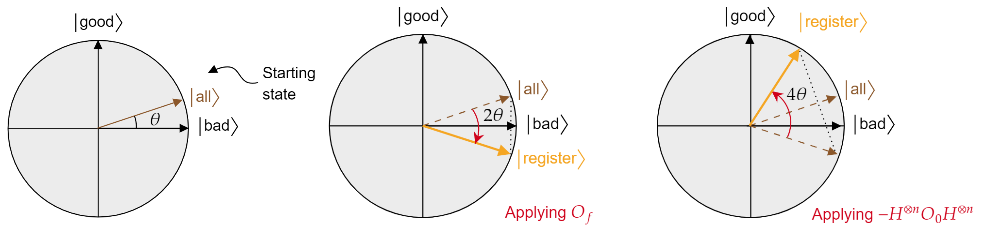


Fig. 6: Example of Grover's algorithm geometric interpretation. Starting from the first state (left) on which it is applied a reflection only on the orthogonal direction of the "bad" states (center), followed by a reflection on the starting state that leads to an amplification over the "good" component.

As the function $\mathcal{P}(.;k)$ evolves over time as a sin^2 (fig. 5), it doesn't converge to an optimal solution. We can thus compute the optimal number of iterations $N_{optimal}$ that allows to maximizing the probability of observing the desired state. Knowing that the first theoretical maximum is reached for $\theta = \pi/2$ we can set an equation w.r.t.

$N_{optimal}$, whose solution is the optimal number of iterations to reach the maximum probability, which will live in the neighbours of $\theta = \pi/2$:

$$\frac{\pi}{2} = (2N_{optimal} + 1)\,\theta$$

$$N_{optimal} = \frac{\pi - 2\theta}{4\theta}$$

$$with\ \theta = arcsin(\beta)$$

Rounding $N_{optimal}$ to the closest integer it is possible to find the optimal number of iterations that needs to be performed in order to maximize the probability of observing the desired state.

4 Experimental Quantum Speed-Up in Reinforcement Learning Agents

This idea behind the work relies on considering a novel RL setting in which agent can interact both classicaly and quantumly with the environment, leveraging the unique properties of quantum mechanics to speed-up their learning. They do so by introducing a quantum-enhanced hybrid agent capable of quantum as well as classical information transfer.

In this section is presented the setting of the paper followed by the explanation of my implementation of the approach, in python, and by the results obtained.

4.1 Setting

The setting of this experiment is as simple as interesting. In fig. 7 we can se the full behaviour of the forward pass in both classical epoch and quantum epoch.

In the classical epoch, the agent prepare the state $|a\rangle|0\rangle$, in which the action a is sampled from its policy π with a probability $p(a)$. The agent then receives the reward with a probability equal to $\varepsilon = |\beta|^2$.

During a quantum epoch things get more complicated. The agent starts the process by preparing the state $|\psi\rangle_A|-\rangle_R$, where

$$|\psi\rangle_A = \sum_a \sqrt{p(a)}|a\rangle = cos(\xi)|lose\rangle_a + sin(\xi)|win\rangle_a = \alpha|0\rangle + \beta|1\rangle$$

represents the superposition of all actions with their probabilities and $|-\rangle_R = \frac{\sqrt{2}}{2}(|0\rangle_R + |1\rangle)_R$.

The environment applies the *Oracle* Unitary to the prepared state

$$U_e|a\rangle_A|0\rangle_R = \begin{cases} |a\rangle_A|1\rangle_R & \text{if } r(a) > 0 \\ |a\rangle_A|0\rangle_R & \text{if } r(a) = 0 \end{cases} \tag{1}$$

which results in a bit flip of the winning state vector's phase

$$U_e|\psi\rangle_A|-\rangle_R = [cos(\xi)|lose\rangle_a - sin(\xi)|win\rangle_a]|-\rangle_R$$

As explained in the sub section 3.4, the Grover's algorithm diffusion part is then performed as a reflection of the initial state $|\psi\rangle_A$ as

$$U_R = 2|\psi\rangle\langle\psi|_A - \mathbb{I}_A$$

leading to an increased probability of sampling the winning state in the next classical epoch equal to $sin^2(3\xi)$. This increased probability is then used to increase the rate of rewards obtained by the agent in the following step, feeding a positive feedback-loop that aims to a faster increase in the agent's winning probability.

Due to the oscillating behaviour of Grover algorithm, the key of the approach relies on stopping the quantum strategy (alternating quantum and classical epoch) when the maximum probability is first reached and continue with the classical ones. The optimal point in which this strategy change should happen is given by the following equation

$$sin^2(\xi) = \frac{sin^2(3\xi)}{2}$$

whose solution represents the value of ξ when the winning probability in the classical strategy is equal to half of the winning probability in the quantum strategy (explanations regarding the use of half of probability are given in the subsec. 4.2.2.

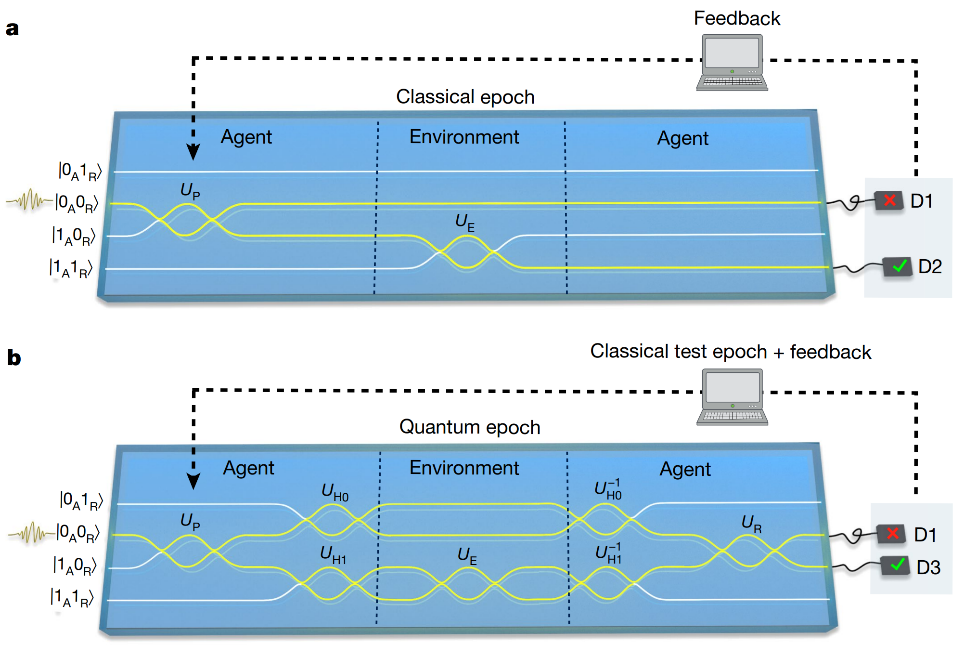


Fig. 7: Experimental setting of a quantum epoch (a) and a classical epoch (b).

4.2 Implementation details

I now describe the central components of the implementated solution, along with some details and motivations for the design choices.

Regarding the full implementation, I invite the reader to check out the Jupyter Notebook in the repository [9]. The code is written in Python and makes use of the libreries Numpy and Qiskit.

4.2.1 Agent

The agent is implemented abstracting some details w.r.t. a classical reinforcement learning agent. Instead of storing all the possible moves, along with the policy, thanks to the simplicity of this toy tasks, I simply keep track of ε (the other set of actions probability is simply 1-ε), i.e. the winning probability, and j, the number of reward received. This heavily simplifies the full formulation seen in Reinforcemente Learning section 2 and by leveraging the following formula, derived in the paper, it is possible to compute the new ε given j:

$$\varepsilon_j = \frac{1+2j}{100+2j}$$

The formula is easily derived as follows. Given the definition of winning probability ϵ and the update rule for $h(a)$ after having reiceived a reward (= 1), we have:

$$\varepsilon = \sum_{\{a|p(a)>0)\}} p(a) = \frac{\sum_{\{a|p(a)>0)\}} h(a)}{\sum_{\{a\}} h(a)}$$

$$h(a) \to h(a) + \lambda r(a) \; with \; \lambda = 2$$

we can now easily see that, by assuming a start probability of $\frac{1}{100}$ for the only winning set actions and being all initial $h(a) = 1$, the above formulation of ε can be rewritten as:

$$\varepsilon_0 = \sum_{\{a|p(a)>0)\}} p(a) = \frac{1}{100}$$

We can now see that incorporating the above formula along with the updated reward one we can rewritten the ϵ for a given reward:

$$\varepsilon_1 = \frac{\sum_{\{a|p(a)>0)\}} h(a) + \lambda r(a)}{\sum_{\{a\}} h(a) + \lambda r(a)} = \frac{\sum_{\{a|p(a)>0)\}} h(a) + 2}{\sum_{\{a\}} h(a) + 2}$$

which is easily extended to the initial formula for j rewards.

4.2.2 Environment

The environment is implemented as the module that takes cares of making the agent able to move and receive the reward. For simplicity of implementation, inside the environment class can be found also the methods that implement the three different strategies and the circuit for the grover iteration. This does not directly reflect the description in the paper of the agent that perform a Grover iteration, but being an actually separate module, the code is functionally equal to adding the module to the agent.

The core of the quantum part, i.e. the quantum circuit that initialize the superposition of actions and perform the grover iteration, is implemented as follows:

1. the statevector is initialized as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \frac{99}{100}|0\rangle + \frac{1}{100}|1\rangle$, and then, after each iteration, $|psi\rangle$ is initialized as $|\psi\rangle = \sqrt{1-\varepsilon}|0\rangle + \sqrt{\varepsilon}$

2. the bit flip operation-oracle action-is implemented through a simple Pauli Z-gate

$$P_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

3. the diffusion step, performed as a reflection of the flipped statevector over the initial $|psi\rangle$, is implemented simply by computing the Unitary matrix that represents the reflection operation

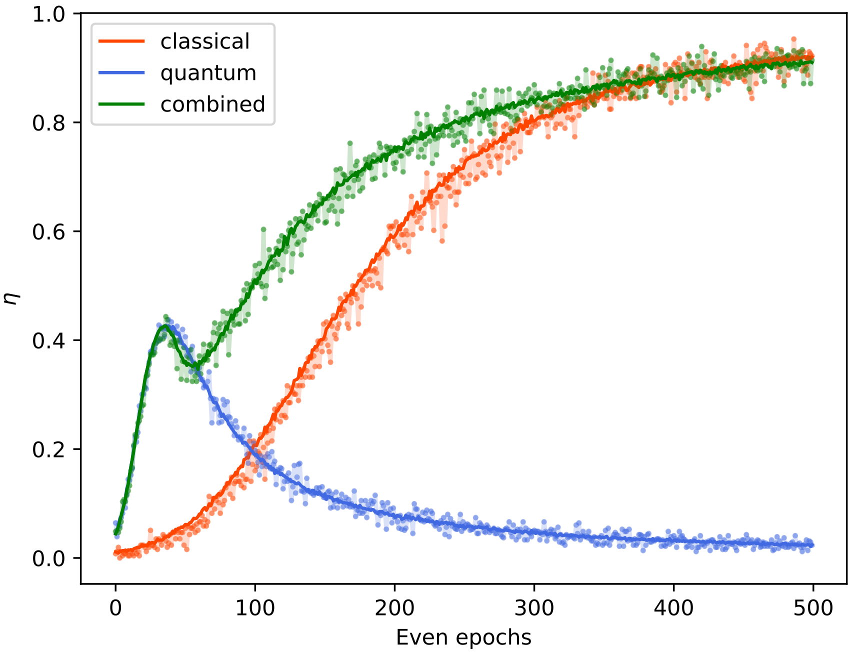
$$U_r = 2|\psi\rangle\langle\psi| - \mathbb{I}$$

It is very important to notice than the results in the paper are obtained by considering the reward different w.r.t. the different strategies. Whenever the agent employs the classical strategy the reward is equal to 2 every time is choosen the winning action. Instead, when the quantum strategy is employed, it requires a further classical strategy to actually ”test” the action with the new probability and obtain a reward. Thus, to avoid this skip in the received rewards while using quantum strategy, the reward is simply splitted in half, i.e. *distributed* over the two epochs (the classical one and the quantum one). The results are compared in the paper results plot, also visible in fig. 8b.

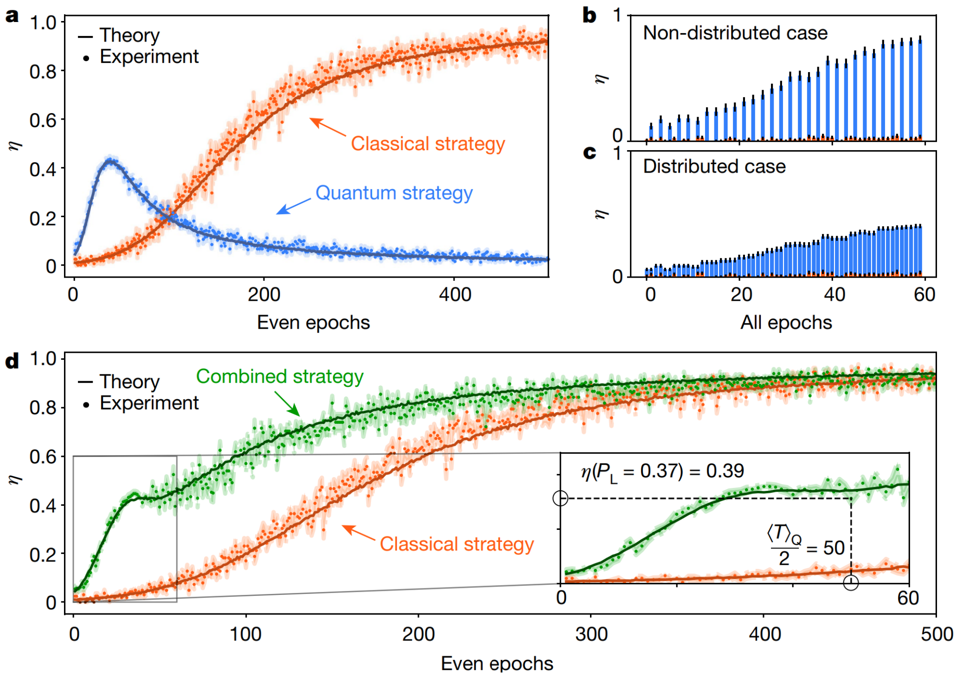
4.3 Results

The obtained results are completely similar to those obtained in the paper (not obvious). Despite the extensive comment presented in the paper, I would underline the importante of the fact that it has been assessed that employing the Grover algorithm to speed-up the initial exploration phase of the agent-the one in which it is more uncertain-is a very effective solution. Before converging to the same classical probability, the model achieve the maximum probability (considering the non-distributed case) after only sixty epochs. It then steadily decreases when the quantum strategy is turned off, but the impact of the Grover algorithm is still hitting, leading to a faster convergence to an average of 80% of winning probability almost a hundred epochs before the classical strategy.

In picture 8a it is possible to see the three strategies in the same plot, underlying the results obtained in the original paper 8b, confirming the results obtained through both experimental and theoretical tests.



(a) Results obtained with my code implementation.



(b) Experimental results by Saggio et al. [1]

Fig. 8: Average reward obtained at each even epoch by $n = 256$ agents (dots) and $n = 10000$ agents (solid line). Both plots aggregate the results obtained by the 3 different strategies to make easier to compare their behaviour. In the image above (8a) are shown the results I obtained through my Python script, while below (8b) are presented the results from the original paper [1].

5 Conclusion

In conclusion, this dissertation has explored the fascinating intersection of reinforcement learning and quantum computing. Reinforcement learning, a powerful technique in artificial intelligence, has shown tremendous potential in solving complex decision-making problems. However, the limitations of classical computing have constrained its scalability and efficiency. Quantum computing, on the other hand, leverages the principles of quantum mechanics to offer exponential computational speedups. By combining the strengths of these two fields, quantum-enhanced reinforcement learning emerges as a promising avenue for tackling intricate tasks in an accelerated manner.

The implementation and analysis of the paper on quantum-enhanced reinforcement learning [1] have shed light on the practical implications of this approach. The utilization of qubits and quantum gates to encode and manipulate information introduces a new paradigm for improving the convergence and performance of reinforcement learning algorithms. The experimental results presented in the paper have demonstrated the advantages of quantum-enhanced techniques over their classical counterparts, leading specifically to a more efficient exploration phase of an agent, that allows for a faster convergence towards the optimal solution.

However, it is important to note that the proposed setting, despite the theoretical strong proof, doesn't account for common friction points in scaling up the problem. For example, the paper doesn't take in account the complexity of the *Oracle*, which could be a crucial point in practical uses of the the proposed solution. Furthermore, to setup the quantum framework, the classical problem is downscaled to 2-degree of freedom (up-down or $|0\rangle$ - $|1\rangle$), assuming to have a complete knowledge a priori of the environment to setup the mapping between states and their membership to either the win set or the loose set.

In summary, the integration of quantum computing with reinforcement learning opens up exciting avenues for tackling complex decision-making problems. Through the implementation and analysis of a specific paper on quantum-enhanced reinforcement learning, this dissertation has contributed to the understanding and exploration of this emerging field. By continuing to push the boundaries of both quantum computing and reinforcement learning, we can pave the way for groundbreaking advancements in artificial intelligence and problem-solving capabilities.
it is given have an arbitrary state vector in the 2-dim Hilbert space spanned by two orthonormal vectors $\rightarrow$ $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, and we want to maximize the probability of observing the state $|1\rangle$, i.e. $\mathcal{P}(|1\rangle) = |\langle\psi|1\rangle|^2 = |\beta|^2$

References

[1] Saggio, V., Asenbeck, B.E., Hamann, A., Strömberg, T., Schiansky, P., Dunjko, V., Friis, N., Harris, N.C., Hochberg, M., Englund, D., Wölk, S., Briegel, H.J., Walther, P.: Experimental quantum speed-up in reinforcement learning agents. Nature **591**(7849), 229–233 (2021) https://doi.org/10.1038/s41586-021-03242-7

[2] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.: Training language models to follow instructions with human feedback. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022). https://openreview.net/forum?id=TG8KACxEON

[3] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA (2018)

[4] Bellman, R.: Dynamic Programming. Dover Publications, ??? (1957)

[5] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. STOC '96, pp. 212–219. Association for Computing Machinery, New York, NY, USA (1996). https://doi.org/10.1145/237814.237866 . https://doi.org/10.1145/237814.237866

[6] Dirac, P.A.M.: The Principles of Quantum Mechanics. Clarendon Press, ??? (1930)

[7] Qiskit: Summary of Quantum Operators. https://qiskit.org/documentation/tutorials/circuits/3_summary_of_quantum_operations.html Accessed 2023-5-31

[8] Microsoft: Description of Grover Algorithm. https://learn.microsoft.com/en-us/azure/quantum/concepts-grovers Accessed 2023-5-31

[9] Caldarella, S.: QuantumEnhancedRL. https://github.com/SimoneCaldarella/QuantumEnhancedRL Accessed 2023-5-31