Reinforcement learning in the quantum domain

Sofiène Jerbi

Institute for Theoretical Physics, University of Innsbruck, Technikerstr. 21a, A-6020 Innsbruck

Supervisors: Hans J. Briegel, Vedran Dunjko

universität innsbruck

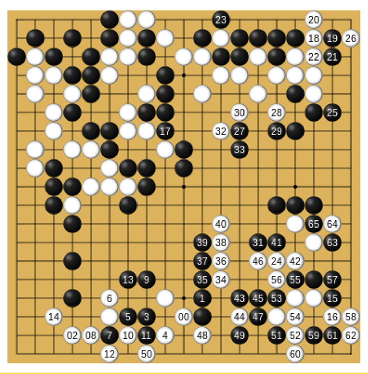
FWF Der Wissenschaftsfonds.

ÖAW ÖSTERREICHISCHE AKADEMIE DER WISSENSCHAFTEN

Atoms, Light, and Molecules Innsbruck Physics Research Center

Reinforcement learning 101

Reinforcement learning (RL) considers the very general framework of an agent taking actions in an environment to accumulate rewards, which can be seen as a sequential decision-making problem. These rewards act as feedback from the environment and quantify the performance of the agent with regard to the task/environment.



Examples of RL task environments:

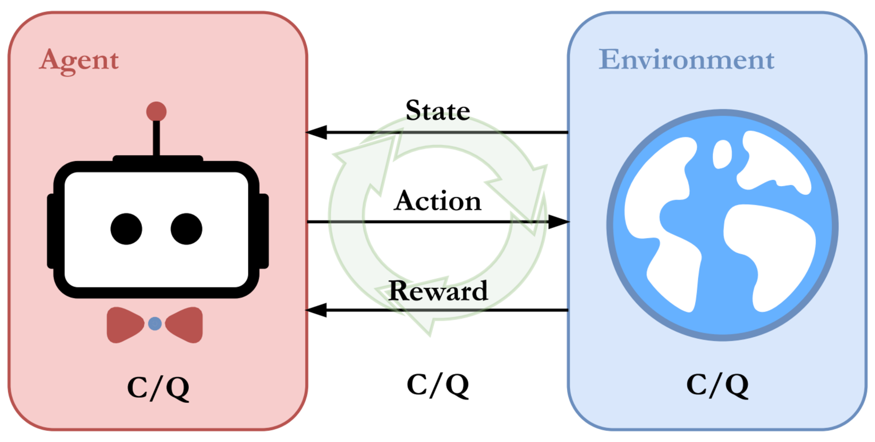
- A board game of Go (or chess), where RL agents have already defeated world champions (AlphaGo)
- An optical table for RL agents to design quantum experiments

Reinforcement learning: quantization scenarios

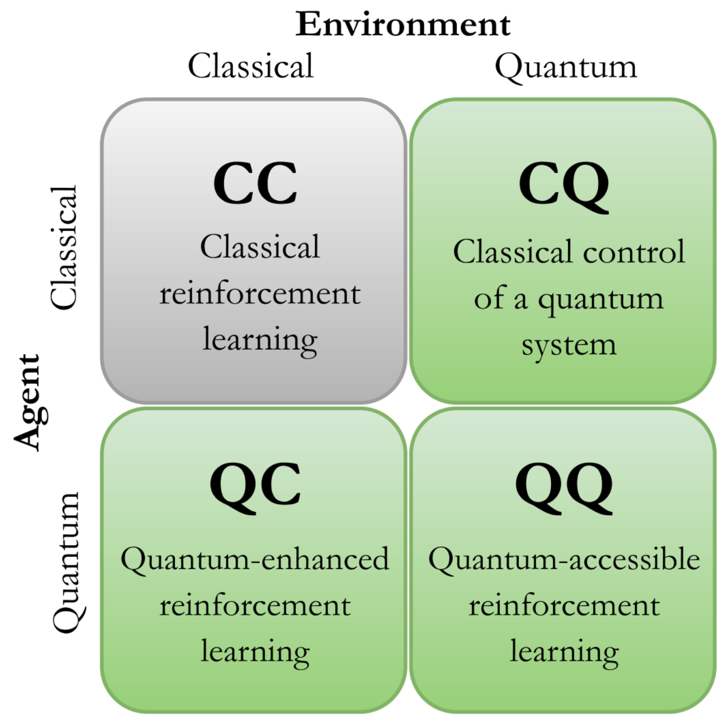
While many quantum algorithms for (un)supervised learning have been proposed in the last decade, relatively few have been dealing with quantum reinforcement learning (QRL).

Paradigm	Supervised learning	Unsupervised learning	Reinforcement learning
Learn	P(labels | data)	Structure in P(data)	Optimal π^*, with π = P(actions | states)
Samples from	P(data, labels)	P(data)	P(history | π, env.)

Table 1: **A comparison of the three paradigms of machine learning.** The history of a reinforcement learning agent is a sequence a states, actions and rewards it has experienced.



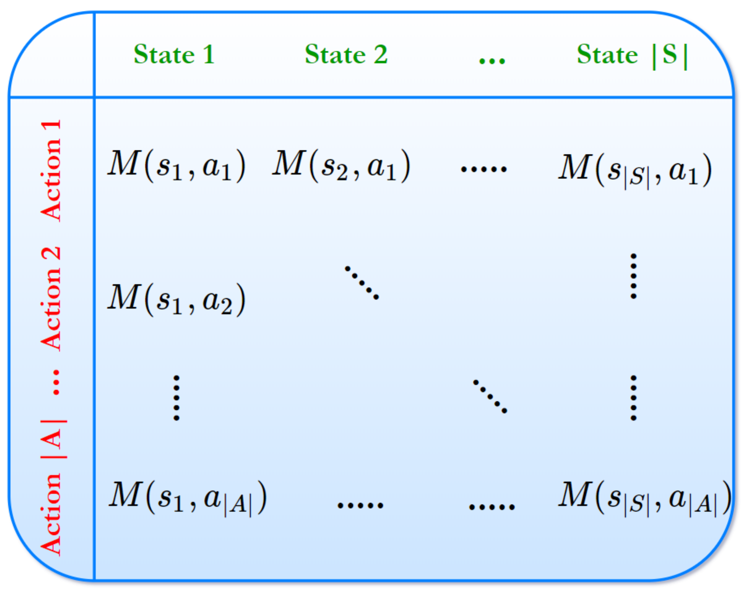
The reinforcement learning framework. An agent is situated in an environment with which it maintains a cyclic interaction. During this interaction, they exchange states, actions and rewards.



Depending on which of the agent, the environment (and their interaction) should be granted quantum mechanical abilities, three different settings can be casted as QRL. My PhD project mainly focuses on the QC and QQ scenarios.

The curse of dimensionality and its solutions

Traditional methods for RL rely on learning a merit function defined on the entire state-action space, that estimates the expected future rewards of an agent when performing a particular action in a given state. When learned on the entire state-action space, an optimal behavior (or policy) for the agent can be derived by following the actions with the largest merit value in any encountered state.



$M(s,a)$ can be $Q(s,a)$ or $h(s,a)$

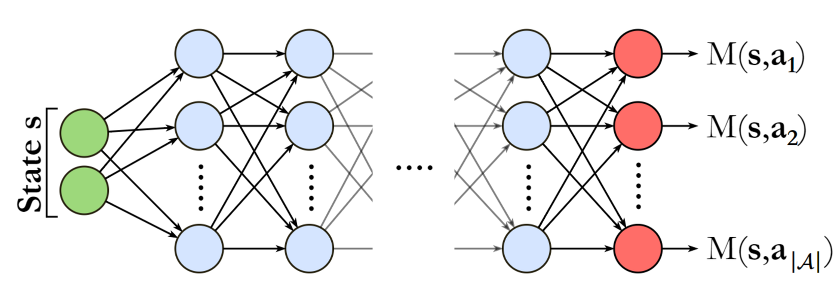
- Projective Simulation (PS) [1]:
  $h^{(t+1)}(s,a) \leftarrow h^{(t)}(s,a) - \gamma_{ps}(h^{(t)}(s,a) - 1) + \tilde{r}$
- Q-learning [2]:
  $Q^{(t+1)}(s,a) \leftarrow (1-\alpha)Q^{(t)}(s,a) + \alpha\left[r + \gamma \max_a Q^{(t)}(s',a)\right]$
- Policy: $\pi(a|s) = \dfrac{e^{M(s,a)}}{\sum_{a'} e^{M(s,a')}}$
- Drawbacks:
  - Lack of generalization: values updated one at a time
  - Slow learning: $|S| \times |A|$ values

The most basic learning algorithms for RL (e.g., Q-learning and Projective Simulation below) rely on a table of values stored in memory to approximate the merit function. These stored values are updated using an update rule proper to the learning method, and according to the rewards collected by interacted with the environment when following a given policy. This policy is itself derived from the stored table through normalization of its columns.

In real-world environments, e.g., Go play or quantum experiments described above, the size of the state and action spaces is gigantic. But basic tabular methods update one value in the table at a time, leading to learning times at least as large as these spaces.

A common solution to this problem relies on function approximation models, such as artificial neural networks. These do not represent each value in the table separately but define a family of functions parametrized by a set of weights. Training then consist in finding the most suited function (i.e., weights) in this family.

Deep Q-networks [5]:



Pros:
- Efficient sampling from policy

Cons:
- Poor representational power in large action spaces (due to the size of the output layer)

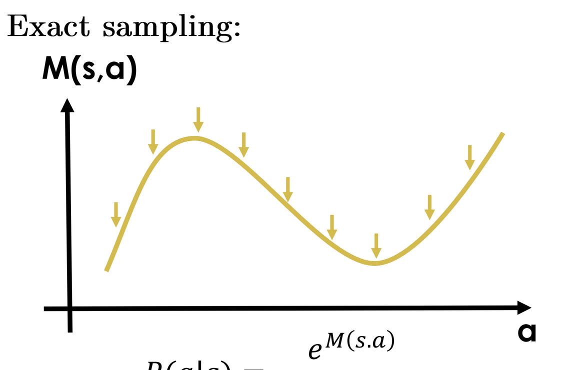
Quantum enhancements for reinforcement learning

Quantum enhancements can be of two kinds:

- Quantum algorithms to speed-up sampling from the agent policy

Using quantum subroutines such as amplitude amplification, one can gain quadratic speed-ups over existing classical methods for sampling [4-6]

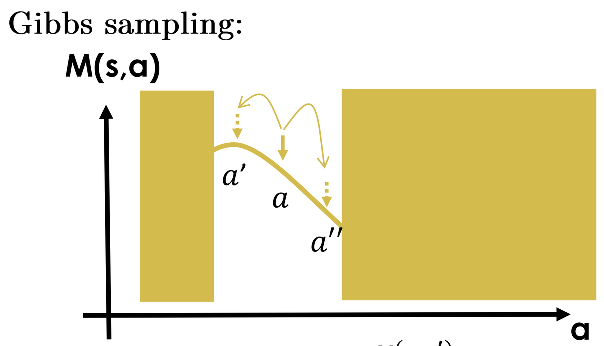
Exact sampling:



$P(a|s) = \frac{e^{M(s,a)}}{\sum_{a'} e^{M(s,a')}}$

Complexity: $\tilde{O}\left(\sqrt{|A|}\right)$ v.s. $\tilde{O}(|A|)$

Gibbs sampling:

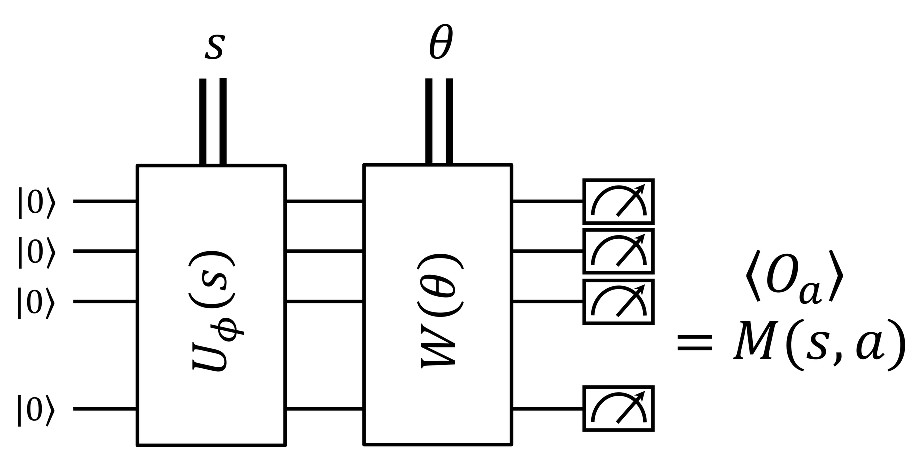


$p(a'|a) = \frac{e^{M(s,a')}}{e^{M(s,a)} + e^{M(s,a')} + e^{M(s,a'')}}$

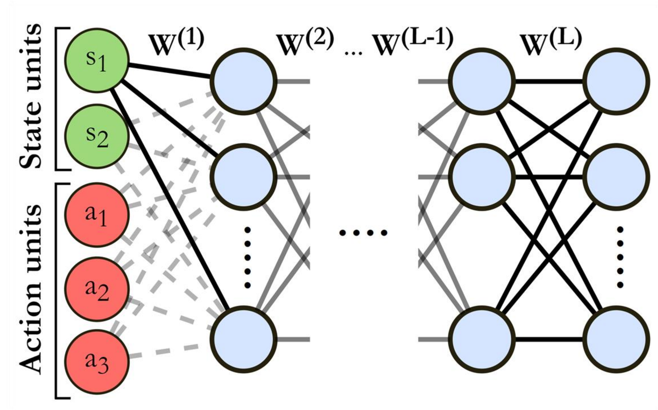
Complexity: $\tilde{O}\left(\frac{1}{\sqrt{\delta}}\right)$ v.s. $\tilde{O}\left(\frac{1}{\delta}\right)$

- Quantum generalizations of function approximation models to gain a learning advantage

Interesting candidates are:



Variational quantum circuits



Quantum Boltzmann machines

References

[1] Briegel, Hans J., and Gemma De las Cuevas. **Projective simulation for artificial intelligence.** *Scientific reports* 2 (2012): 400.

[2] Watkins, Christopher JCH, and Peter Dayan. **Q-learning.** *Machine learning* 8.3-4 (1992): 279-292.

[3] Mnih, Volodymyr, et al. **Human-level control through deep reinforcement learning.** *Nature* 518.7540 (2015): 529.

[4] Van Apeldoorn, Joran, et al. **Quantum SDP-solvers: Better upper and lower bounds.** *Quantum* 4 (2020): 230.

[5] Wocjan, Pawel, and Anura Abeyesinghe. **Speedup via quantum sampling.** *Physical Review A* 78.4 (2008): 042336.

[6] Harrow, Aram W., and Annie Y. Wei. **Adaptive Quantum Simulated Annealing for Bayesian Inference and Estimating Partition Functions.** *arXiv:1907.09965* (2019).