Quantum Reinforcement Learning and Projective Simulation

Sofiène Jerbi,
sofiene.jerbi@telecom-paristech.fr

Paris, France

Quantum Algorithms

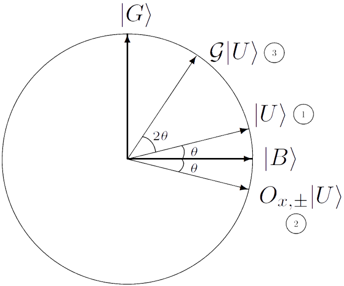


Figure: Grover's search algorithm

1. Start at uniform distribution
2. Reflect over $|B\rangle$
3. Reflect over $|U\rangle$

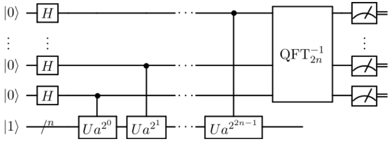


Figure: Shor's factoring algorithm

Based on the Fast Quantum Fourier Transform (QFT)

Outline

Outline

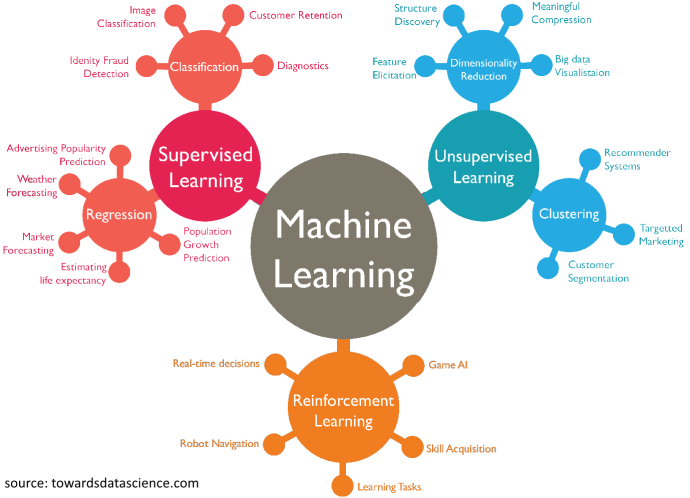
Classical & Quantum Machine Learning

Projective Simulation & Quantum Walks

Contributions

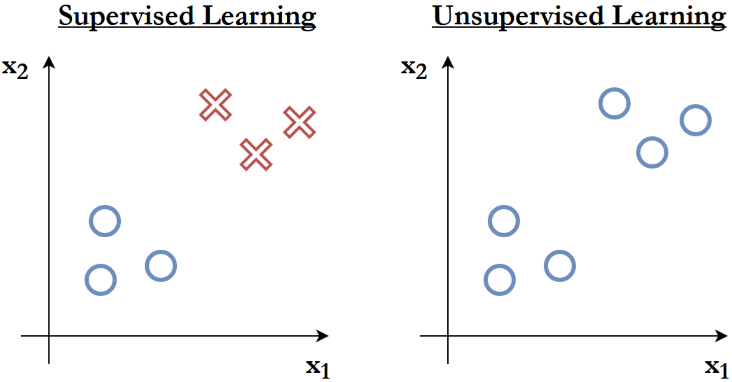
Conclusions & Prospects

Machine Learning

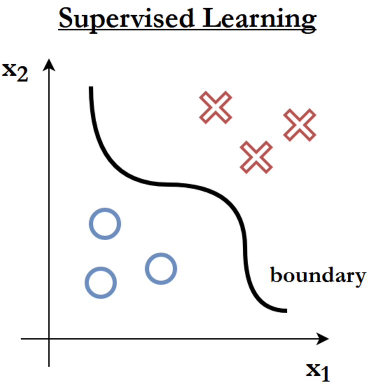


source: towardsdatascience.com

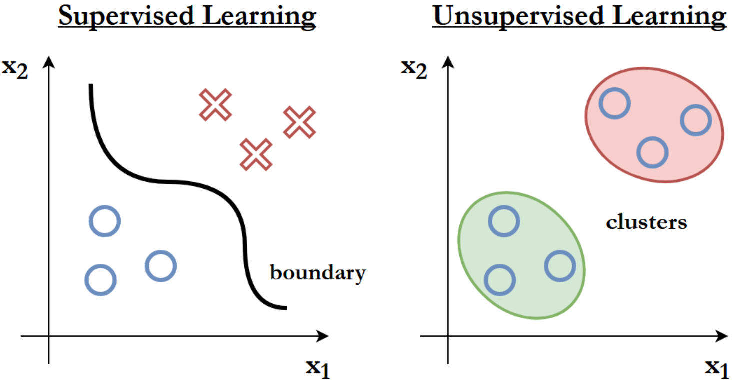
Supervised vs. Unsupervised Learning



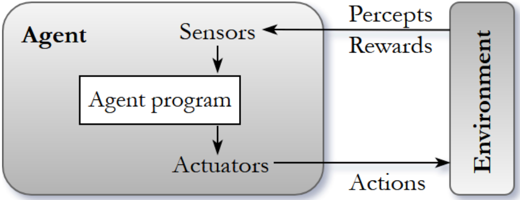
Supervised vs. Unsupervised Learning



Supervised vs. Unsupervised Learning



Reinforcement Learning



Existing Quantum Machine Learning

Box 1 Table | Speedup techniques for given quantum machine learning subroutines

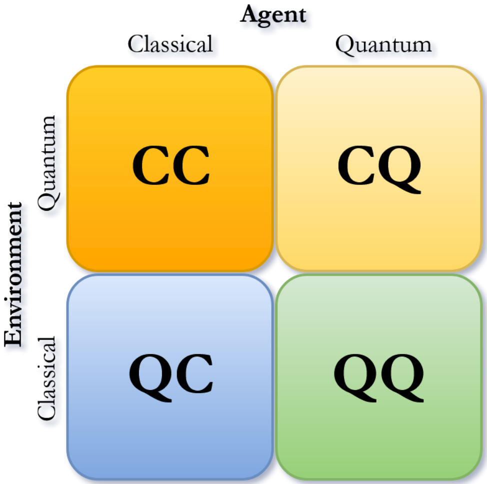
Method	Speedup	Amplitude amplification	HHL	Adiabatic	qRAM
Bayesian inference[106,107]	$O(\sqrt{N})$	Yes	Yes	No	No
Online perceptron[108]	$O(\sqrt{N})$	Yes	No	No	Optional
Least-squares fitting[9]	$O(\log N)$*	Yes	Yes	No	Yes
Classical Boltzmann machine[20]	$O(\sqrt{N})$	Yes/No	Optional/No	No/Yes	Optional
Quantum Boltzmann machine[22,61]	$O(\log N)$*	Optional/No	No	No/Yes	No
Quantum PCA[11]	$O(\log N)$*	No	Yes	No	Optional
Quantum support vector machine[13]	$O(\log N)$*	No	Yes	No	Yes
Quantum reinforcement learning[30]	$O(\sqrt{N})$	Yes	No	No	No

*There exist important caveats that can limit the applicability of the method[51].

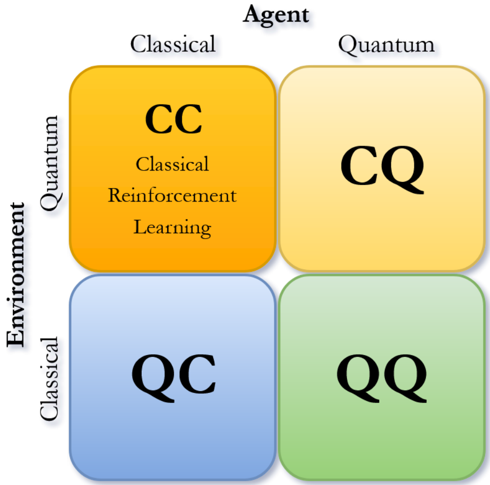
source: doi:10.1038/nature23474

TELECOM ParisTech

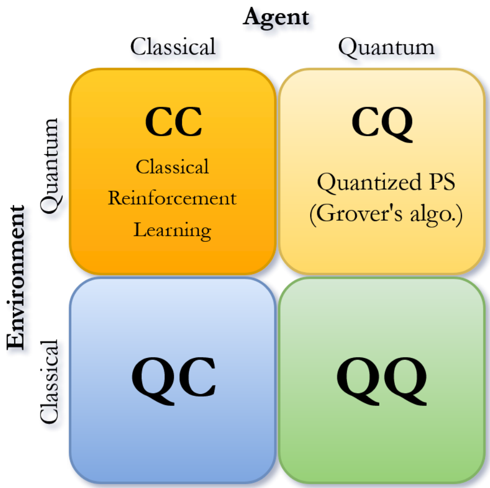
Where should be the quantum?



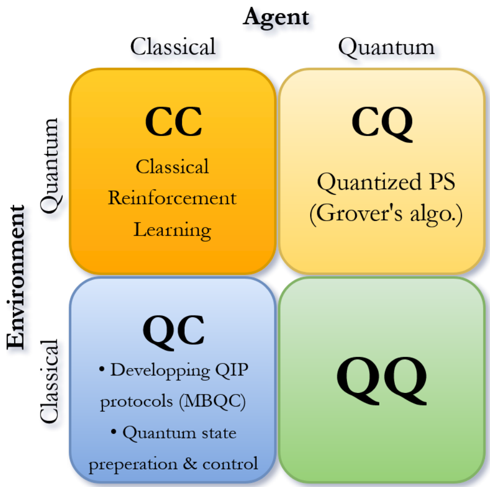
Where should be the quantum?



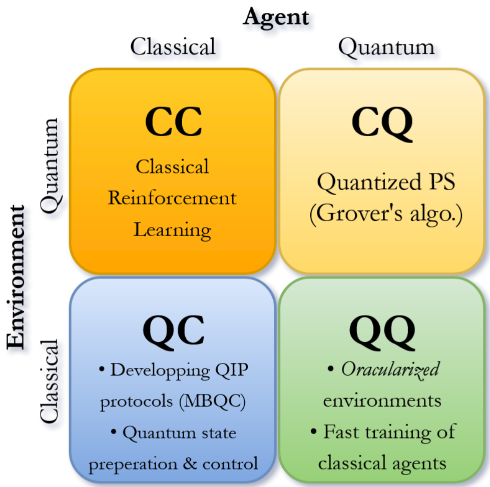
Where should be the quantum?



Where should be the quantum?



Where should be the quantum?



Outline

Projective Simulation

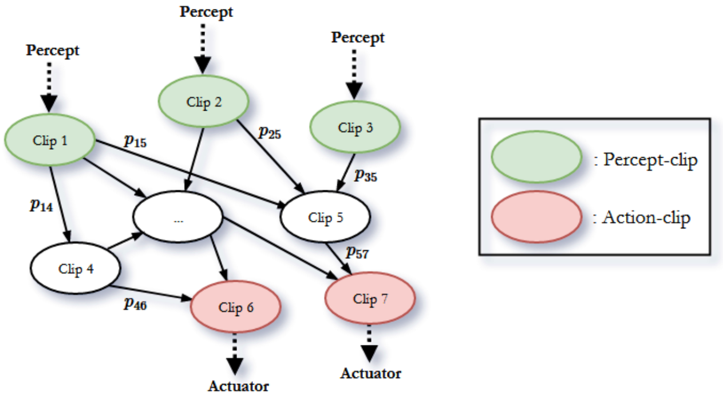
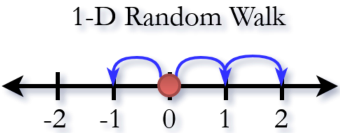
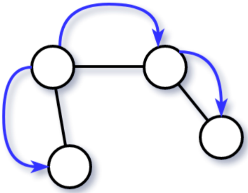


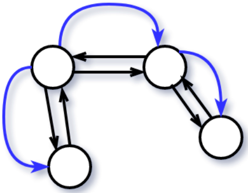
Figure: The Episodic Compositional Memory (ECM)

Random Walks





Random Walk on an Undirected Graph

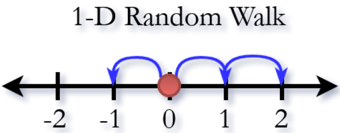


Random Walk on a Stochastic Graph

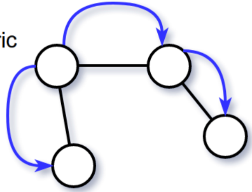
Random Walks

Transition matrix

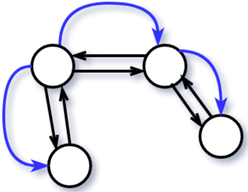
$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{n1} & \cdots & \cdots & p_{nn} \end{pmatrix}$$



P symmetric



Random Walk on an Undirected Graph



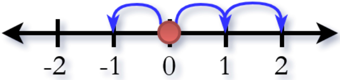
Random Walk on a Stochastic Graph

Random Walks

Transition matrix

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{n1} & \cdots & \cdots & p_{nn} \end{pmatrix}$$

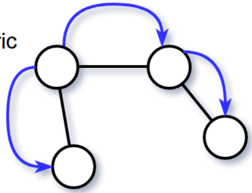
1-D Random Walk



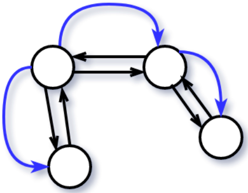
$$v = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}, \sum_i p_i = 1$$

k steps from v_0: $P^k v_0$

P symmetric



Random Walk on an Undirected Graph

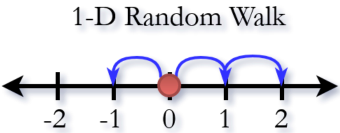


Random Walk on a Stochastic Graph

Random Walks

Transition matrix

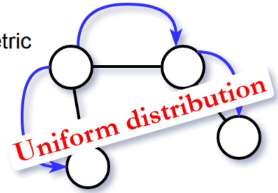
$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{n1} & \cdots & \cdots & p_{nn} \end{pmatrix}$$



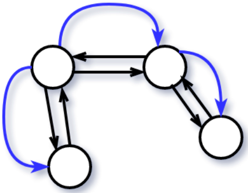
$$v = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}, \sum_i p_i = 1$$

k steps from v_0: $P^k v_0$

P symmetric



Random Walk on an Undirected Graph



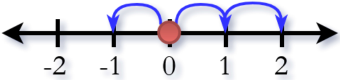
Random Walk on a Stochastic Graph

Random Walks

Transition matrix

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{n1} & \cdots & \cdots & p_{nn} \end{pmatrix}$$

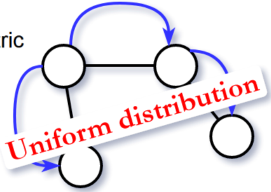
1-D Random Walk



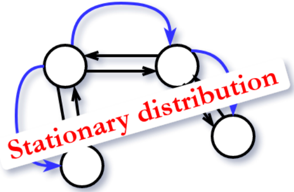
$$v = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}, \sum_i p_i = 1$$

k steps from v_0: $P^k v_0$

P symmetric



Random Walk on an Undirected Graph



Random Walk on a Stochastic Graph

Random Walk algorithm

The parameters

- Cumulated probability of marked elements ϵ
- Spectral gap $\delta = 1 - max_{i \neq 0}|\lambda_i|$

The Algorithm

Start with some arbitrary distribution v.
Repeat until hitting a marked vertex (roughly $1/\epsilon$ times):

- *Run a random walk for roughly $1/\delta$ steps (updating v).*
- *Pick a random vertex y according to the distribution v.*
- *Check if y is marked.*

Complexity (time and number of queries)

$$\mathbf{S} + \frac{1}{\epsilon}(\mathbf{C} + \frac{1}{\delta}\mathbf{U}) \tag{1}$$

Quantum Walks

Szegedy Walk Operator

A unitary doesn't have enough degrees of freedom to hold every entry in the transition matrix P. We then need to define a unitary $W(P)$ that acts no longer on vertices but on edges $|x\rangle_{\mathrm{I}}\ |y\rangle_{\mathrm{II}}$.

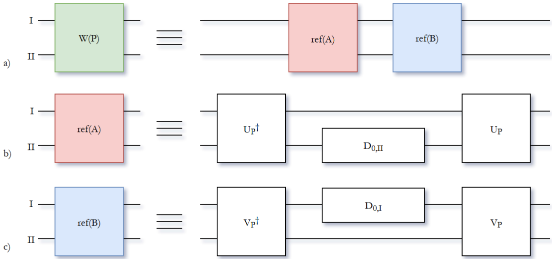


Figure: Szegedy Walk Operator Construction

Quantum Walks

The Algorithm

Start by setting up the uniform distribution over edges $|U\rangle$.
Repeat roughly $1/\sqrt{\epsilon}$ *times:*

- *Reflect over* $|B\rangle$ *(edges with unmarked endpoints).*
- *Reflect over* $|U\rangle$.

Measure the register I *and check if the resulting vertex* x *is marked.*

Complexity (time and number of queries)

$$\mathbf{S} + \frac{1}{\sqrt{\epsilon}}(\mathbf{C} + \frac{1}{\sqrt{\delta}}\mathbf{U}) \tag{2}$$

Quantum Walk Circuit

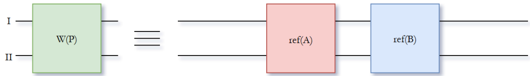


Figure: Szegedy Walk Operator

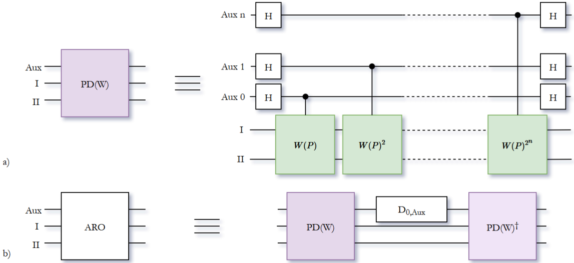


Figure: Kitaev's Phase Detection Algorithm & Approximate Reflection Operator

Outline

Classical & Quantum Machine Learning

Projective Simulation & Quantum Walks

Contributions

Conclusions & Prospects

Main papers

Quantum speedup for active learning agents

Giuseppe Davide Paparo[1], Vedran Dunjko[2,3,4], Adi Makmal[2,3], Miguel Angel Martin-Delgado[1], and Hans J. Briegel[2,3]

[1] Departamento de Física Teórica I, Universidad Complutense, 28040 Madrid, Spain
[2] Institut für Theoretische Physik, Universität Innsbruck, Technikerstraße 25, A-6020 Innsbruck, Austria
Institut für Quantenoptik und Quanteninformation der Österreichischen Akademie der Wissenschaften, A-6020 Innsbruck, Austria
Ruđer Bošković Institute, Bijenička cesta 54, 10002 Zagreb, Croatia

14 Jul 2014

Quantum-enhanced deliberation of learning agents using trapped ions

Vedran Dunjko,[1,2,3,*] Nicolai Friis,[1,†] and Hans J. Briegel[1,2,‡]

[1] Institute for Quantum Optics and Quantum Information, Austrian Academy of Sciences, Technikerstraße 21a, A-6020 Innsbruck, Austria
[2] Institute for Theoretical Physics, University of Innsbruck, Technikerstraße 25, A-6020 Innsbruck, Austria
[3] Laboratory of Evolutionary Genetics, Division of Molecular Biology, Ruđer Bošković Institute, Bijenička cesta 54, HR-10000 Zagreb, Croatia
(Dated: February 3, 2015)

[quant-ph] 31 Jan 2015

A scheme that successfully employs quantum mechanics in the design of autonomous learning agents has recently been reported in the context of the projective simulation (PS) model for artificial intelligence. In that approach, the key feature of a PS agent, a specific type of memory which is explored via random walks, was shown to be amenable to quantization, allowing for a speed-up. In this work we propose an implementation of such classical and quantum agents in systems of trapped ions. We employ a generic construction by which the classical agents are 'upgraded' to their quantum counterparts by a nested process of adding coherent control, and we outline how this construction can be realized in ion traps. Our results provide a flexible modular architecture for the design of PS agents. Furthermore, we present numerical simulations of simple PS agents which analyze the robustness of our proposal under certain noise models.

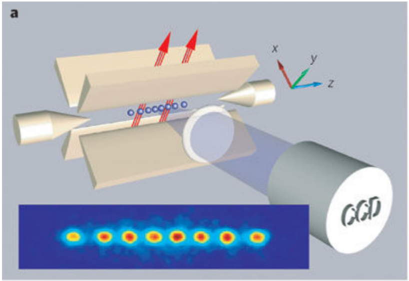
PACS numbers: 07.05.Mh, 03.67.Lx, 37.10.Ty, 05.40.Fb

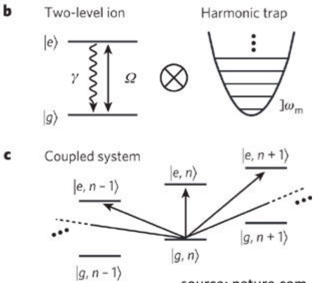
I. INTRODUCTION

In the past decades, quantum physics has been employed to enhance communication and information processing with significant success, laying the foundation for the now well established fields of quantum computation and quantum information [1–5]. In contrast, the potential of merging the related, but distinct, field of artificial

In this work we present a proposal for the experimental implementation of both classical and quantum PS agents in systems of trapped ions. While the classical variants of PS agents can easily be realized in physical systems without requiring quantum control, we show here how certain implementations of classical agents in ion traps can be used to construct quantum PS agents. This is achieved in a generic way through a nested process of

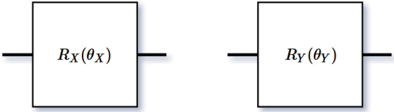
Trapped Ions Quantum Computer

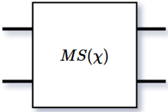




source: nature.com

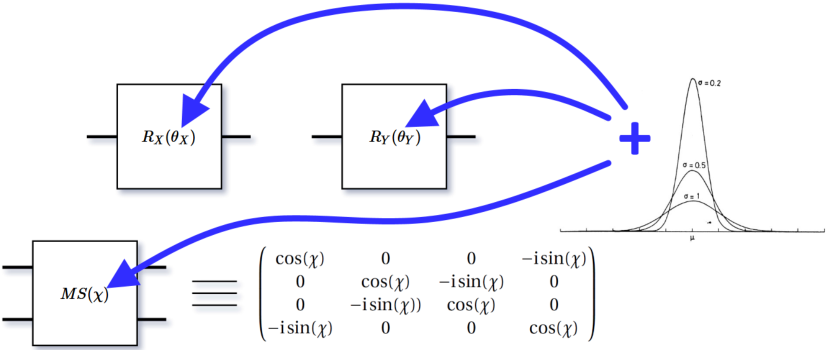
Universal Set of Gates for Trapped-ions QC





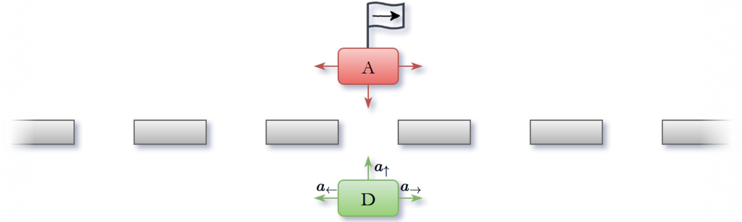
$$\equiv \begin{pmatrix} \cos(\chi) & 0 & 0 & -i\sin(\chi) \\ 0 & \cos(\chi) & -i\sin(\chi) & 0 \\ 0 & -i\sin(\chi)) & \cos(\chi) & 0 \\ -i\sin(\chi) & 0 & 0 & \cos(\chi) \end{pmatrix}$$

Universal Set of Gates for Trapped-ions QC



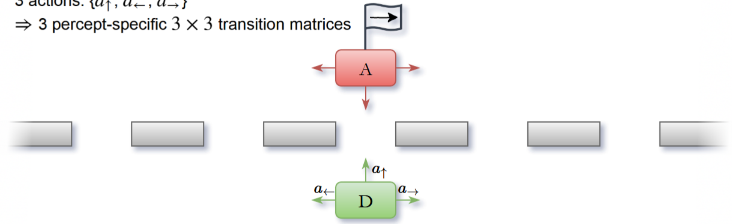
$$MS(\chi) \equiv \begin{pmatrix} \cos(\chi) & 0 & 0 & -i\sin(\chi) \\ 0 & \cos(\chi) & -i\sin(\chi) & 0 \\ 0 & -i\sin(\chi)) & \cos(\chi) & 0 \\ -i\sin(\chi) & 0 & 0 & \cos(\chi) \end{pmatrix}$$

Test-bed Application: Invasion Game

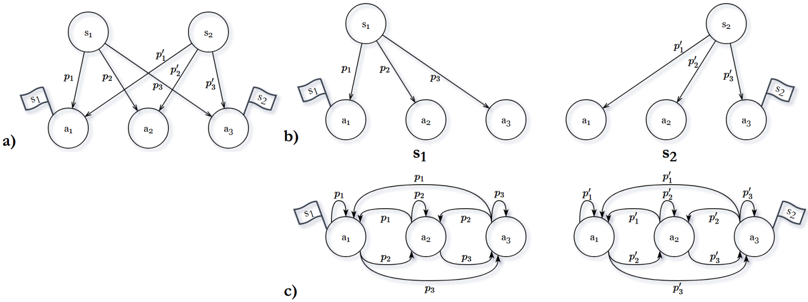


Test-bed Application: Invasion Game

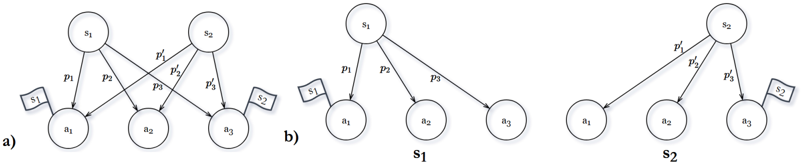
3 percepts: $\{\uparrow, \leftarrow, \rightarrow\}$
3 actions: $\{a_\uparrow, a_\leftarrow, a_\rightarrow\}$
$\Rightarrow$ 3 percept-specific 3×3 transition matrices



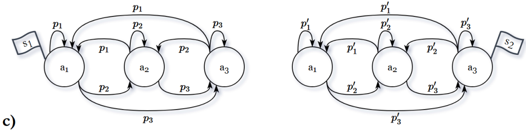
Constructing our Reflecting Projective Simulation (RPS) model



Constructing our Reflecting Projective Simulation (RPS) model



$$P_{s_1} = \begin{pmatrix} p_1 & p_1 & p_1 \\ p_2 & p_2 & p_2 \\ p_3 & p_3 & p_3 \end{pmatrix}$$



Two sets of simulations

Rank-one RPS (ϵ dependency): Many simplifications

- All columns of P are identical
- P has only one eigenvalue $(+1) \Rightarrow \delta = 1$
- No need for controlled operations nor ancilla qubits
- Stationary distribution $\pi = (p_1, p_2, p_3)$ with $p_1 + p_2 = \epsilon$

Rank-two RPS (δ dependency): More complex

- Transition matrix transformation:

$$P = \begin{pmatrix} p_1 + \lambda & p_1 & p_1 \\ p_2 - \lambda & p_2 & p_2 \\ p_3 & p_3 & p_3 \end{pmatrix} \tag{3}$$

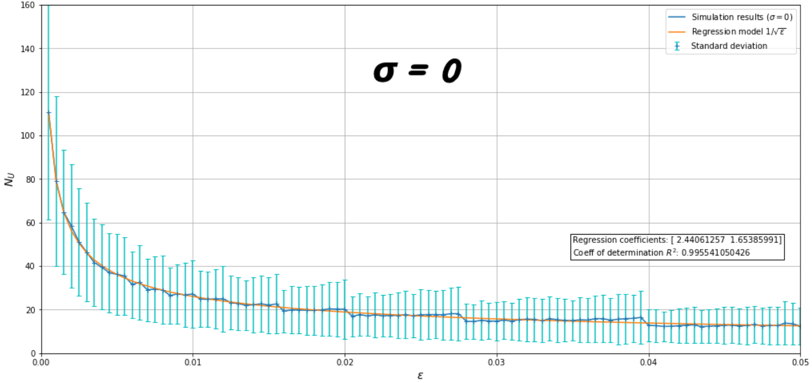
- $\delta = 1 - \lambda$, $\pi = (\frac{p_1}{1-\lambda}, p_2 - p_1 \times \frac{\lambda}{1-\lambda}, p_3)$, $\epsilon = \frac{p_1}{1-\lambda} = 0.25$

Running the simulations

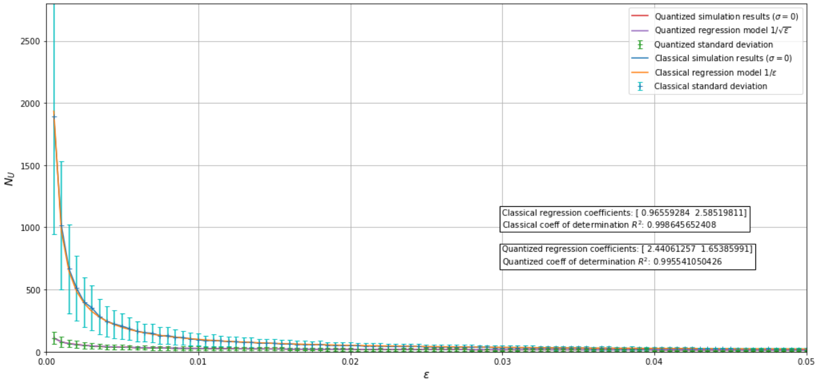
For each set of simulations:

- We set ϵ or δ arbitrarily close to 0 while cancelling the effect of the other.
- For each value of ϵ (δ), we run 1000 simulations.
- In each simulation we repeat the following procedure until a flagged action is hit:
  - Pick $m \in [\![0, 1/\sqrt{\epsilon}]\!]$ ($[\![0, 1/\sqrt{\delta}]\!]$) randomly and apply m steps of Random Walk
  - Measure register I and check if it is marked

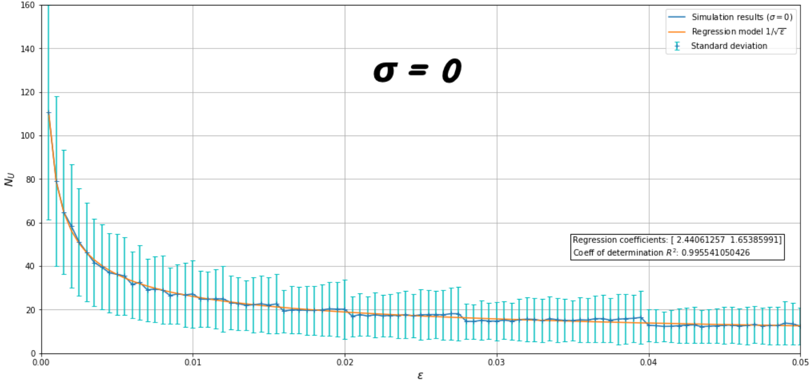
Showing the $1/\sqrt{\epsilon}$ dependency (Rank-one)



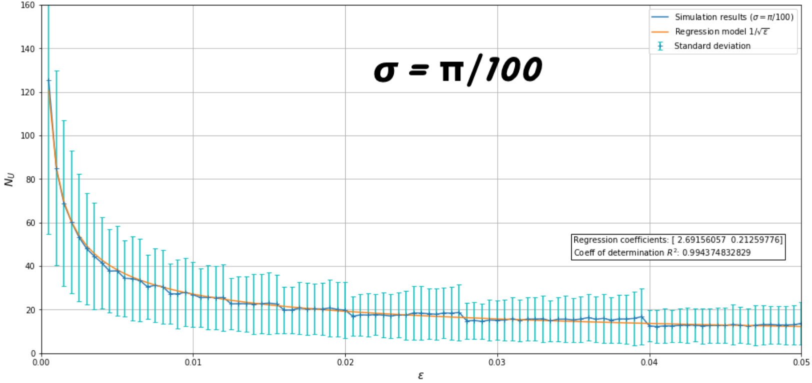
Showing the $1/\sqrt{\epsilon}$ dependency (Rank-one)



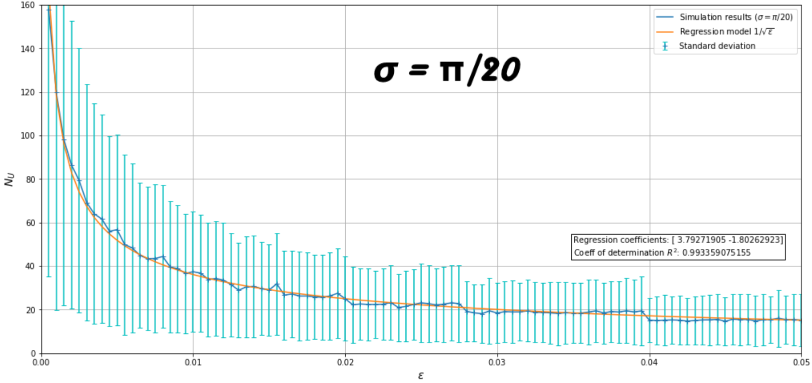
Showing the $1/\sqrt{\epsilon}$ dependency (Rank-one)



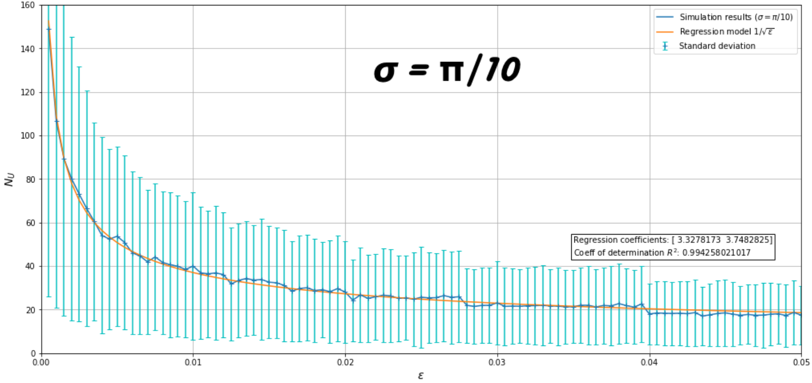
Showing the $1/\sqrt{\epsilon}$ dependency (Rank-one)



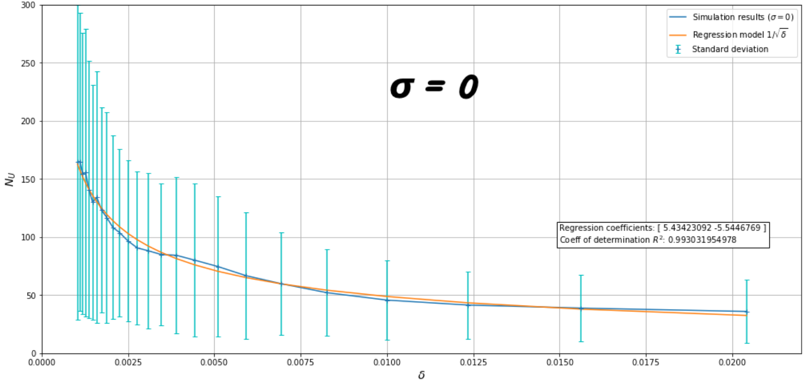
Showing the $1/\sqrt{\epsilon}$ dependency (Rank-one)



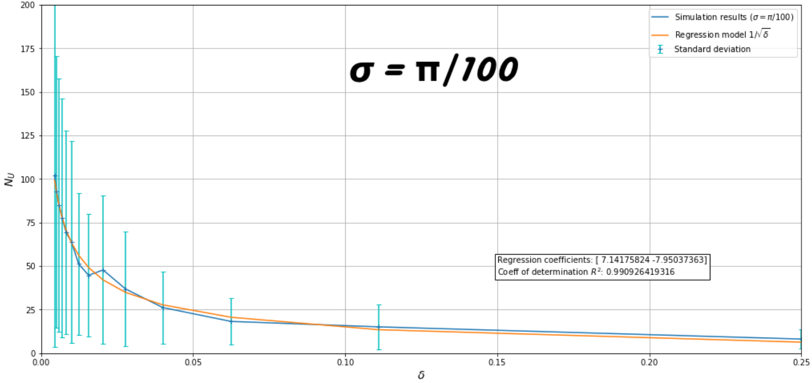
Showing the $1/\sqrt{\epsilon}$ dependency (Rank-one)



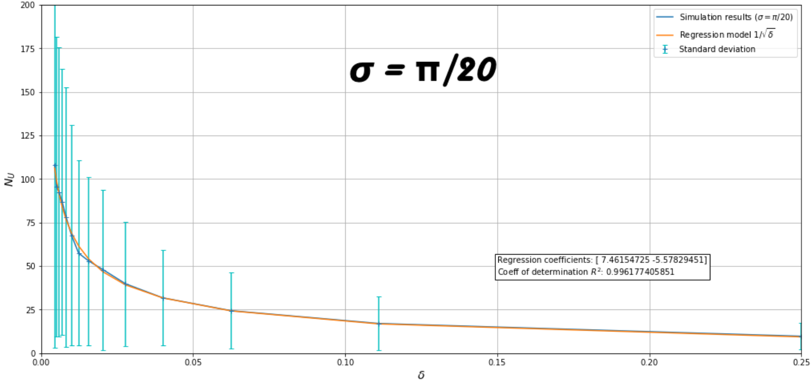
Showing the $1/\sqrt{\delta}$ dependency (Rank-two)



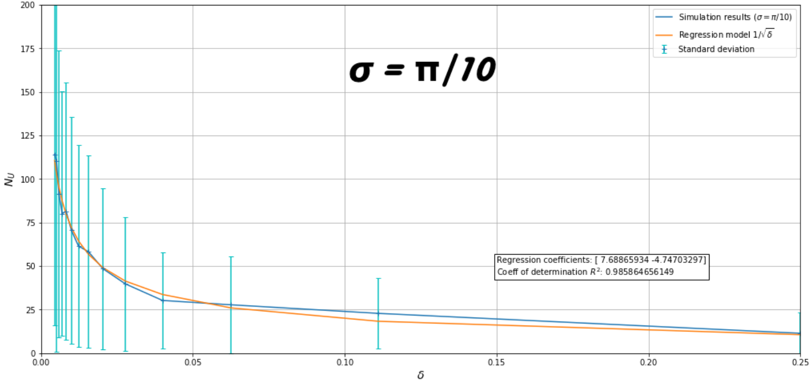
Showing the $1/\sqrt{\delta}$ dependency (Rank-two)



Showing the $1/\sqrt{\delta}$ dependency (Rank-two)



Showing the $1/\sqrt{\delta}$ dependency (Rank-two)



Outline

Classical & Quantum Machine Learning

Projective Simulation & Quantum Walks

Contributions

Conclusions & Prospects

Conclusions

Setting the context of Quantum Reinforcement Learning

- Where should be the quantum? Different scenarios and applications
- Theory of Projective Simulation: quantum-enhancement of Classical RL, offers a quadratic speed-up in deliberation time

Robustness of speed-ups

- Analyzed the effects of noise error in specific implementation with trapped ions
- Speed-ups are robust to small error

Caveats

Error model still very simple

- Multi-qubit controlled operations
- Global dephasing

Open problem in Projective Simulation

- Quantum Walks do not allow to get a path between percept-clip and selected action-clip, which is important in the update procedure of the ECM
- Not a problem in our "two-layered" case, but still an open problem for bigger graphs

Prospects

Curse of dimensionality

- Representing directly state-action associations in the memory of an RL agent does not allow to generalize to unprecedented situations.
- Especially a problem when the dimension of the the state space gets big.

Deep Reinforcement Learning

- Use Convolutional Neural Networks (CNNs) to model the state-action coinjoined probability distribution continuously allowing to interpolate learned associations to all possible input states.

Prospects

Embedding Boltzmann Machines in PS

- Boltzmann Machines are a different sort of Recurrent Neural Networks (RNNs) that model coinjoined distributions.
- Idea to show equivalence between two models to benefit from both quadratic speed-up and generalization property.

Want to read more about this project?

The full project report and the iPython code can be found on:

- https://github.com/sjerbi/Quantum-Reinforcement-Learning