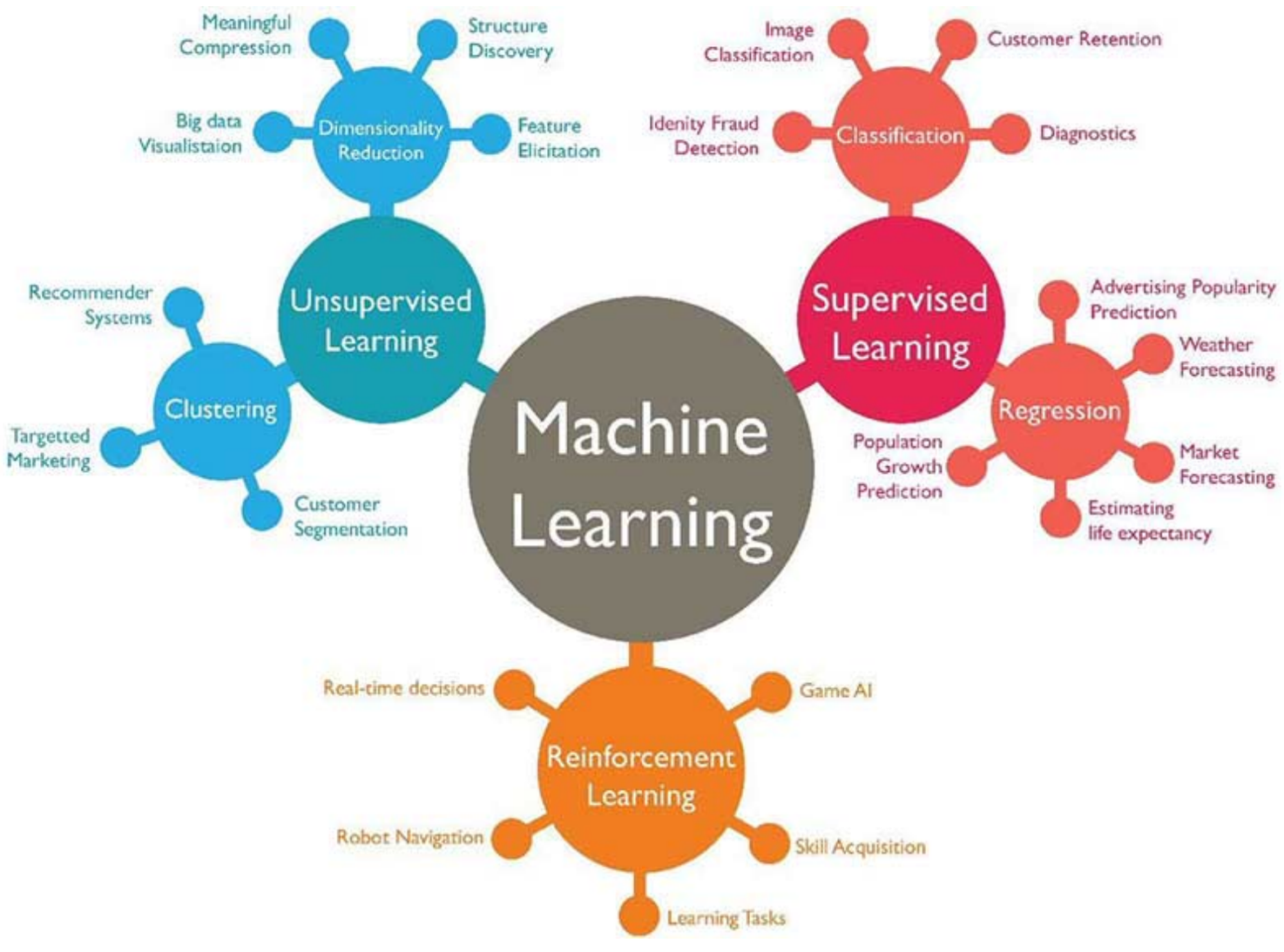
MPI-PKS

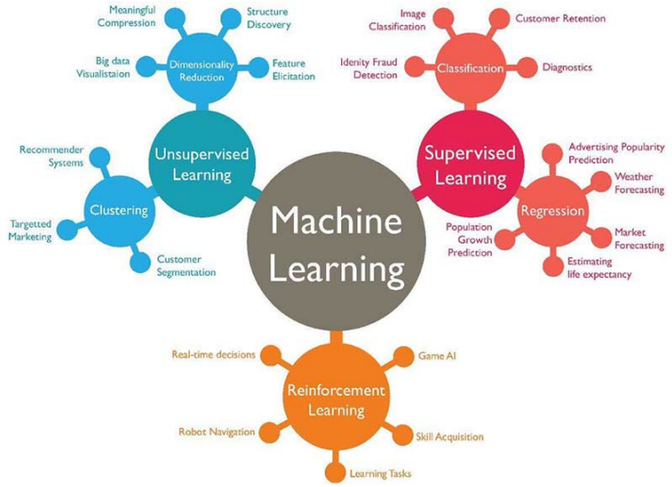
From Quantum Matter to Quantum Computers

Max Planck Institute
for the Physics of Complex Systems

INTRODUCTION TO REINFORCEMENT LEARNING

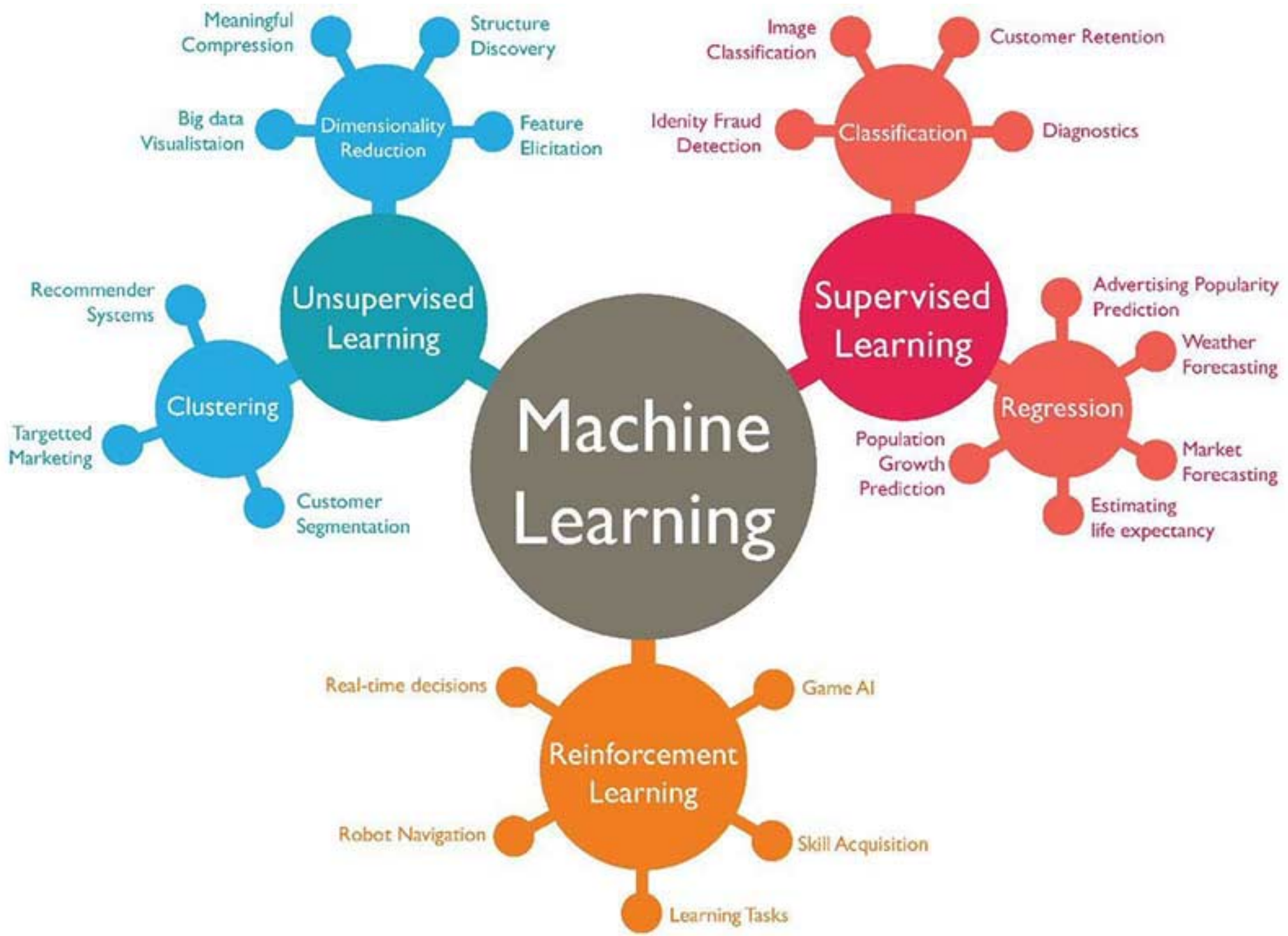


Overview

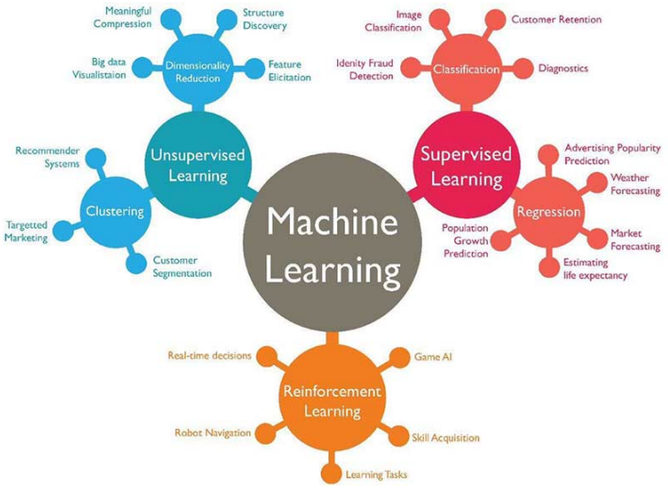


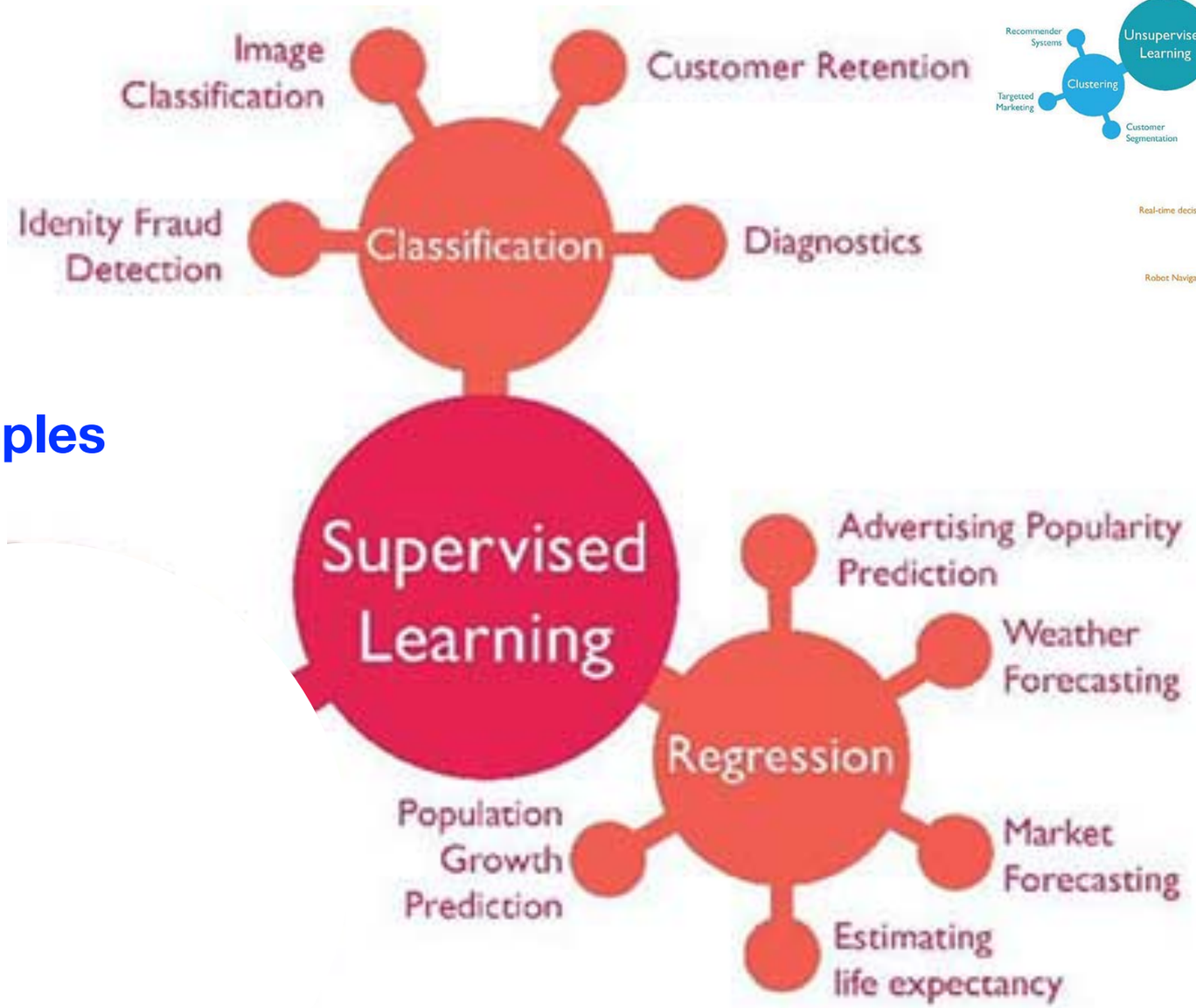
- What is Reinforcement Learning (RL)?
- RL framework in a nutshell
- RL algorithms
- Applications of RL in quantum physics

Machine Learning



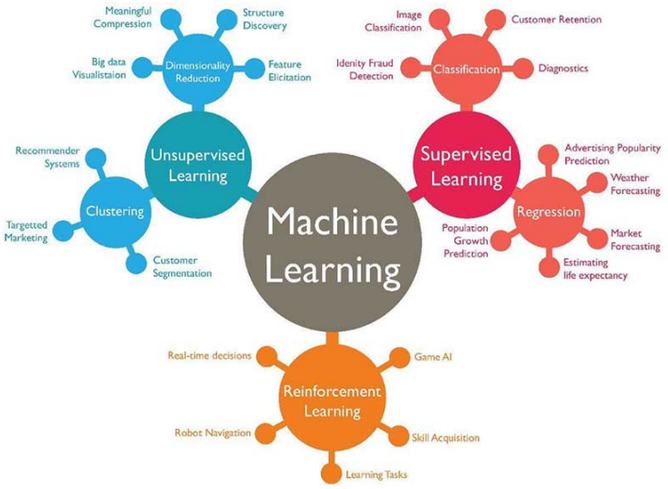
Supervised Learning

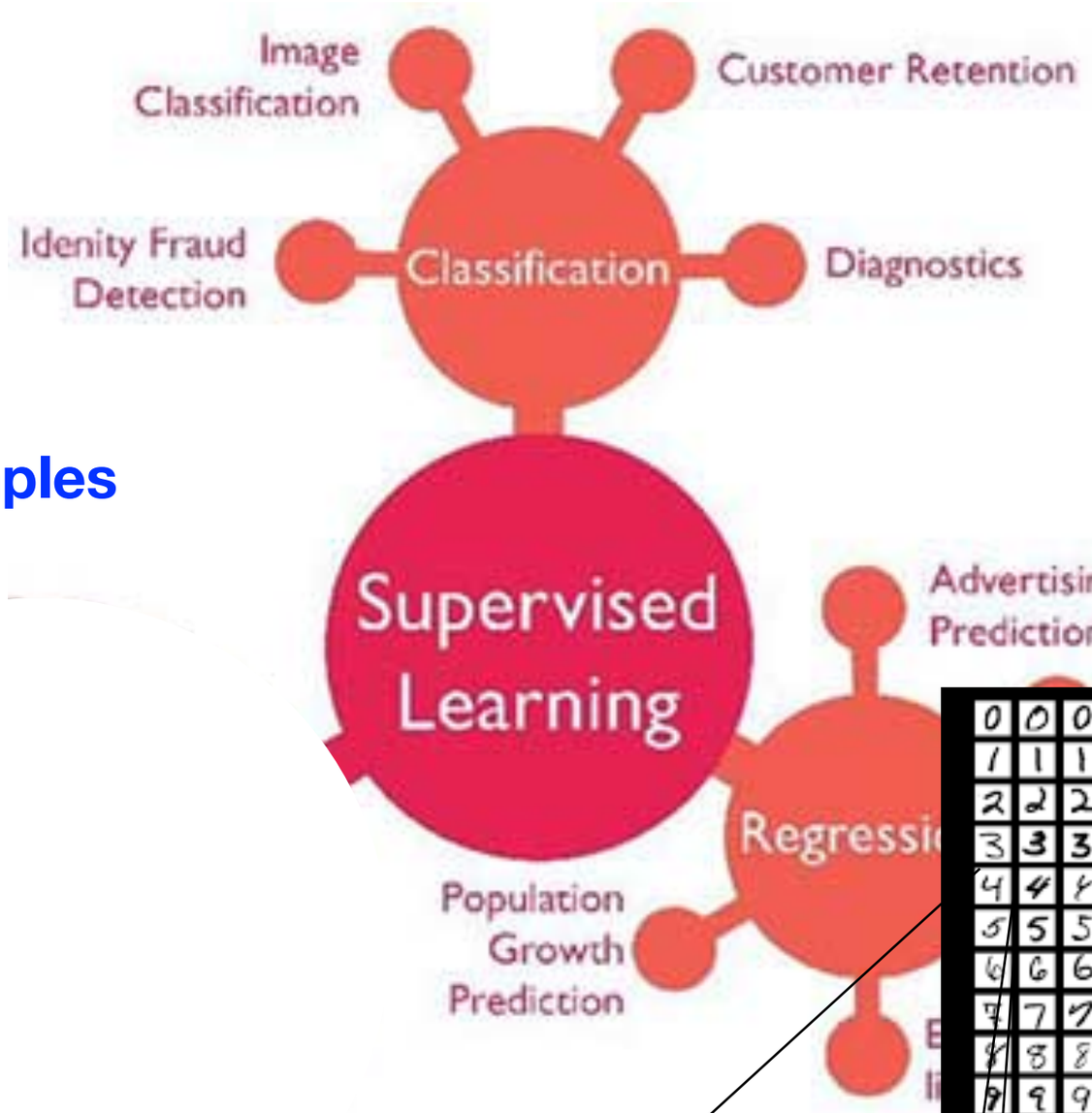




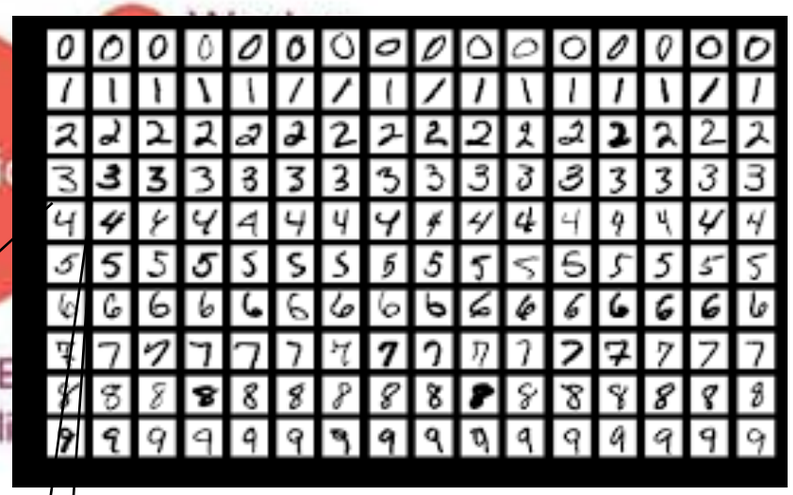
learning from examples
(labeled data)

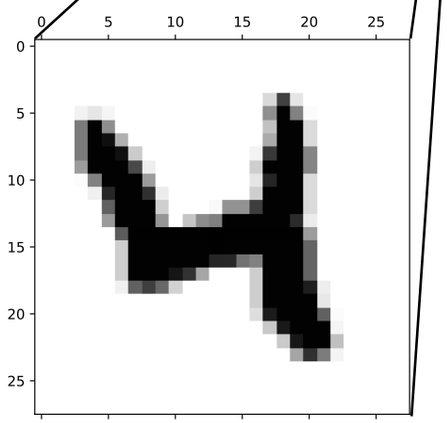
Supervised Learning





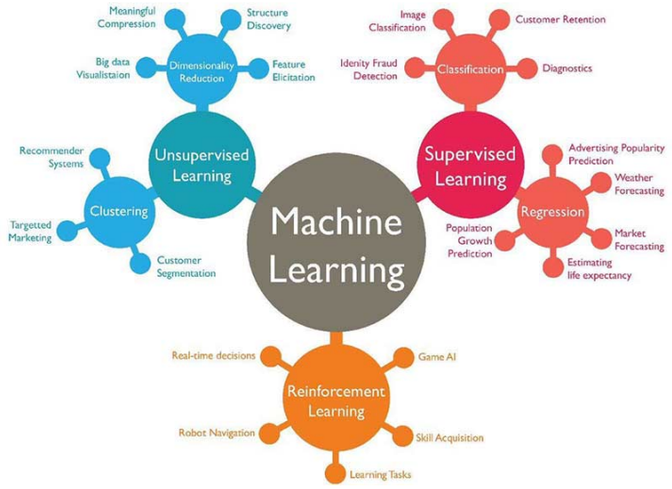
learning from examples
(labeled data)





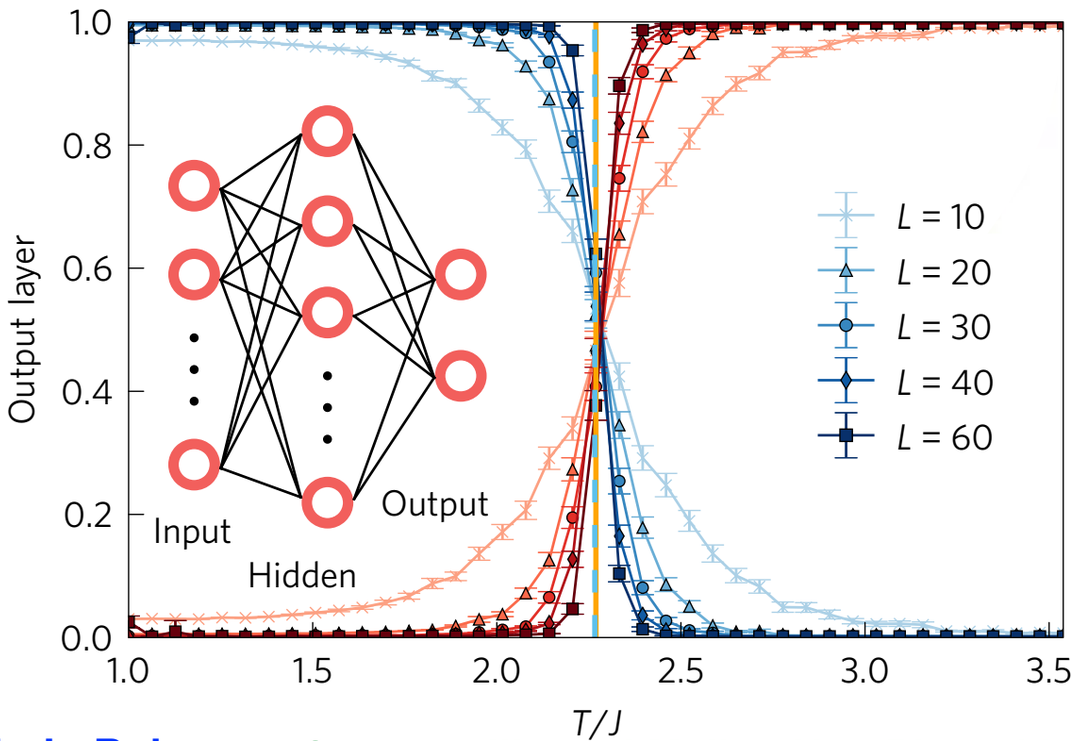
recognize hand-written digits

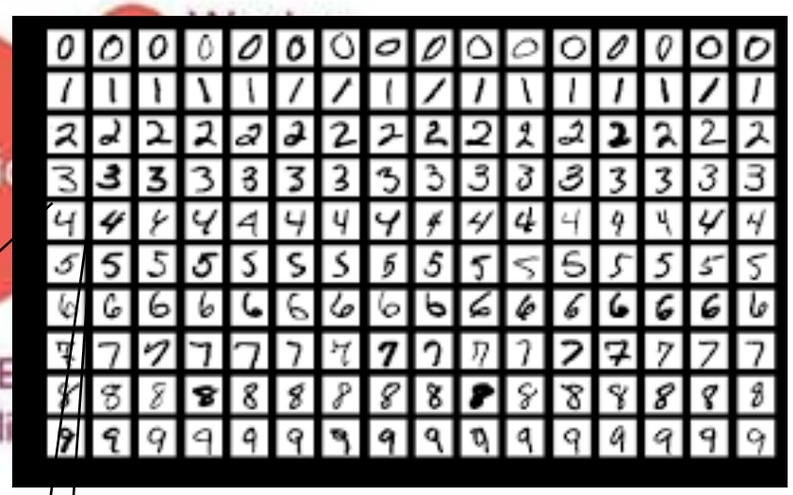
Supervised Learning

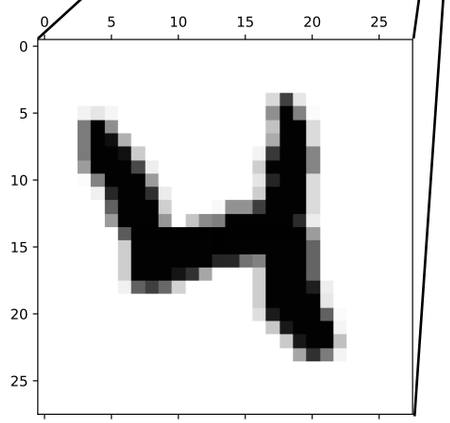


learning from examples (labeled data)

classify phases of matter, determine critical point from data

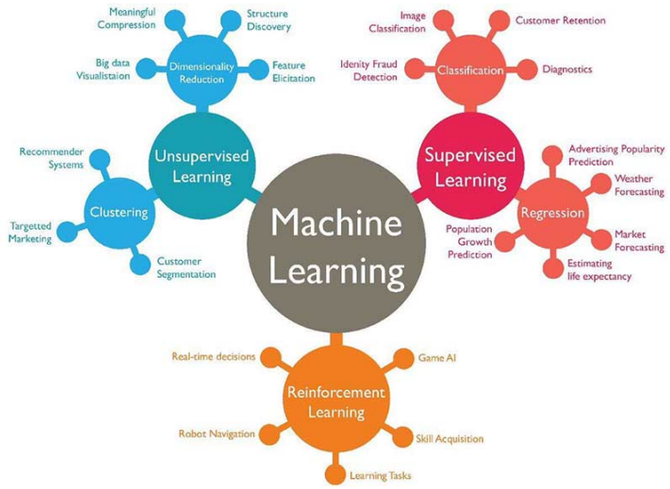


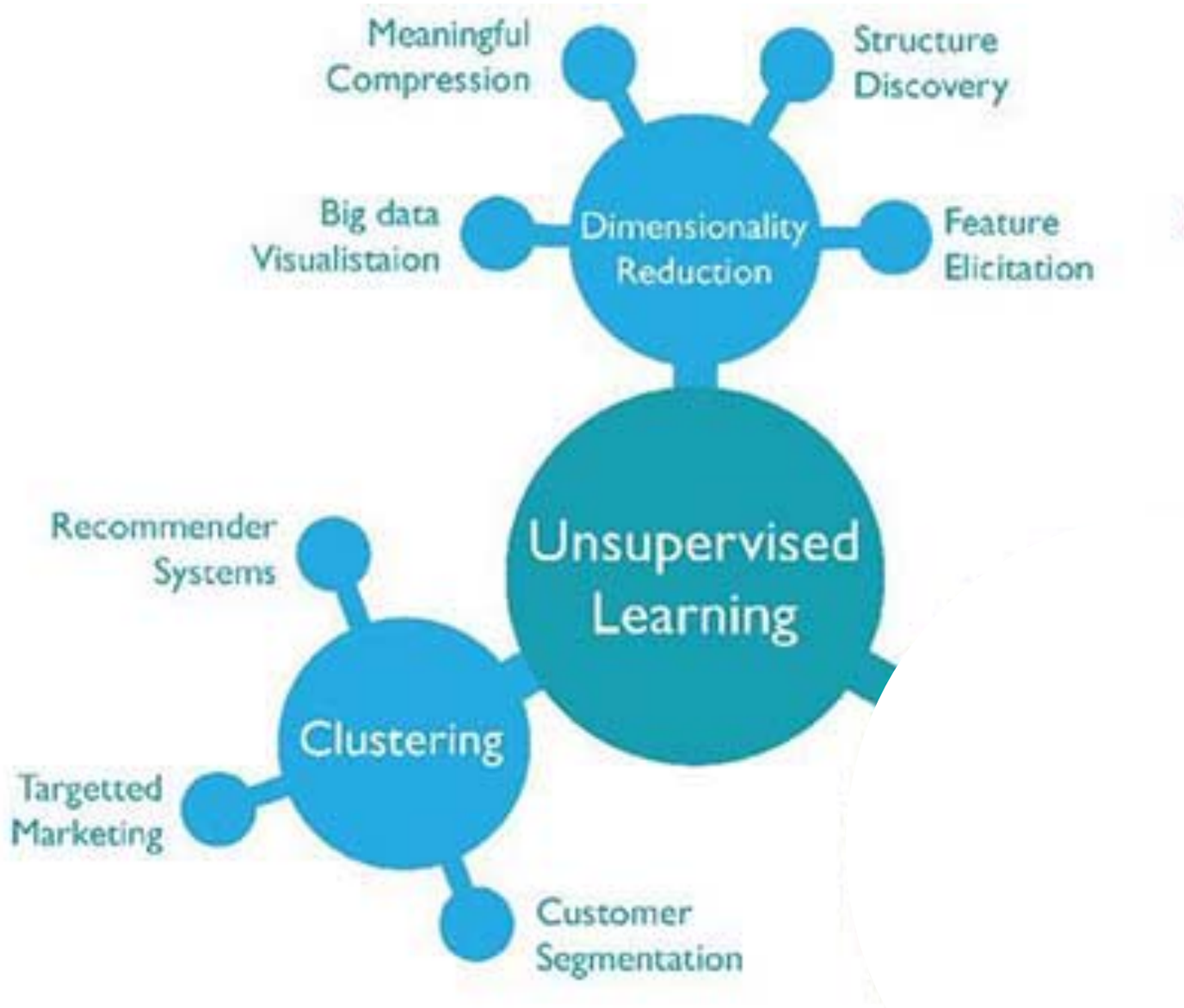




recognize hand-written digits

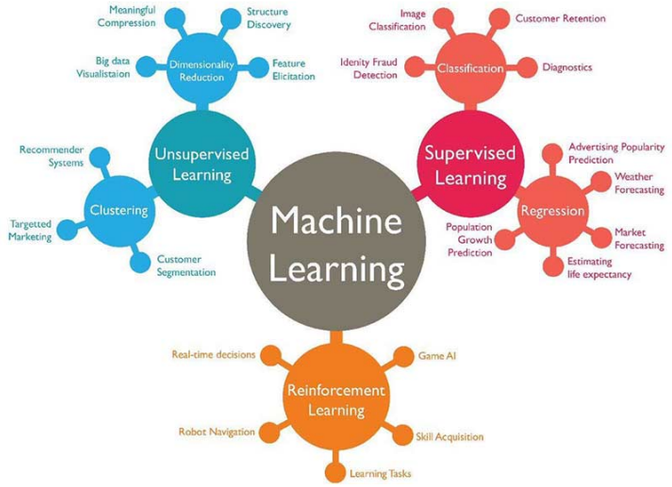
Unsupervised Learning





learning the structure of unlabeled data

Unsupervised Learning



learning the structure of unlabeled data

Neural Quantum States

recall: variational wave functions

$$|\psi(\theta)\rangle$$

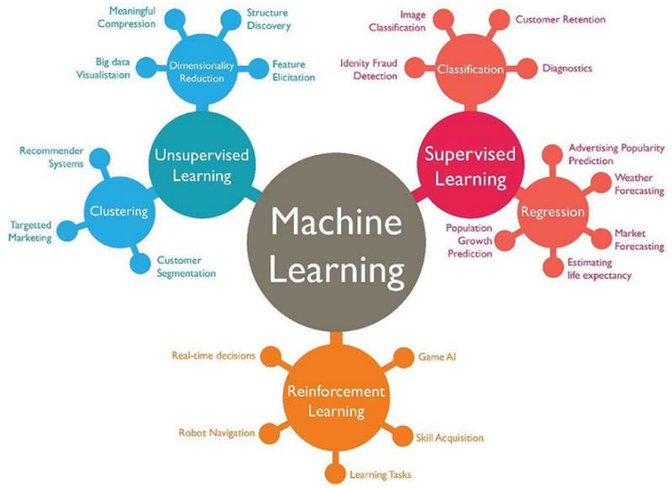
$$\psi(x;\theta) \propto \exp\left(-\frac{(x-\theta_1)^2}{2\theta_2^2}\right)$$

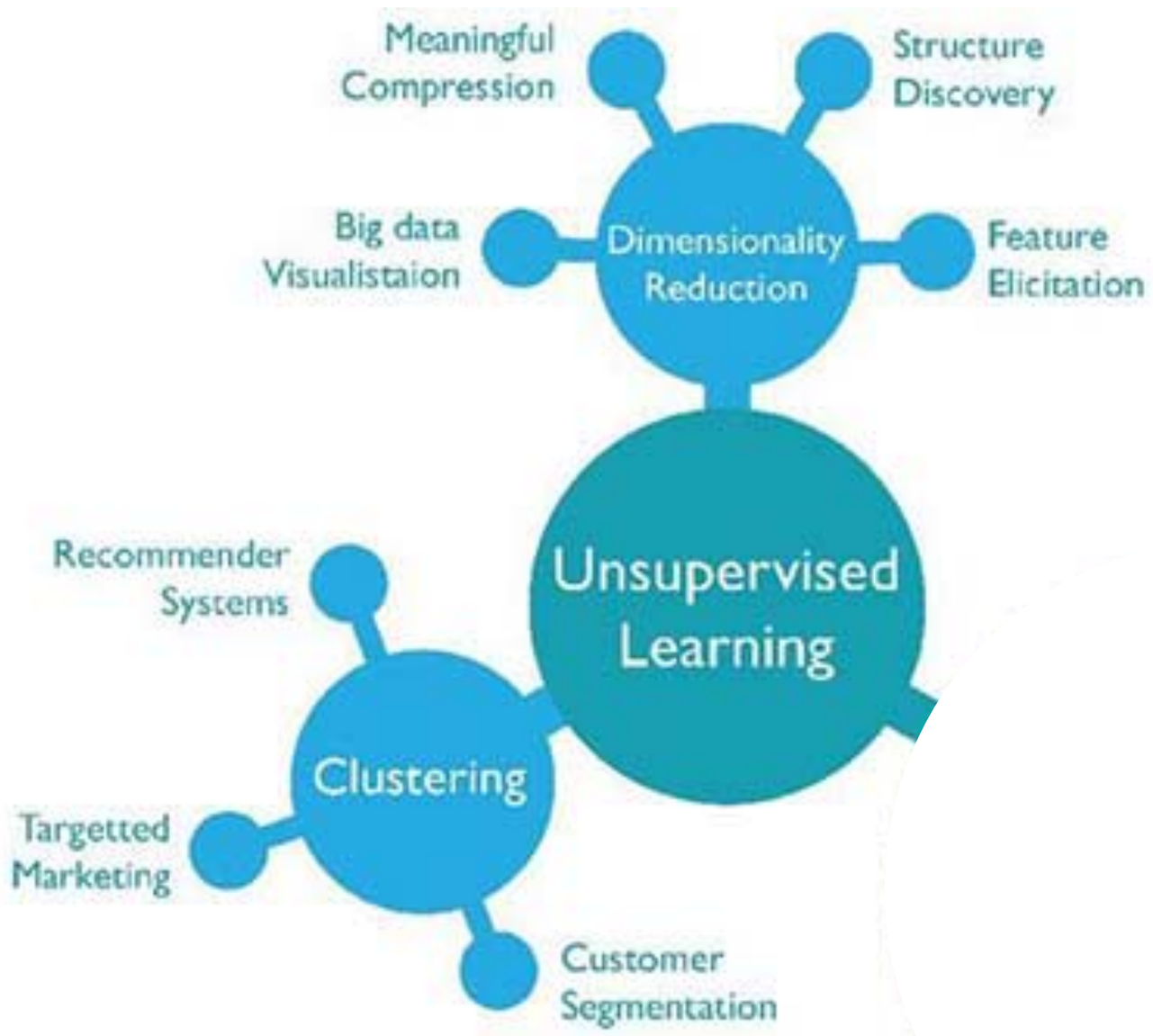
$$E(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle$$

$$\partial_\theta E(\theta) \stackrel{!}{=} 0$$

Carleo and Troyer, Science 2017
Schmitt and Heyl, PRL 2020
+ many more

Unsupervised Learning

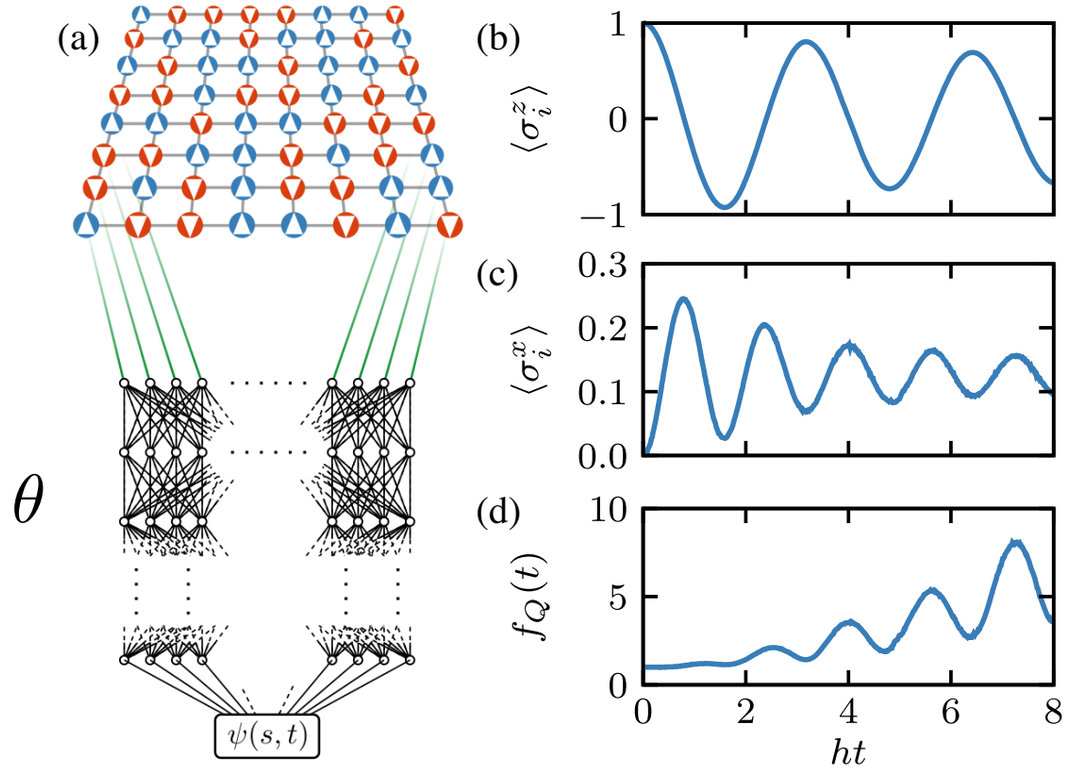




learning the structure of unlabeled data

Neural Quantum States $|\psi(\theta)\rangle$

$$E(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle$$



$$\psi(x;\theta) \propto \exp\left(-\frac{(x-\theta_1)^2}{2\theta_2^2}\right)$$

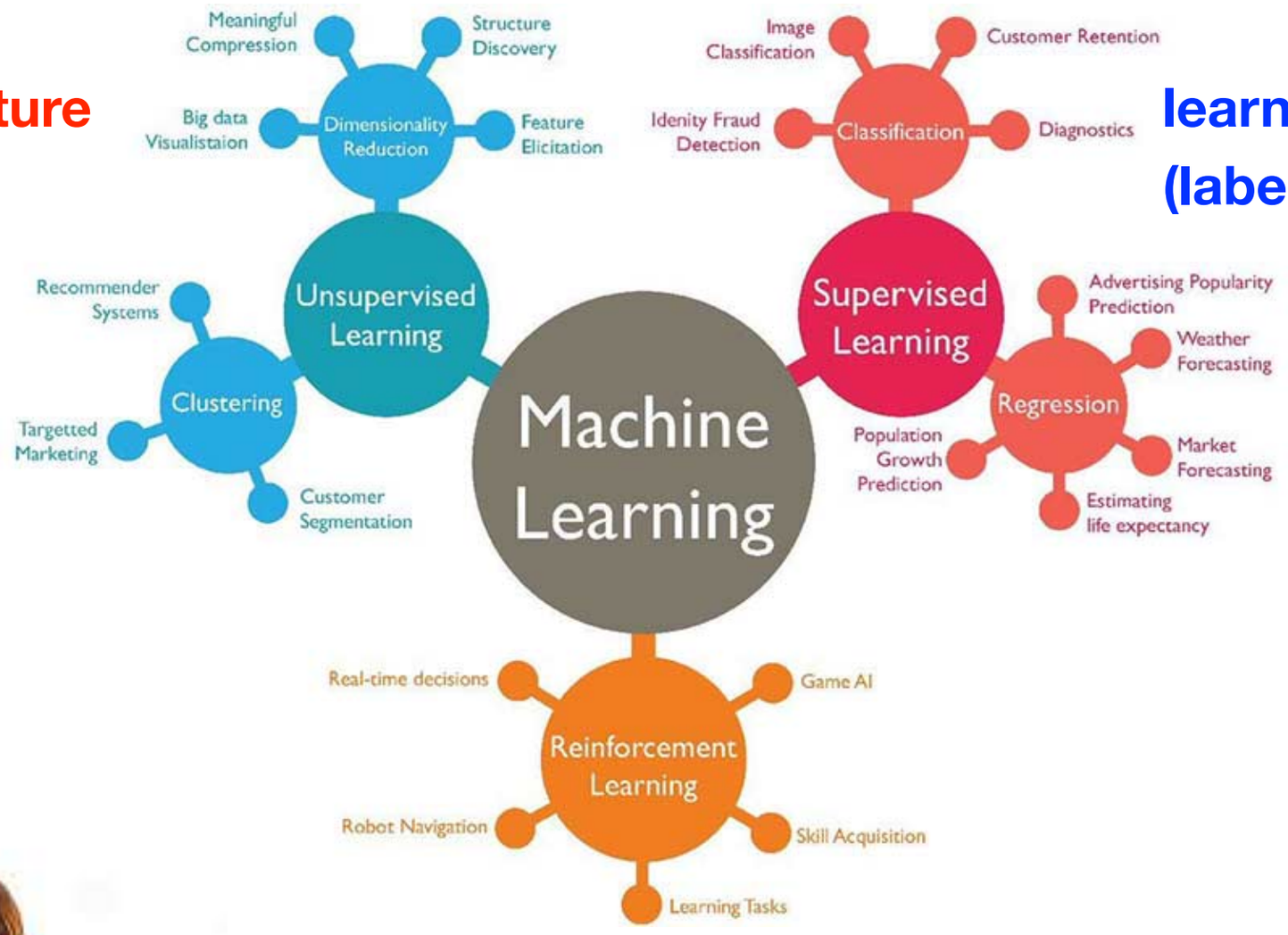
Carleo and Troyer, Science 2017

Schmitt and Heyl, PRL 2020

+ many more

Reinforcement Learning

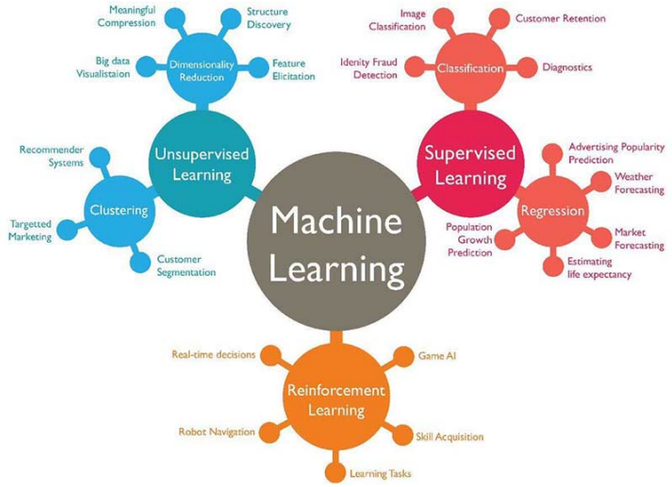
learning the structure of unlabeled data



learning from examples (labeled data)

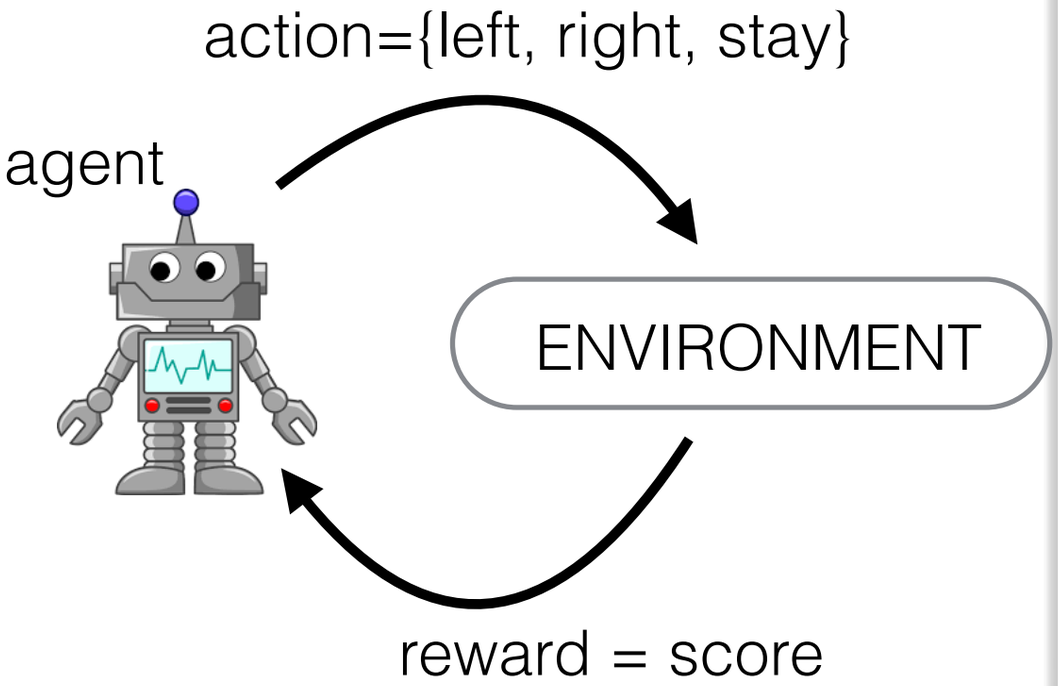
learning from experience

Overview



- What is Reinforcement Learning (RL)?
- RL framework in a nutshell
- RL algorithms
- Applications of RL in quantum physics

What is Reinforcement Learning?



What is Reinforcement Learning Useful for?

Mastering the game of Go with deep neural networks and tree search

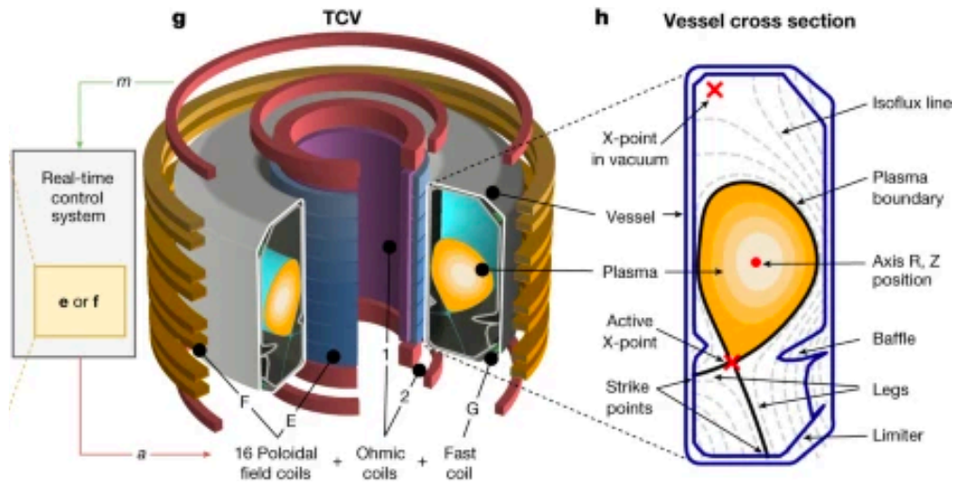
Silver, et. al, Nature *529* 484–489 (2016)

What is Reinforcement Learning Useful for?

Mastering the game of Go with deep neural networks and tree search

Silver, et. al, Nature *529* 484–489 (2016)

Magnetic control of tokamak plasmas through deep RL



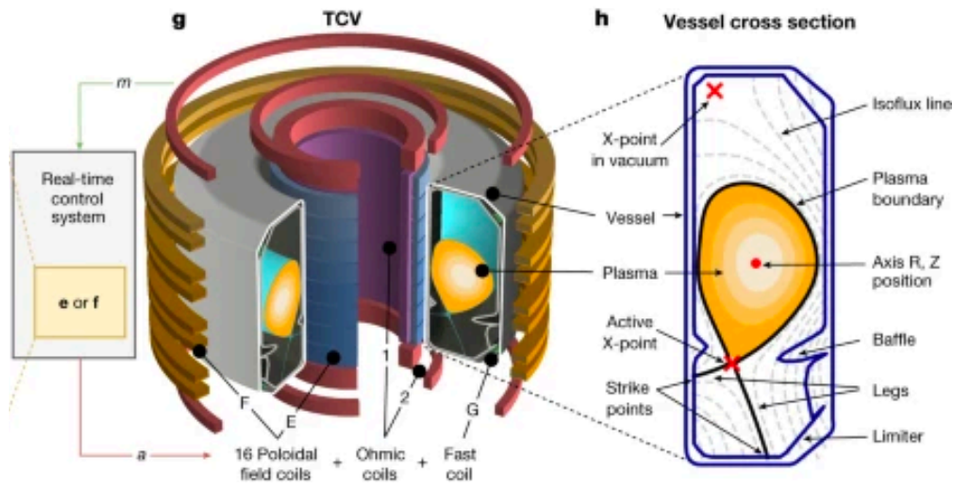
Degrave, et. al, Nature *602* 414–419 (2022)

What is Reinforcement Learning Useful for?

Mastering the game of Go with deep neural networks and tree search

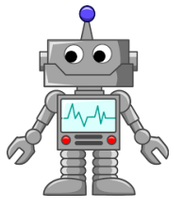
Silver, et. al, Nature *529* 484–489 (2016)

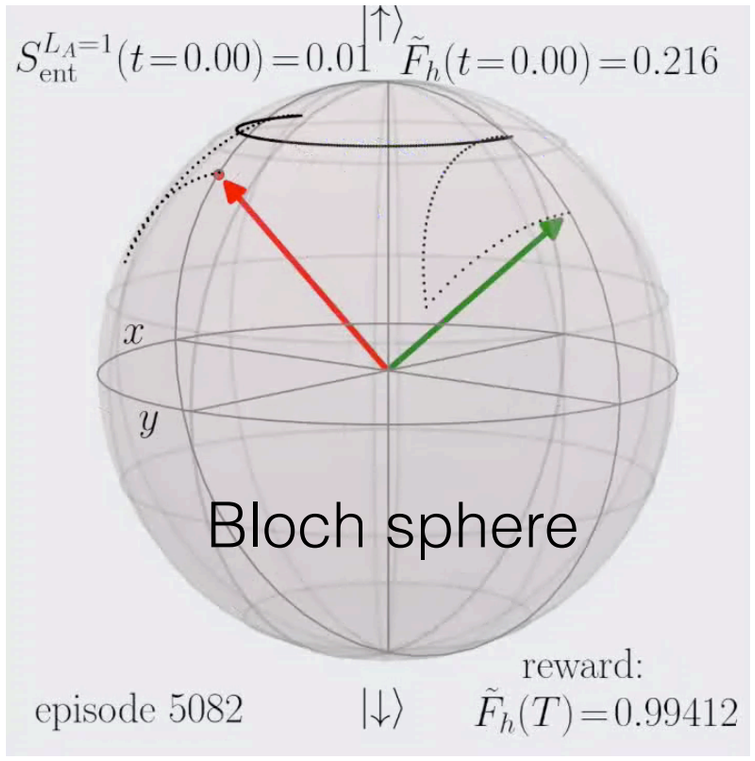
Magnetic control of tokamak plasmas through deep RL



Degrave, et. al, Nature *602* 414–419 (2022)

RL-aided quantum technologies





quantum control

quantum error correction

quantum gate design

quantum circuit design

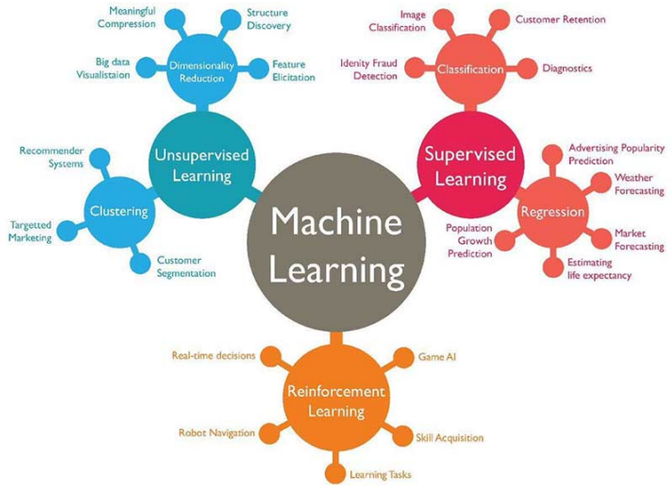
When should you consider using RL?

- *model-free:* requires no pre-knowledge of the controlled physical system

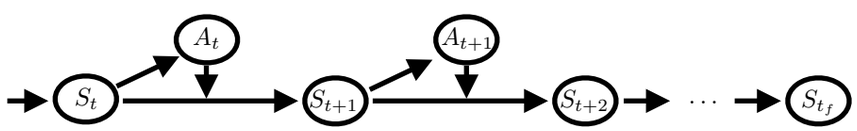
- *adaptive:* transfers acquired knowledge that might allow us to identify connections between unrelated phenomena

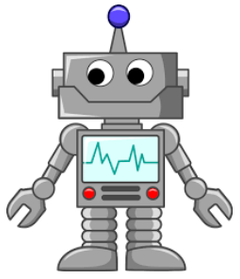
- *autonomous:* provides novel insights into automating complex manipulation protocols

Overview



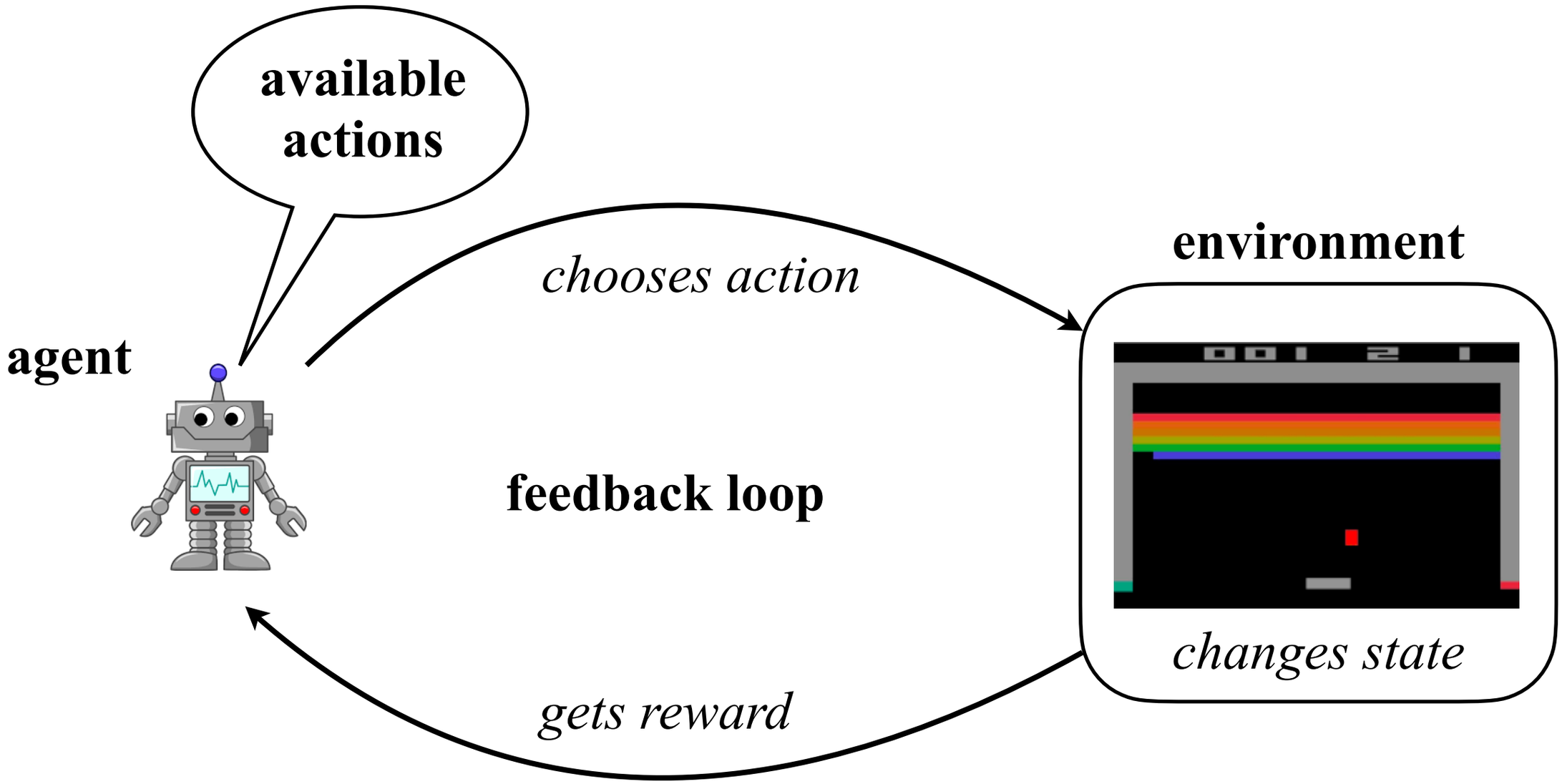
- What is Reinforcement Learning (RL)?
- RL framework in a nutshell
- RL algorithms
- Applications of RL in quantum physics





Reinforcement Learning (RL) in a Nutshell

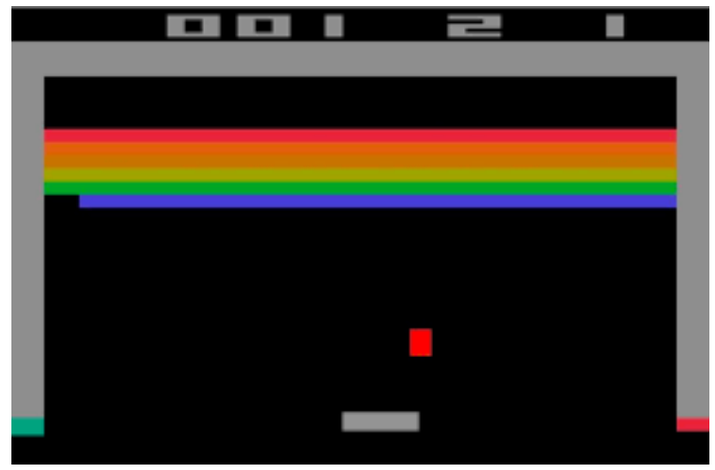
→ RL formalism



RL in a Nutshell

→ RL formalism

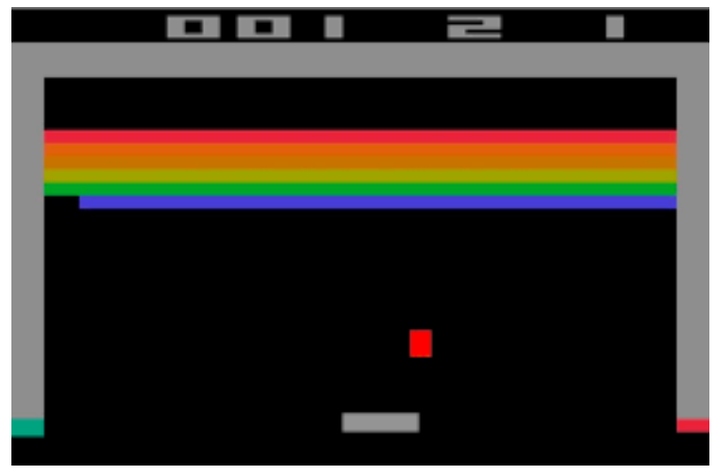
- action space $\mathcal{A} = \{\text{left},\ \text{stay},\ \text{right}\}$
- state space $\mathcal{S}$ pixelized image of the screen
- reward space $\mathcal{R} = \text{score}$



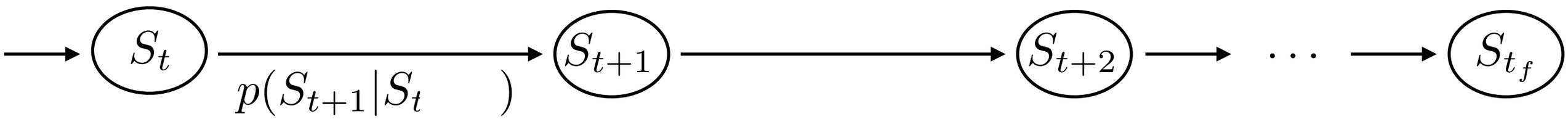
RL in a Nutshell

→ RL formalism

- action space $\mathcal{A} = \{\text{left},\ \text{stay},\ \text{right}\}$
- state space $\mathcal{S}$ pixelized image of the screen
- reward space $\mathcal{R} = \text{score}$



→ RL episode is a Markov decision process

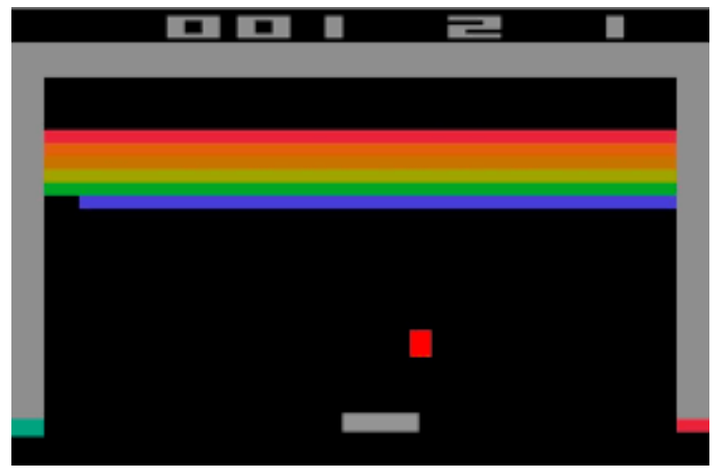


- transition probability: $p(S_{t+1}|S_t\quad)$

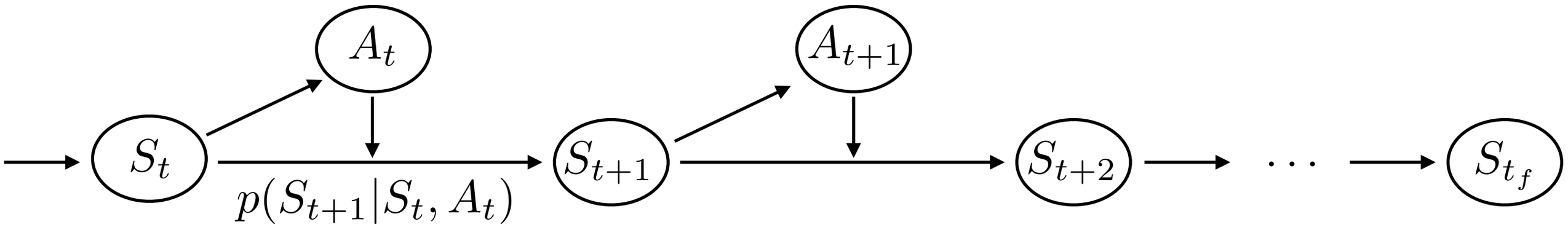
RL in a Nutshell

→ RL formalism

- action space $\mathcal{A} = \{\text{left},\ \text{stay},\ \text{right}\}$
- state space $\mathcal{S}$ pixelized image of the screen
- reward space $\mathcal{R} = \text{score}$



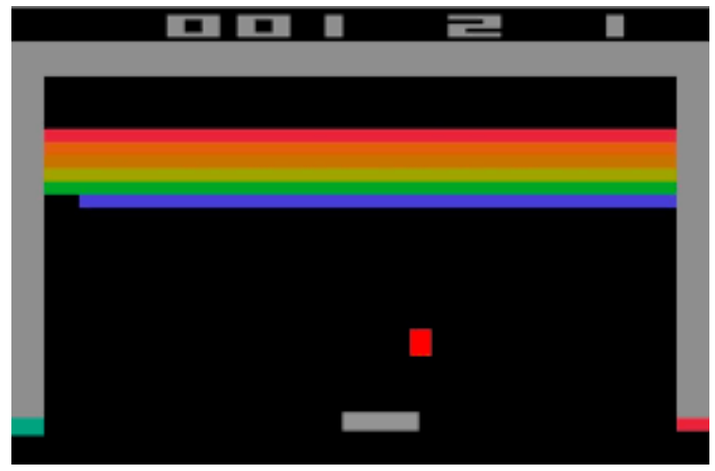
→ RL episode is a Markov **decision** process



- transition probability: $p(S_{t+1}|S_t, A_t)$

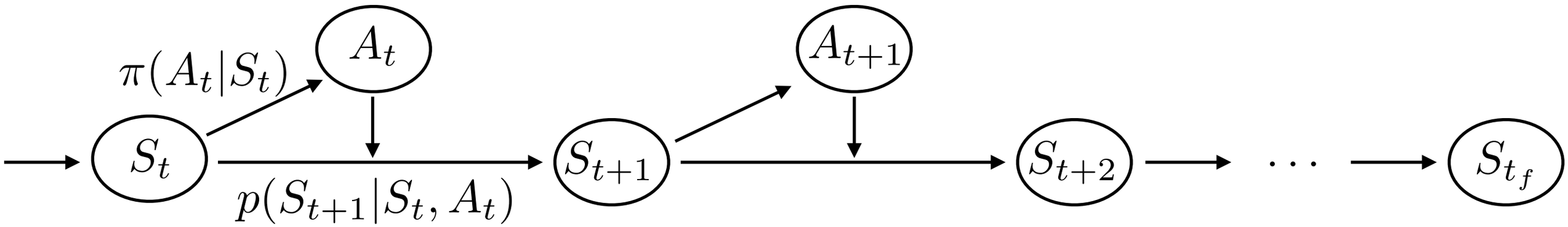
RL in a Nutshell

→ RL formalism



- action space $\mathcal{A} = \{\text{left},\ \text{stay},\ \text{right}\}$
- state space $\mathcal{S}$ pixelized image of the screen
- reward space $\mathcal{R} = \text{score}$

→ RL episode is a Markov **decision** process

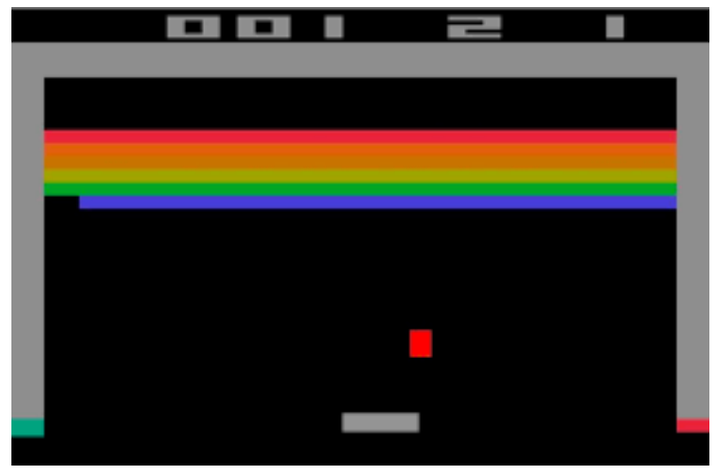


- transition probability: $p(S_{t+1}|S_t, A_t)$
- policy: $\pi(A_t|S_t)$ probability to take action A_t in the state S_t

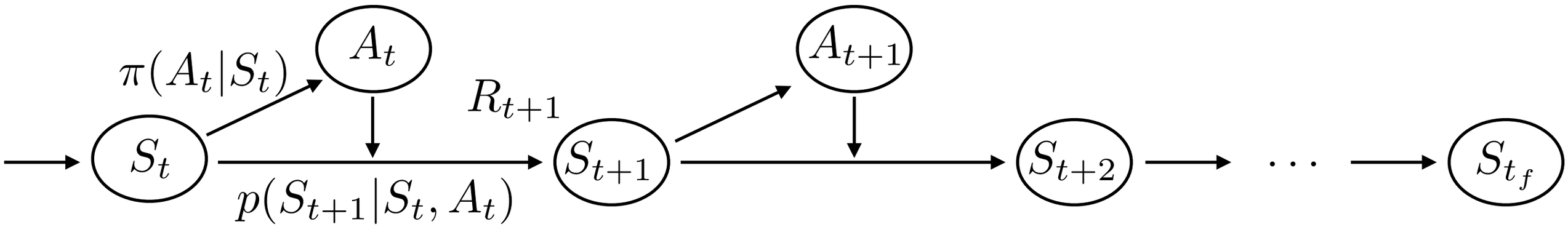
RL in a Nutshell

→ RL formalism

- action space $\mathcal{A} = \{\text{left},\ \text{stay},\ \text{right}\}$
- state space $\mathcal{S}$ pixelized image of the screen
- reward space $\mathcal{R} = \text{score}$



→ RL episode is a Markov **decision** process

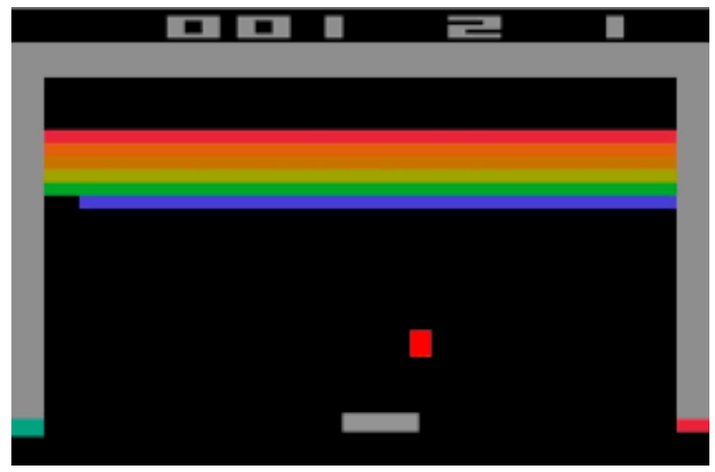


- transition probability: $p(S_{t+1}|S_t, A_t)$
- policy: $\pi(A_t|S_t)$ probability to take action A_t in the state S_t

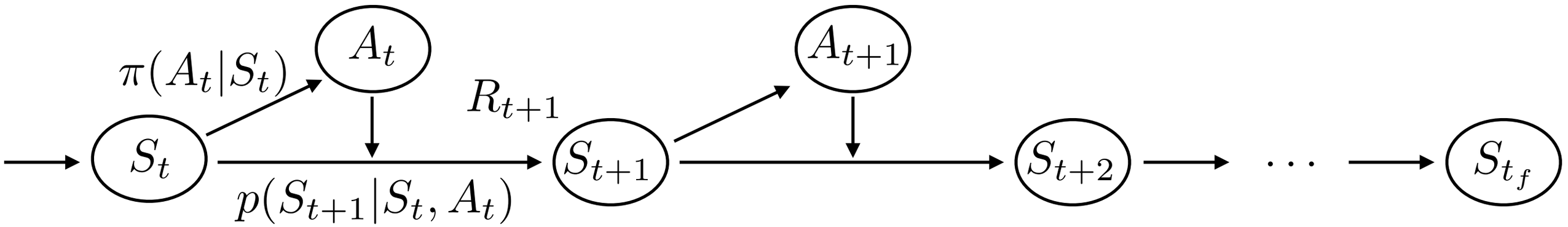
RL in a Nutshell

→ RL formalism

- action space $\mathcal{A} = \{\text{left},\ \text{stay},\ \text{right}\}$
- state space $\mathcal{S}$ pixelized image of the screen
- reward space $\mathcal{R} = \text{score}$



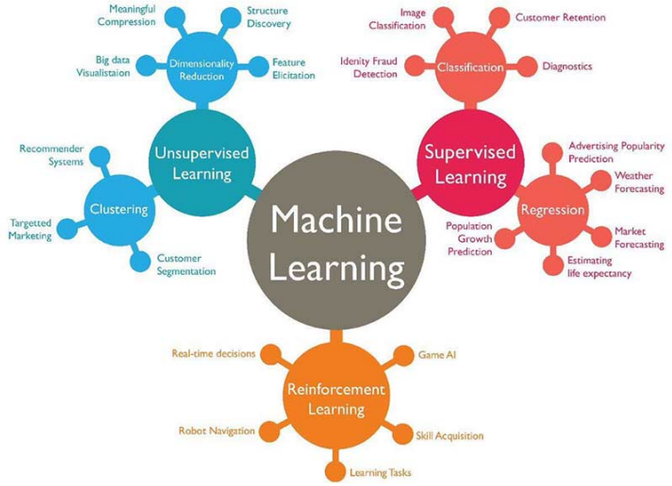
→ RL episode is a Markov **decision** process



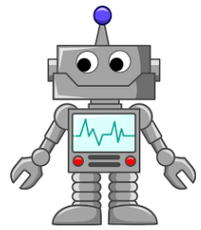
→ RL **objective**: *find policy* which maximizes the total *expected return*

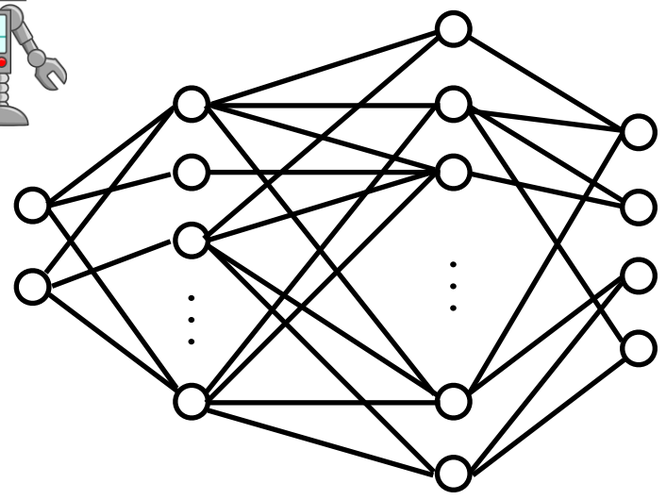
$$J = \mathbb{E}_{a\sim\pi(a|s)}[R_{t+1} + \cdots + R_{t_f}|S_0 = s]$$

Overview

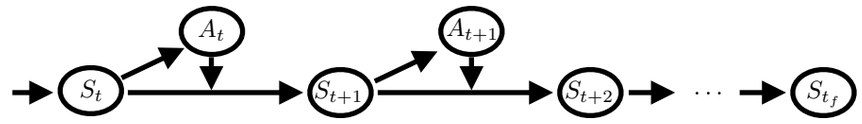


- What is Reinforcement Learning (RL)?
- RL framework in a nutshell
- RL algorithms
- Applications of RL in quantum physics





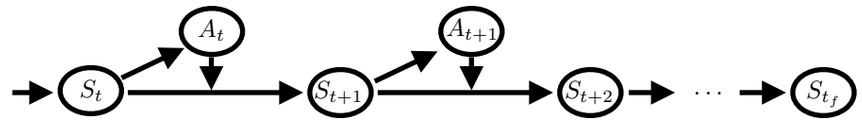
Policy Gradient



→ *in practice:* **evaluate** RL objective by sampling trajectories

$$J = \mathbb{E}_{a\sim\pi(a|s)}\left[\sum_t R_t \middle| S_0 = s\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}}\sum_t R_t(\tau_j)$$

Policy Gradient



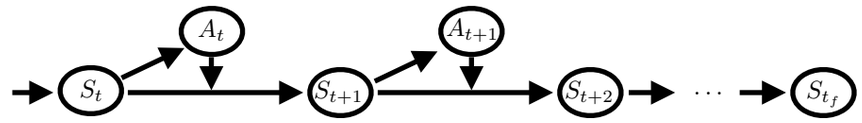
→ *in practice:* **evaluate** RL objective by sampling trajectories

$$J = \mathbb{E}_{a\sim\pi(a|s)}\left[\sum_t R_t \middle| S_0 = s\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}}\sum_t R_t(\tau_j)$$

→ **improve** policy

- parametrize policy: $\pi(a|s) \approx \pi_\theta(a|s)$

Policy Gradient



→ *in practice:* **evaluate** RL objective by sampling trajectories

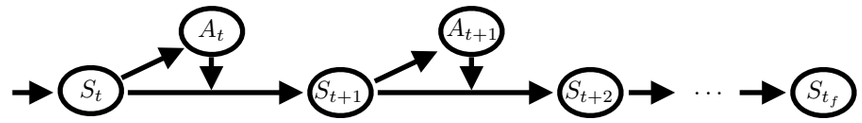
$$J = \mathbb{E}_{a\sim\pi(a|s)}\left[\sum_t R_t \middle| S_0 = s\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}}\sum_t R_t(\tau_j)$$

→ **improve** policy

- parametrize policy: $\pi(a|s) \approx \pi_\theta(a|s)$
- compute gradients:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log\pi_\theta \sum_t R_t \middle| S_0 = s\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}}\nabla_\theta \log\pi_\theta(\tau_j)\sum_t R_t(\tau_j)$$

Policy Gradient



→ *in practice:* **evaluate** RL objective by sampling trajectories

$$J = \mathbb{E}_{a\sim\pi(a|s)}\left[\sum_t R_t \middle| S_0 = s\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}}\sum_t R_t(\tau_j)$$

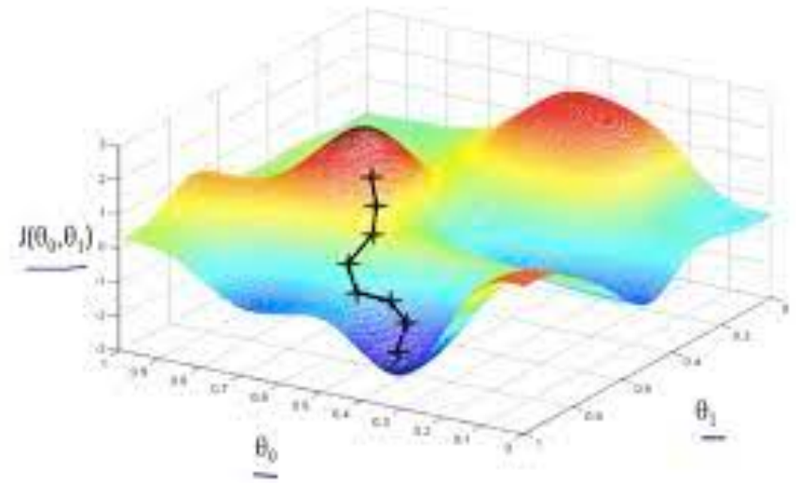
→ **improve** policy

- parametrize policy: $\pi(a|s) \approx \pi_\theta(a|s)$
- compute gradients:

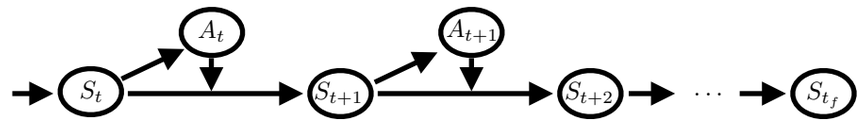
$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log\pi_\theta \sum_t R_t \middle| S_0 = s\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}}\nabla_\theta \log\pi_\theta(\tau_j)\sum_t R_t(\tau_j)$$

- update parameters θ to maximize the reward

$$\theta_{\mathrm{new}} = \theta_{\mathrm{old}} + \alpha\nabla_\theta J(\theta)$$



Policy Gradient

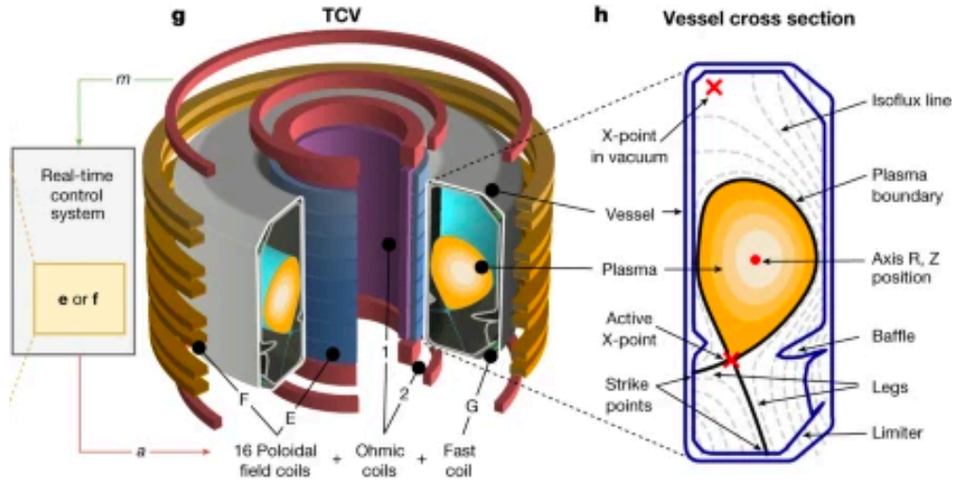


→ *in practice:* **evaluate** RL objective by sampling trajectories

$$J = \mathbb{E}_{a\sim\pi(a|s)}\left[\sum_t R_t \middle| S_0 = s\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}}\sum_t R_t(\tau_j)$$

→ **improve** policy

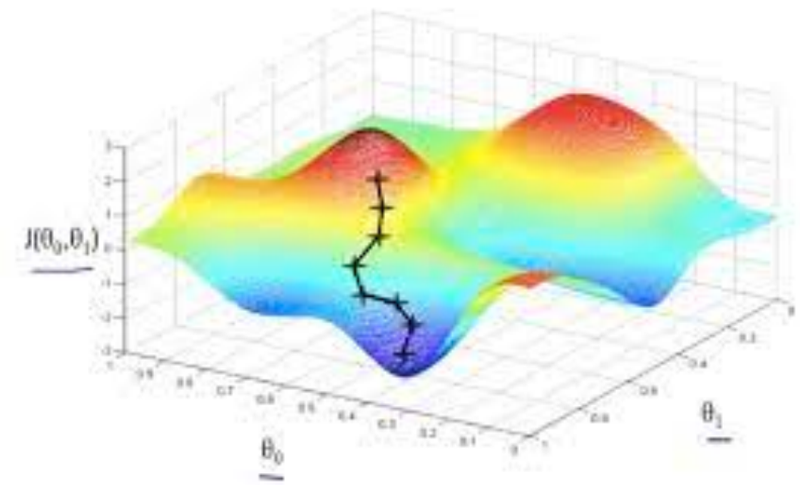
- parametrize policy: $\pi(a|s) \approx \pi_\theta(a|s)$
- compute gradients:



$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log\pi_\theta \sum_t R_t \middle| S_0 = s\right] \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}}\nabla_\theta \log\pi_\theta(\tau_j)\sum_t R_t(\tau_j)$$

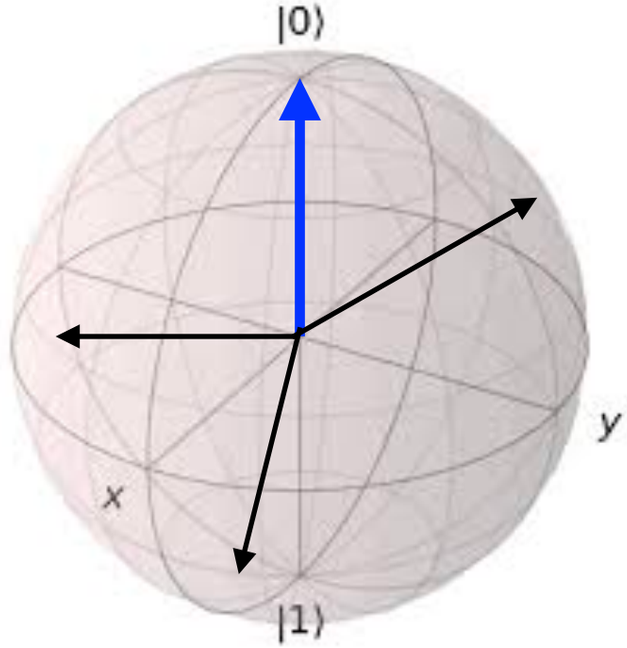
- update parameters θ to maximize the reward

$$\theta_{\mathrm{new}} = \theta_{\mathrm{old}} + \alpha\nabla_\theta J(\theta)$$



RL for Quantum State Initialization

Bloch sphere



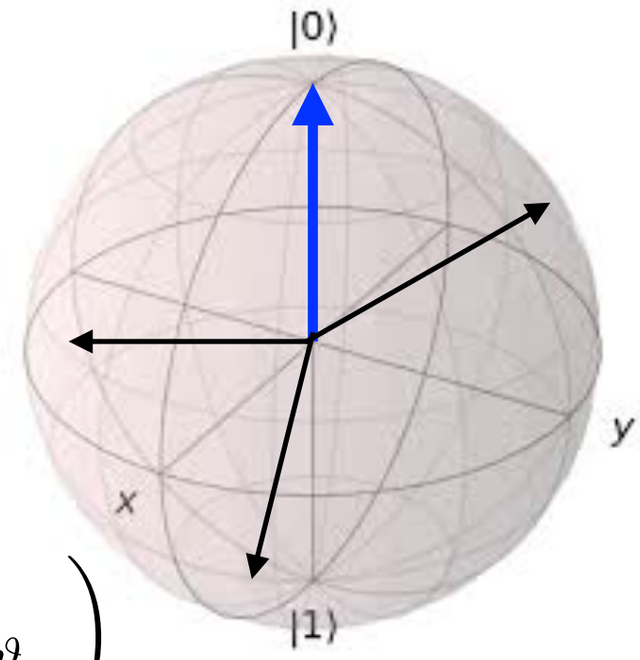
→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

RL for Quantum State Initialization

Bloch sphere



→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

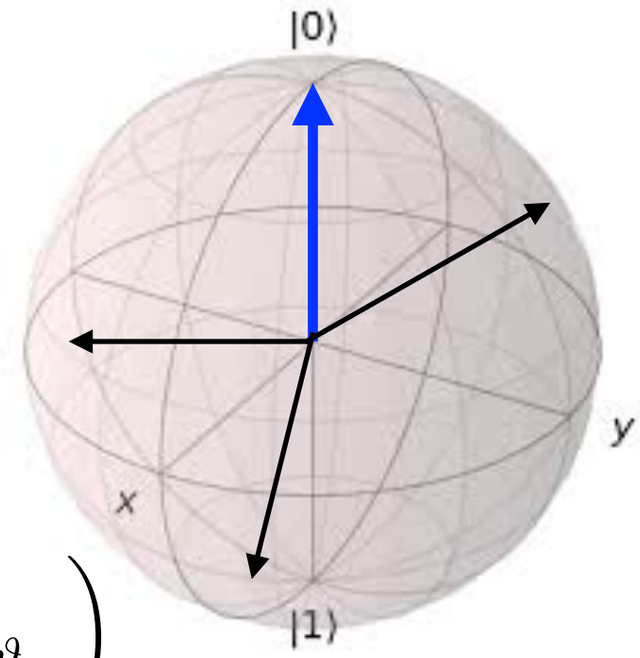
- define **RL states**:

parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

RL state space: $\mathcal{S} = \{(\vartheta, \varphi) | \vartheta \in [0, \pi],\ \varphi \in [0, 2\pi)\}$

RL for Quantum State Initialization

Bloch sphere



→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

- define **RL states**:

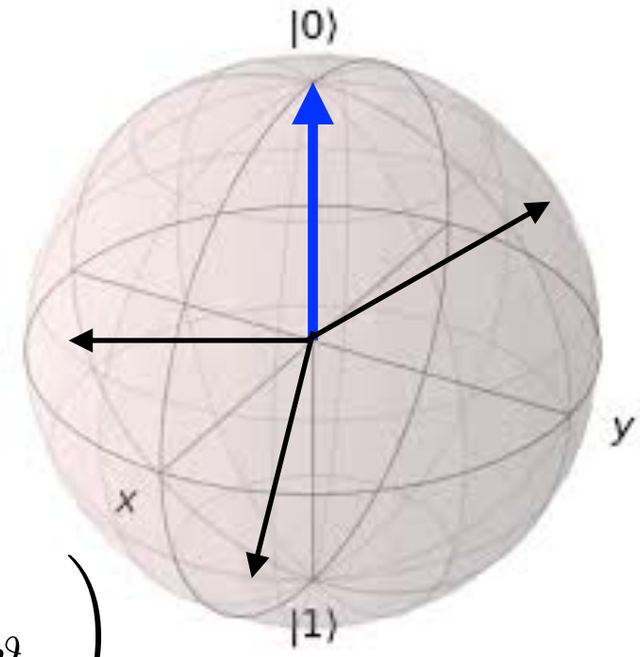
parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

RL state space: $\mathcal{S} = \{(\vartheta, \varphi)|\vartheta \in [0, \pi],\ \varphi \in [0, 2\pi)\}$

- define **RL actions**: $\mathcal{A} = \{I, U_\alpha|\alpha = x, y, z\}$ $\qquad |\psi'\rangle = U_\alpha|\psi\rangle$ $\qquad s \mapsto s'$

RL for Quantum State Initialization

Bloch sphere



→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

- define **RL states**:

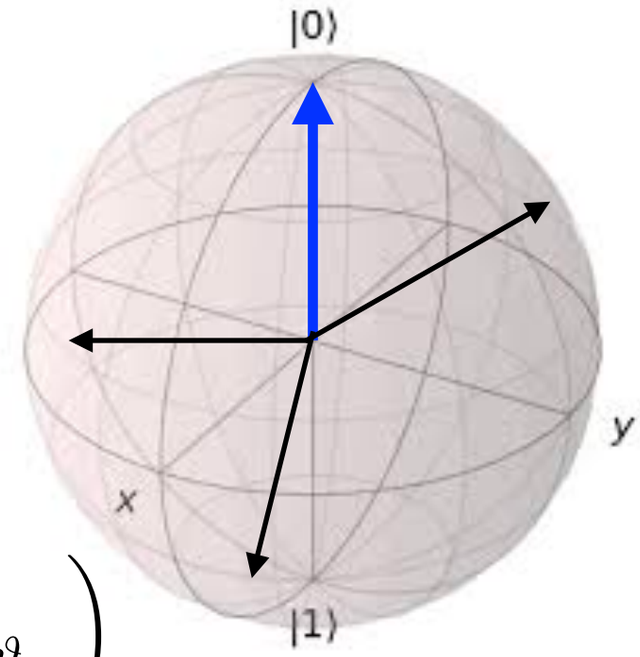
parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

RL state space: $\mathcal{S} = \{(\vartheta, \varphi)|\vartheta \in [0, \pi],\ \varphi \in [0, 2\pi)\}$

- define **RL actions**: $\mathcal{A} = \{I, U_\alpha|\alpha = x, y, z\}$ $\qquad |\psi'\rangle = U_\alpha|\psi\rangle$ $\qquad s \mapsto s'$
- define reward: $r_t = |\langle 0|\psi_t\rangle|^2$

RL for Quantum State Initialization

Bloch sphere



→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

- define **RL states**:

parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

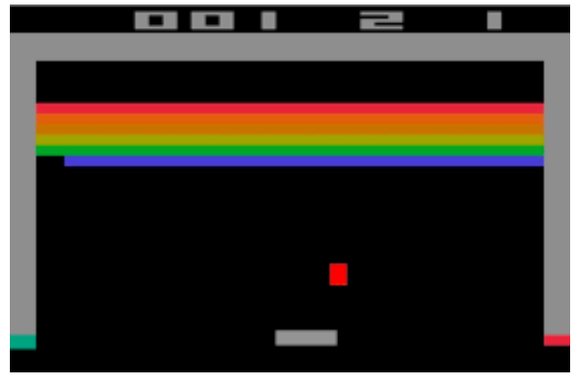
RL state space: $\mathcal{S} = \{(\vartheta, \varphi)|\vartheta \in [0, \pi],\ \varphi \in [0, 2\pi)\}$

- define **RL actions**: $\mathcal{A} = \{I, U_\alpha|\alpha = x, y, z\}$ $\qquad |\psi'\rangle = U_\alpha|\psi\rangle$ $\qquad s \mapsto s'$
- define reward: $r_t = |\langle 0|\psi_t\rangle|^2$
- choose **RL algorithm:** policy gradient

simulate trial trajectories τ_j

$$\nabla_\theta J(\theta) \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}} \nabla_\theta \log \pi_\theta(\tau_j) \sum_t R_t(\tau_j)$$

Value Function Algorithms



→ Value iteration methods

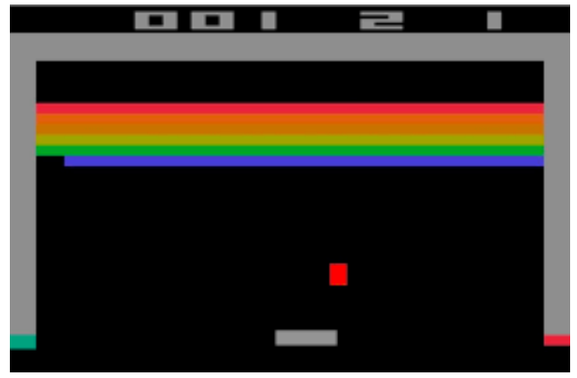
- value function: ***expected*** total return under the policy $\pi(a|s)$ from state s

$$v_\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}[G_t|S_t = s] \qquad G_t = R_{t+1} + R_{t+2} + \dots$$

$$= R_{t+1} + G_{t+1}$$

	$v_\pi(s)$
S_1	1.3
S_2	2.1
S_3	1.5

Value Function Algorithms



→ Value iteration methods

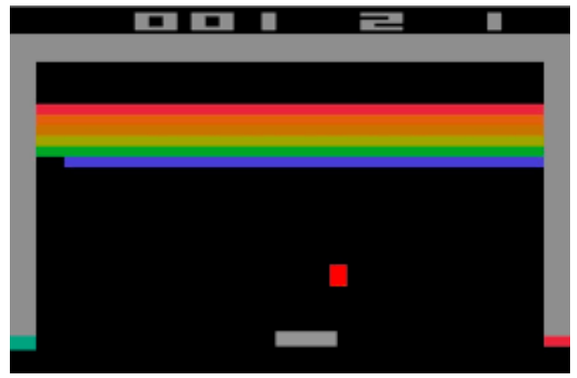
- value function: ***expected*** total return under the policy $\pi(a|s)$ from state s

$$v_\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}[G_t|S_t = s] \qquad G_t = R_{t+1} + G_{t+1}$$

problem: cannot reconstruct policy from value function

	$v_\pi(s)$
S_1	1.3
S_2	2.1
S_3	1.5

Value Function Algorithms



→ Value iteration methods

- value function: ***expected*** total return under the policy $\pi(a|s)$ from state s

$$v_\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}[G_t|S_t = s] \qquad G_t = R_{t+1} + G_{t+1}$$

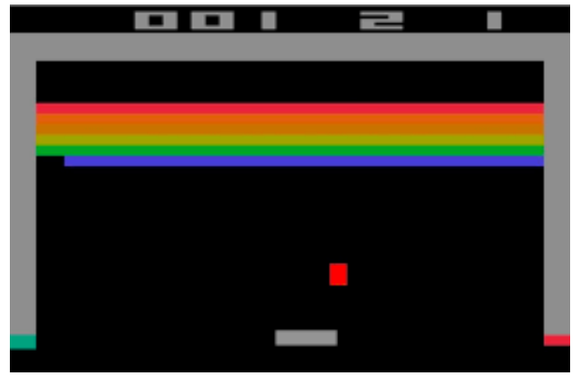
- action-value (or Q-) function: ***expected*** total return under the policy $\pi(a|s)$ starting from state s and taking action a:

$$Q_\pi(s,a) = \mathbb{E}_{a\sim\pi(a|s)}[G_t|S_t = s, A_t = a]$$

	$v_\pi(s)$
S_1	1.3
S_2	2.1
S_3	1.5

$Q(s,a)$	A_1	A_2	A_3
S_1	1.5	0.5	2.1
S_2	1.3	4.4	0.2
S_3	0.9	0.9	3.3

Value Function Algorithms



→ Value iteration methods

- value function: ***expected*** total return under the policy $\pi(a|s)$ from state s

$$v_\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}[G_t|S_t = s] \qquad G_t = R_{t+1} + G_{t+1}$$

- action-value (or Q-) function: ***expected*** total return under the policy $\pi(a|s)$ starting from state s and taking action a:

$$Q_\pi(s,a) = \mathbb{E}_{a\sim\pi(a|s)}[G_t|S_t = s, A_t = a] = \sum_{s'} p(s'|s,a) \times [r(s',a) + v_\pi(s')]$$

	$v_\pi(s)$
S_1	1.3
S_2	2.1
S_3	1.5

$$v_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a)$$

$Q(s,a)$	A_1	A_2	A_3
S_1	1.5	0.5	2.1
S_2	1.3	4.4	0.2
S_3	0.9	0.9	3.3

Q-Learning

- optimal action-value function: $Q_*(s,a) = \max_\pi Q_\pi(s,a)$
- optimal value function: $v_*(s) = \max_a Q_*(s,a)$
- optimal policy: $\pi_*(a|s) = \text{argmax}_{a'} Q_*(s,a')$

Q-Learning

→ optimal action-value function: $Q_*(s,a) = \max_\pi Q_\pi(s,a)$

→ optimal value function: $v_*(s) = \max_a Q_*(s,a)$

→ optimal policy: $\pi_*(a|s) = \text{argmax}_{a'} Q_*(s,a')$

$$Q_\pi(s,a) = \mathbb{E}_{a\sim\pi(a|s)}[G_t|S_t = s, A_t = a] = \sum_{s'} p(s'|s,a) \times [r(s,s',a) + v_\pi(s')]$$

Bellman's equation: $Q_*(s,a) = \sum_{s'} p(s'|s,a) \left[r(s,s',a) + \max_{a'} Q_*(s',a') \right]$

appears on both sides of equation; compute recursively

Q-Learning

→ optimal action-value function: $Q_*(s,a) = \max_\pi Q_\pi(s,a)$

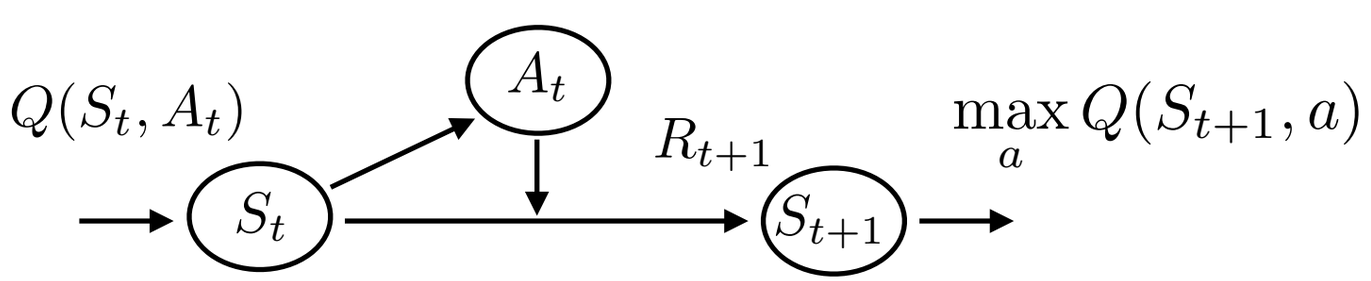
$\pi_*(a|s) = \text{argmax}_{a'} Q_*(s,a')$

Bellman's equation: $Q_*(s,a) = \sum_{s'} p(s'|s,a)\left[r(s,s',a) + \max_{a'} Q_*(s',a')\right]$

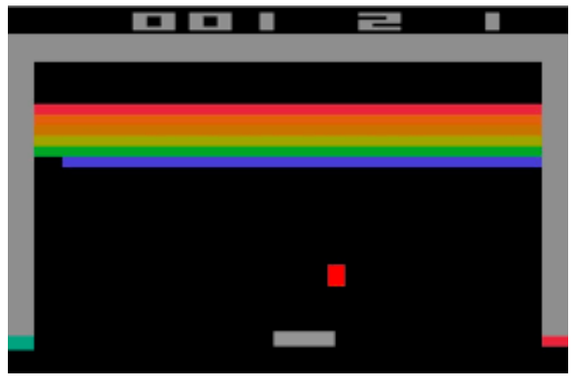
appears on both sides of equation; compute recursively

→ Q-learning: iterate following two steps until convergence

1. error according to definition $\delta_t = Q(S_t,A_t) + R_{t+1} - \max_a Q(S_{t+1},a)$
2. update current function $Q_{\text{new}} = Q_{\text{old}} + \alpha\delta_t \qquad \alpha \in [0,1)$



Q-Learning



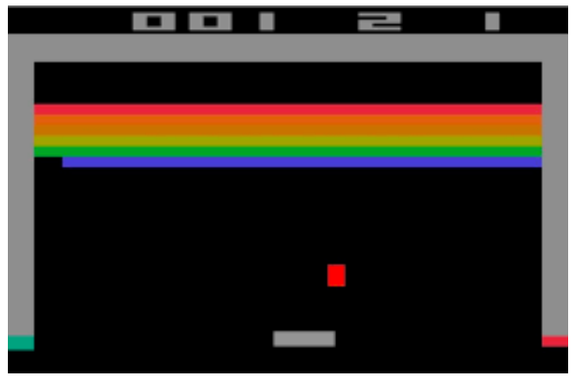
→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 1

$Q(s,a)$	A_1	A_2	A_3
S_1	1.5	0.5	2.1
S_2	0.3	4.4	0.7
S_3	0.9	0.9	1.3

Q-Learning



→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 1

$Q(s,a)$	A_1	A_2	A_3
S_1	1.5	0.5	2.1
S_2	0.3	4.4	0.7
S_3	0.9	0.9	1.3

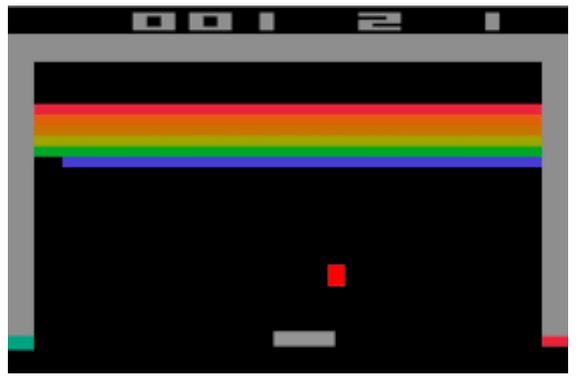
$$A = \text{argmax}_a Q(S, a)$$

$$\pi(S_1) = A_3$$

$$\pi(S_2) = A_2$$

$$\pi(S_3) = A_3$$

Q-Learning



→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 1

$Q(s,a)$	A_1	A_2	A_3
S_1	1.5	0.5	2.1
S_2	0.3	4.4	0.7
S_3	0.9	0.9	1.3

$$A = \text{argmax}_a Q(S, a)$$

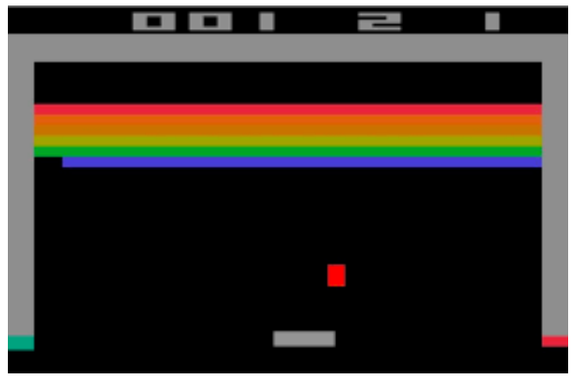
$$\pi(S_1) = A_3$$

$$\pi(S_2) = A_2$$

$$\pi(S_3) = A_3$$

1. error according to definition $\delta_t = Q(S_t, A_t) + R_{t+1} - \max_a Q(S_{t+1}, a)$
2. update current function $Q_{\text{new}} = Q_{\text{old}} + \alpha\delta_t \qquad \alpha \in [0, 1)$

Q-Learning



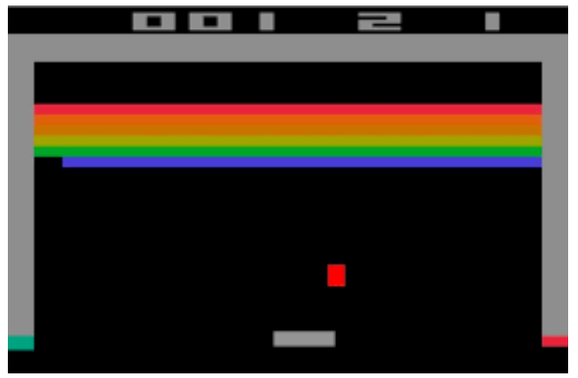
→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 2

$Q(s,a)$	A_1	A_2	A_3
S_1	1.6	0.4	1.8
S_2	0.6	2.8	0.1
S_3	0.6	1.1	1.0

Q-Learning



→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 2

$Q(s,a)$	A_1	A_2	A_3
S_1	1.6	0.4	1.8
S_2	0.6	2.8	0.1
S_3	0.6	1.1	1.0

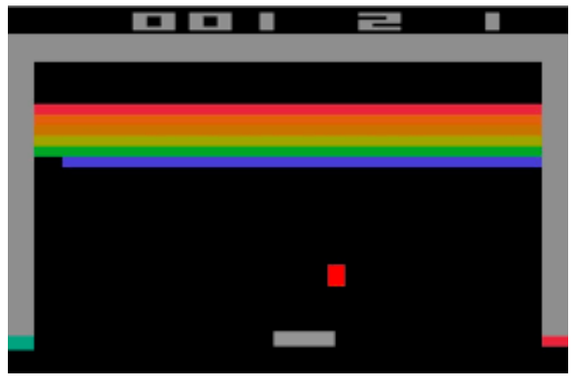
$$A = \text{argmax}_a Q(S, a)$$

$$\pi(S_1) = A_3$$

$$\pi(S_2) = A_2$$

$$\pi(S_3) = A_2$$

Q-Learning



→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 2

$Q(s,a)$	A_1	A_2	A_3
S_1	1.6	0.4	1.8
S_2	0.6	2.8	0.1
S_3	0.6	1.1	1.0

$$A = \text{argmax}_a Q(S, a)$$

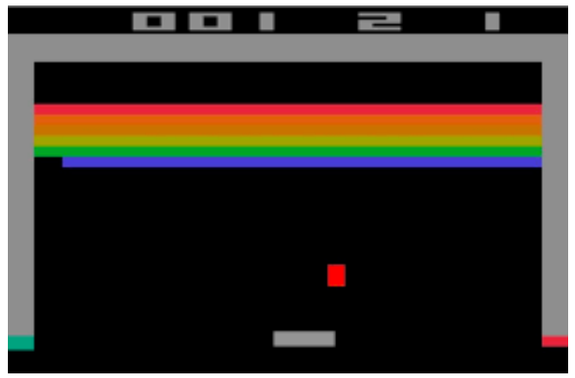
$$\pi(S_1) = A_3$$

$$\pi(S_2) = A_2$$

$$\pi(S_3) = A_2$$

1. error according to definition $\delta_t = Q(S_t, A_t) + R_{t+1} - \max_a Q(S_{t+1}, a)$
2. update current function $Q_{\text{new}} = Q_{\text{old}} + \alpha\delta_t \qquad \alpha \in [0, 1)$

Q-Learning



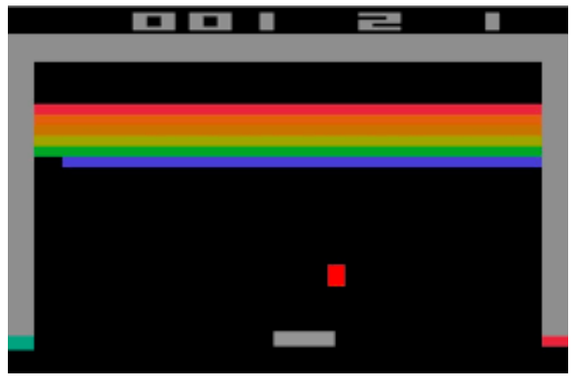
→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 100

$Q(s,a)$	A_1	A_2	A_3
S_1	0.6	1.3	0.6
S_2	1.6	0.8	0.9
S_3	0.8	1.7	0.5

Q-Learning



→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 100

$Q(s,a)$	A_1	A_2	A_3
S_1	0.6	1.3	0.6
S_2	1.6	0.8	0.9
S_3	0.8	1.7	0.5

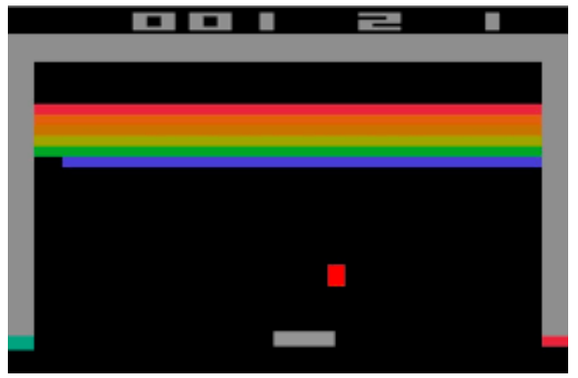
$$A = \text{argmax}_a Q(S, a)$$

$$\pi(S_1) = A_2$$

$$\pi(S_2) = A_1$$

$$\pi(S_3) = A_2$$

Q-Learning



→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 100

$Q(s,a)$	A_1	A_2	A_3
S_1	0.6	1.3	0.6
S_2	1.6	0.8	0.9
S_3	0.8	1.7	0.5

$$A = \text{argmax}_a Q(S, a)$$

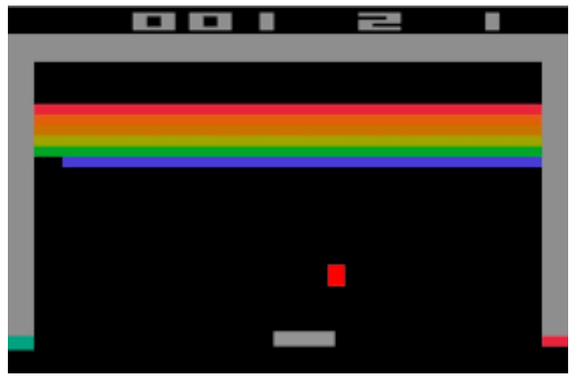
$$\pi(S_1) = A_2$$

$$\pi(S_2) = A_1$$

$$\pi(S_3) = A_2$$

$$\delta_t = Q(S_t, A_t) + R_{t+1} - \max_a Q(S_{t+1}, a) = 0$$

Q-Learning



→ improve policy using current Q-function

- take action which maximizes the Q-value at each step

iteration 100

converged

$Q(s,a)$	A_1	A_2	A_3
S_1	0.6	1.3	0.6
S_2	1.6	0.8	0.9
S_3	0.8	1.7	0.5

$$A = \text{argmax}_a Q(S, a)$$

$$\pi(S_1) = A_2$$

$$\pi(S_2) = A_1$$

$$\pi(S_3) = A_2$$

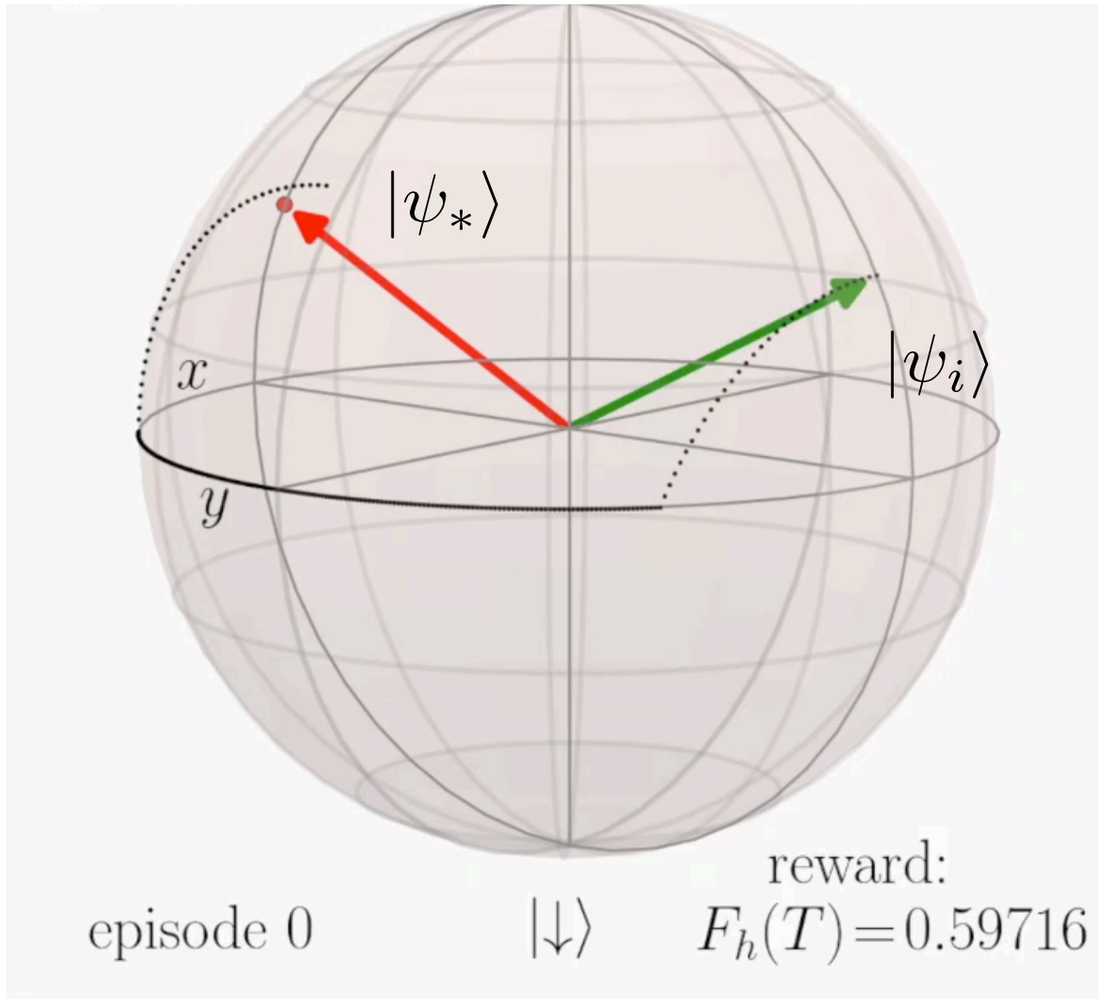
$$\delta_t = Q(S_t, A_t) + R_{t+1} - \max_a Q(S_{t+1}, a) = 0$$

RL Applied to Quantum State Preparation

$$H(t) = -S^z - h_x(t) S^x$$

$$S^x = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$S^z = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$



state preparation = fidelity optimization

$$F_h(t) = |\langle\psi_*|\psi(t)\rangle|^2$$

$$|\psi(t)\rangle = \mathcal{T} \mathrm{e}^{-i\int_0^t \mathrm{d}t' H(t')} |\psi_i\rangle$$

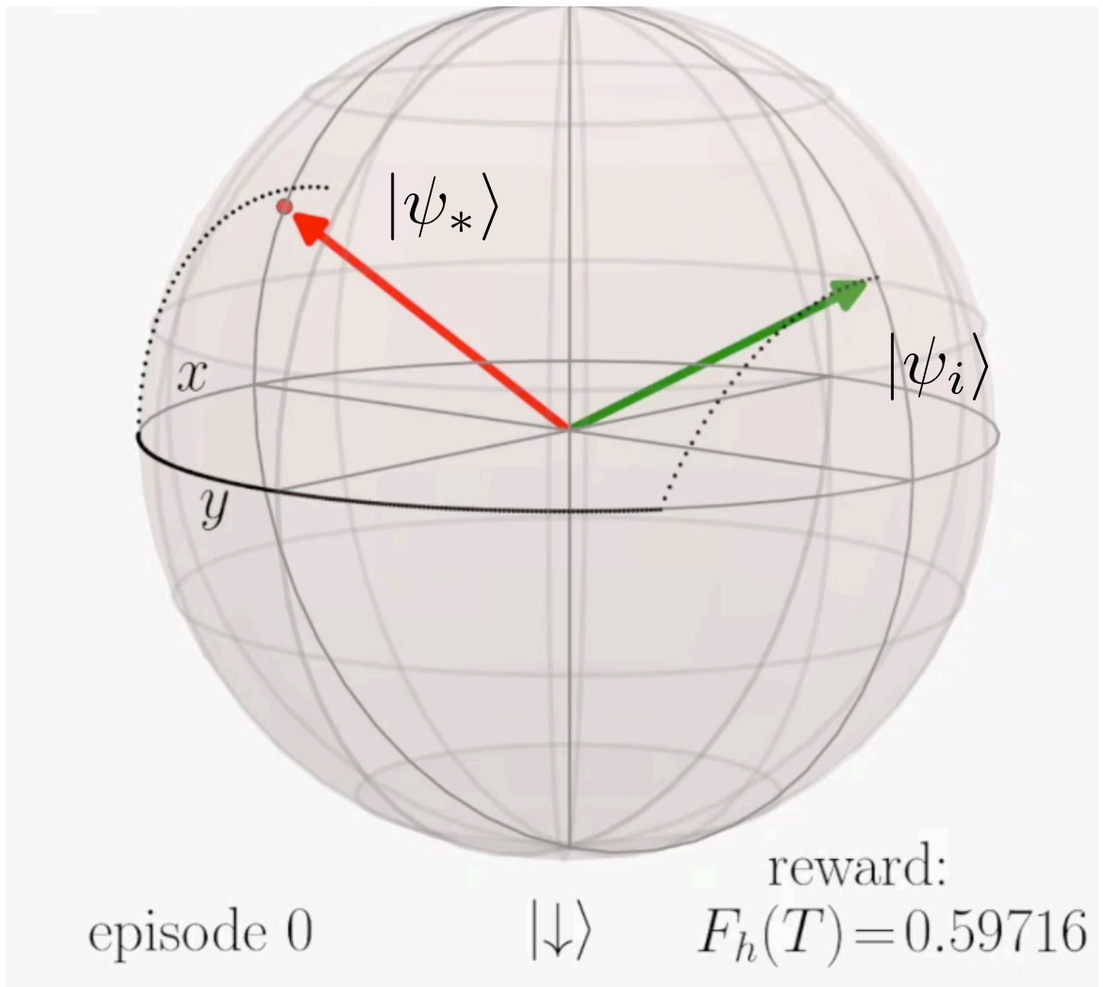
RL Applied to Quantum State Preparation

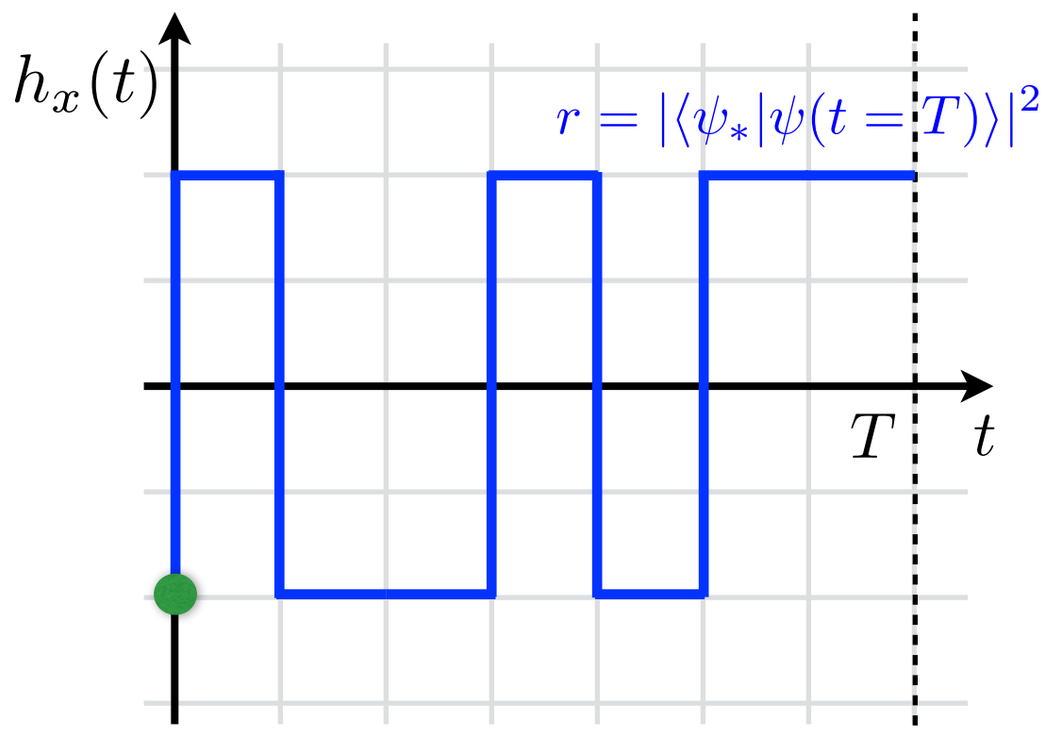
$$H(t) = -S^z - h_x(t) S^x$$

$$S^x = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad S^z = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$F_h(t) = |\langle \psi_* | \psi(t) \rangle|^2$$

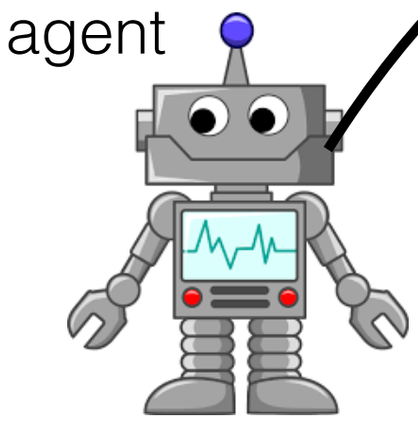
task: find optimal field within family of bang-bang protocols $h \in \{\pm 4\}$





episode completed

RL Applied to Quantum State Preparation



1

feedback loop,
updates RL agent

3

ENVIRONMENT

2

$$H(t) = H_0 + H_{\text{ctrl}}(t)$$

$|\psi_i\rangle$: GS of H_0

$|\psi_*\rangle$: target state

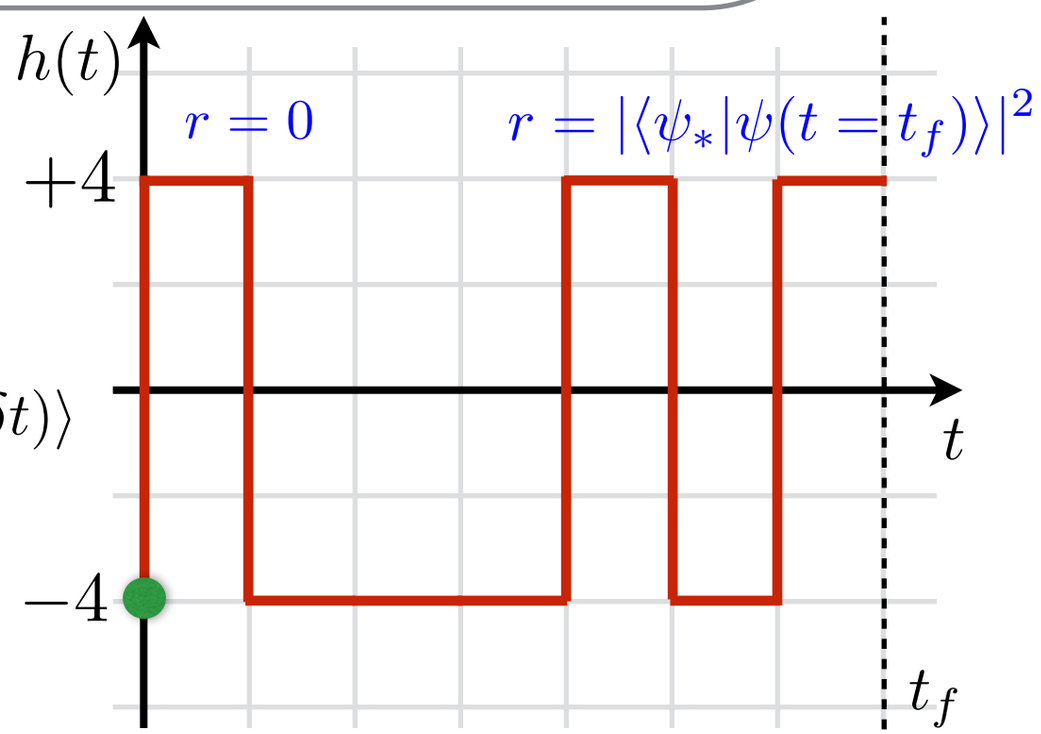
$$i\partial_t|\psi(t)\rangle = H(t)|\psi(t)\rangle \; t \in [0, t_f]$$

reward $r = \begin{cases} 0 & , 0 \le t < t_f \\ |\langle\psi_*|\psi(t=t_f)\rangle|^2 & , t = t_f \end{cases}$

1 start from state $s_0 = [h(0)] = [-4]$
take action $a_0 : \delta h = +8$
go to state $s_1 = [h(0), h(\delta t)] = [-4, +4]$

2 solve Schrödinger Eq. and obtain the QM state $|\psi(\delta t)\rangle$

3 calculate reward r
and use it to update $Q(s, a)$
which in turn is used to choose subsequent actions



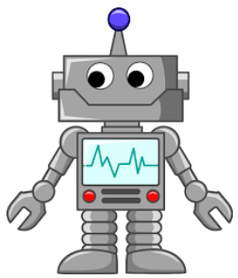
episode completed

MB et al, PRX 8 031086 (2018)

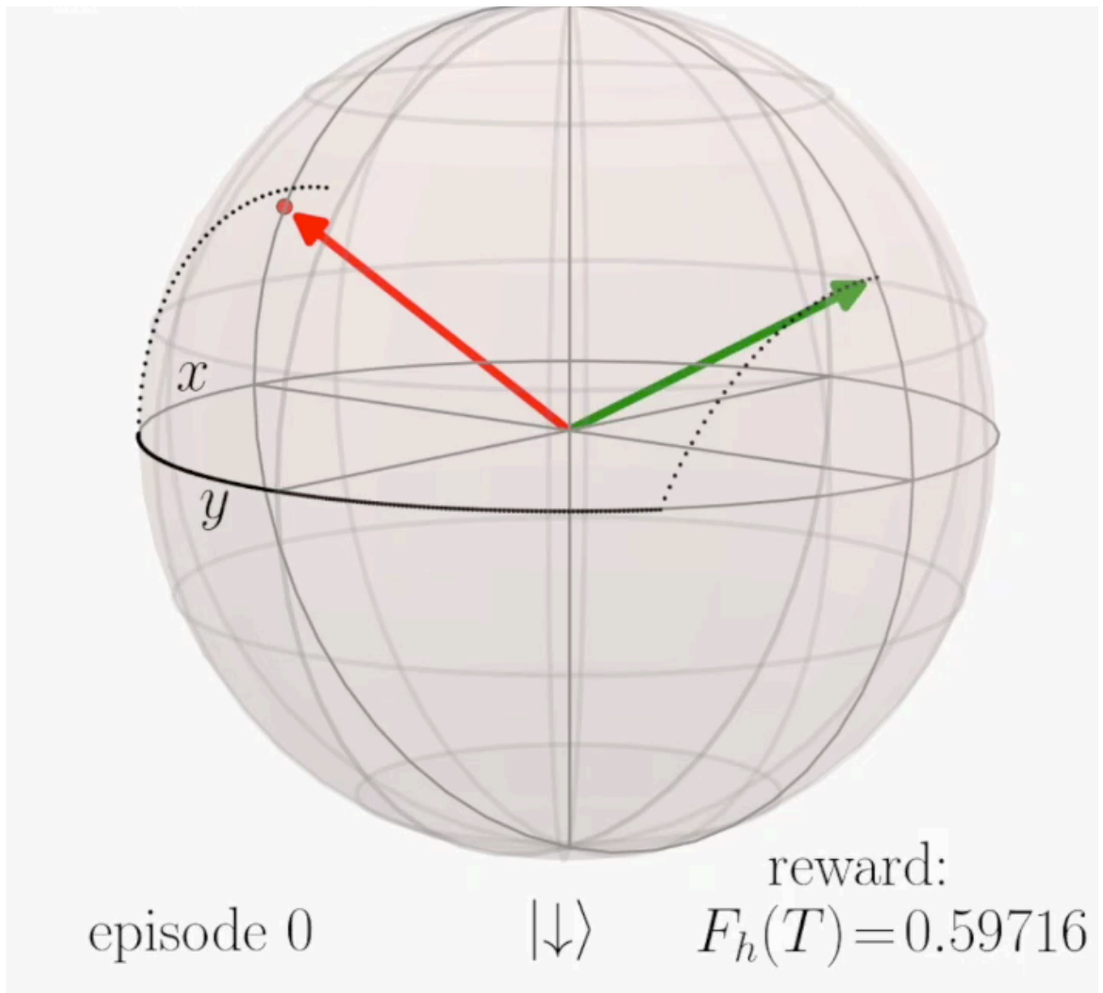
RL Applied to Quantum State Preparation

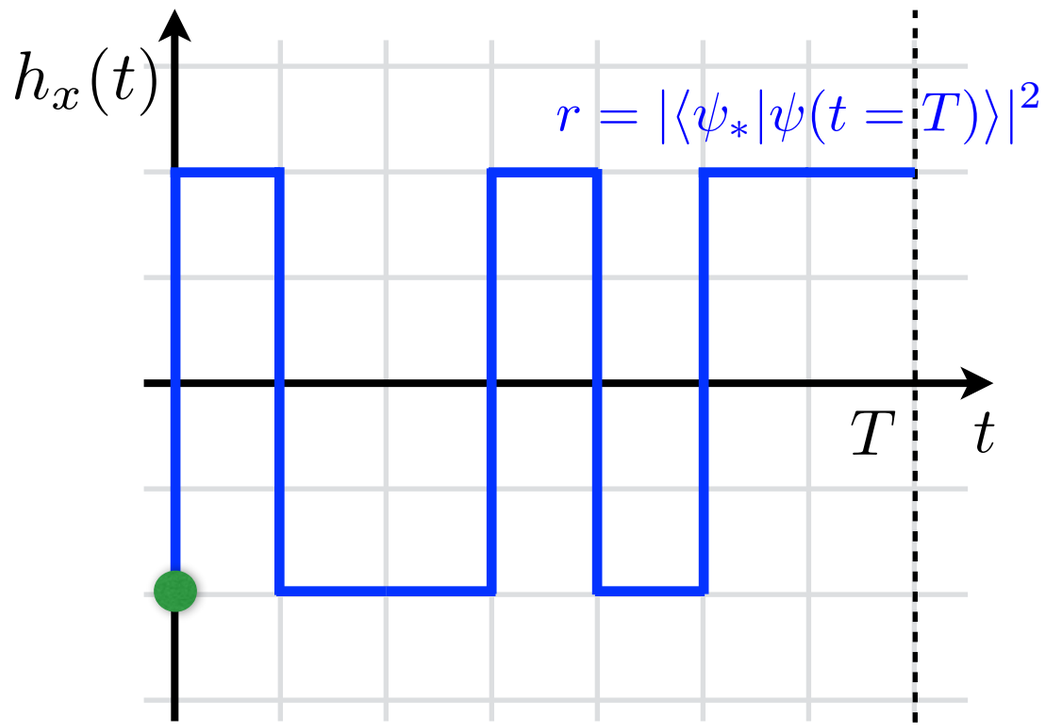
$$H(t) = -S^z - h_x(t) S^x$$

$h \in \{\pm 4\}$ bang-bang protocols



movie of the learning process

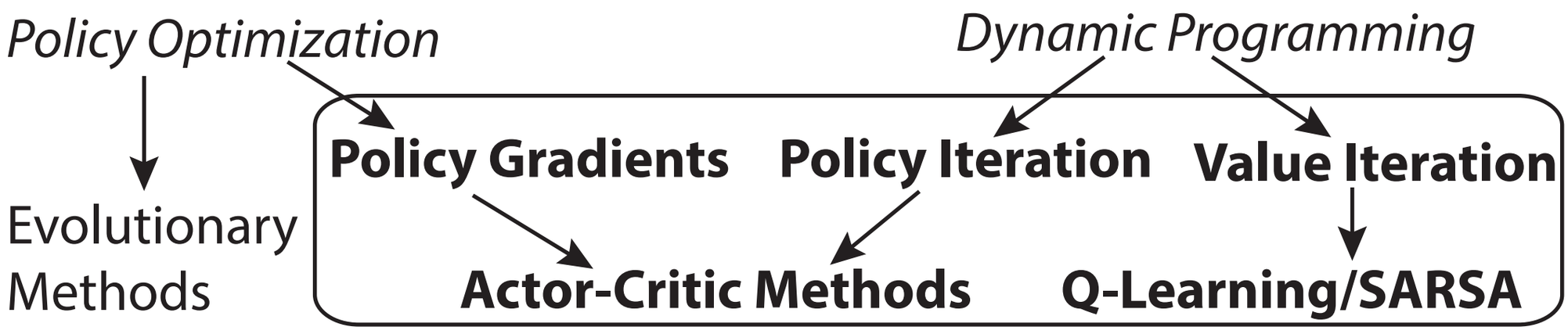




episode completed

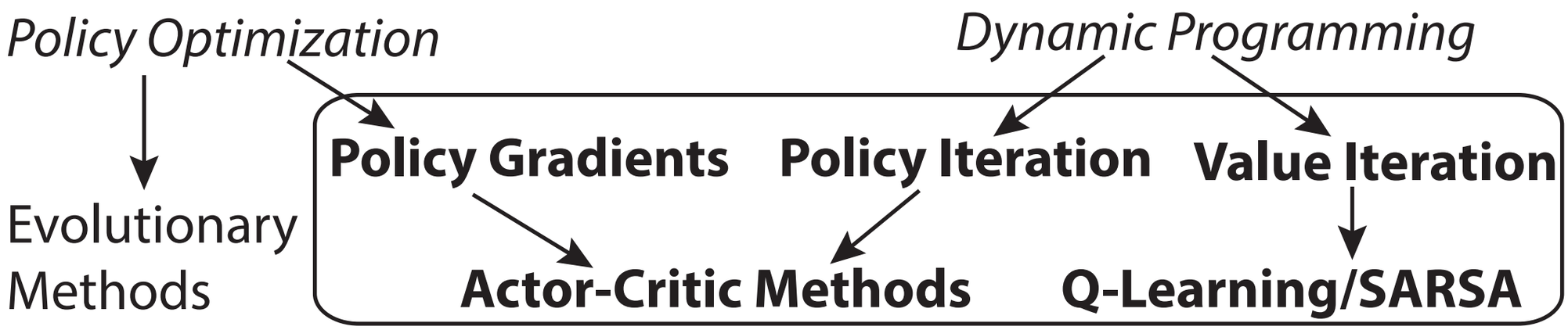
Overview of RL Algorithms

→ overview of RL algorithms

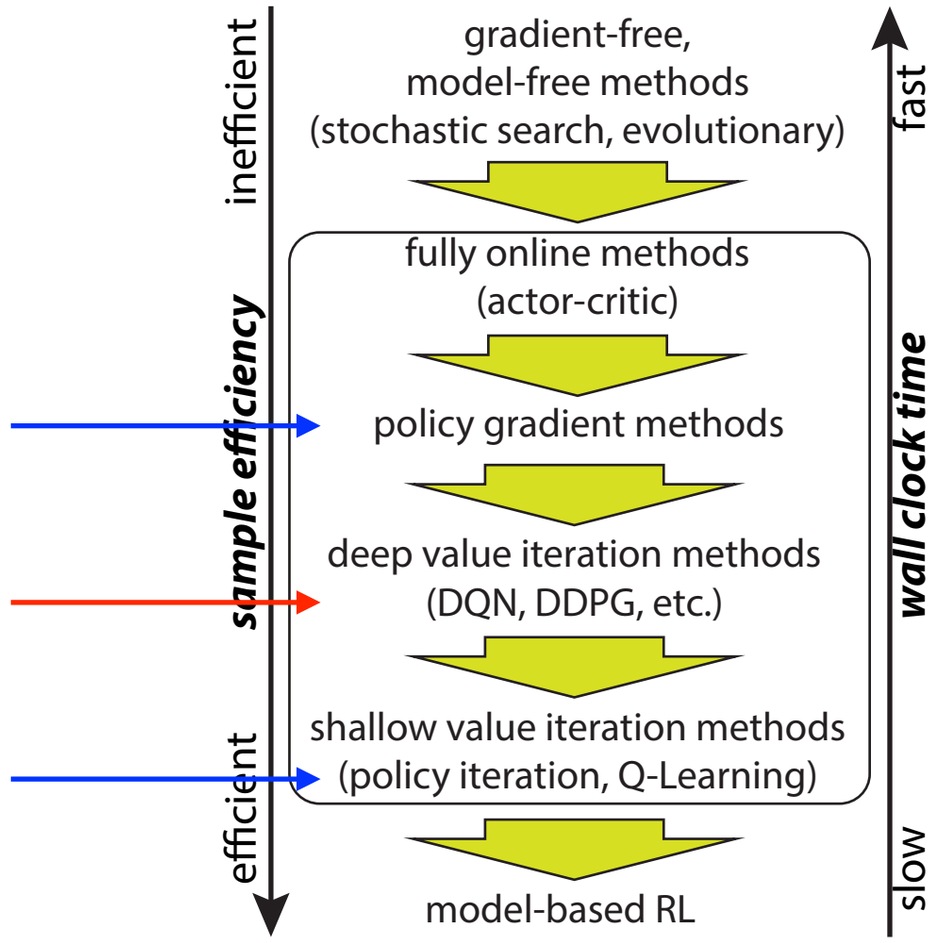


Overview of RL Algorithms

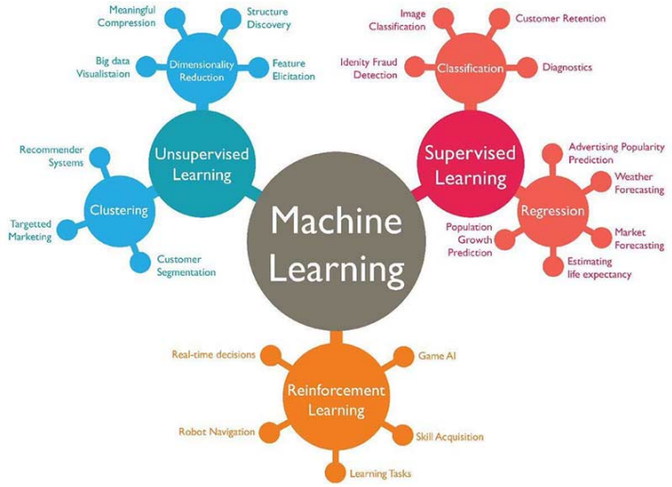
→ overview of RL algorithms



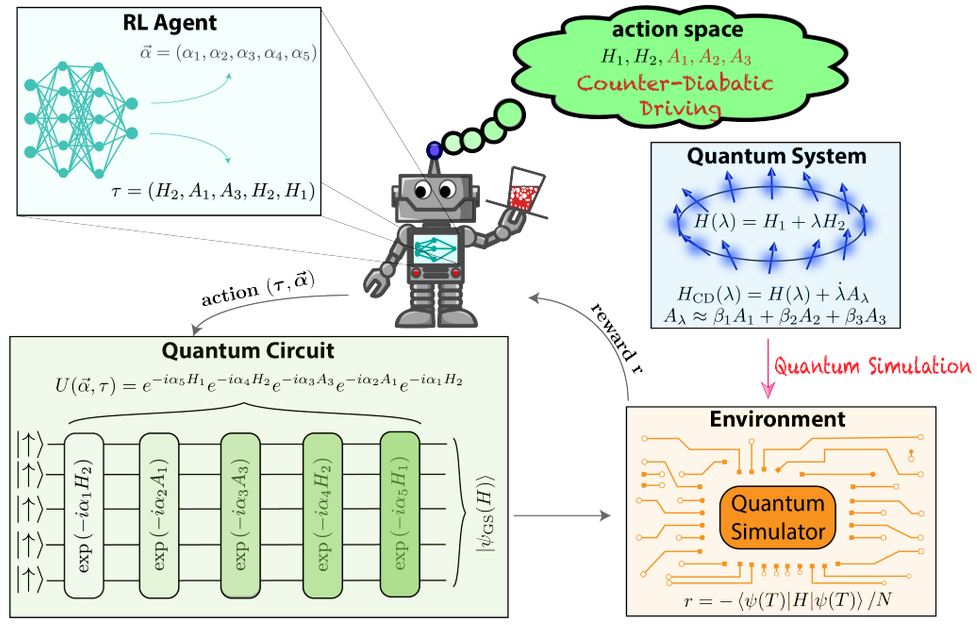
→ which algorithm to use?



Overview

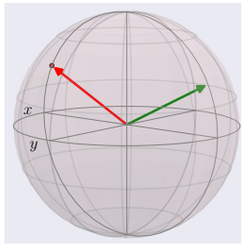


- What is Reinforcement Learning (RL)?
- RL framework in a nutshell
- RL algorithms
- Applications of RL in quantum physics



Applications of RL in Quantum Physics

→ **quantum control**

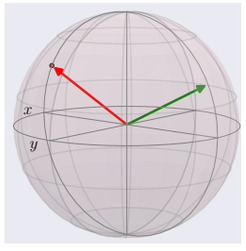


MB et al, PRX 8 031086 (2018)
Niu et al, npj 5 33 (2019)
Sivak et al, PRX 12, 011059 (2022)
Zhang, npj 9 85 (2019)
Yao et al, PRX 11 (3), 031070 (2021)
Porotti, Comm Phys 2 (2019)

+ many more

Applications of RL in Quantum Physics

→ **quantum control**

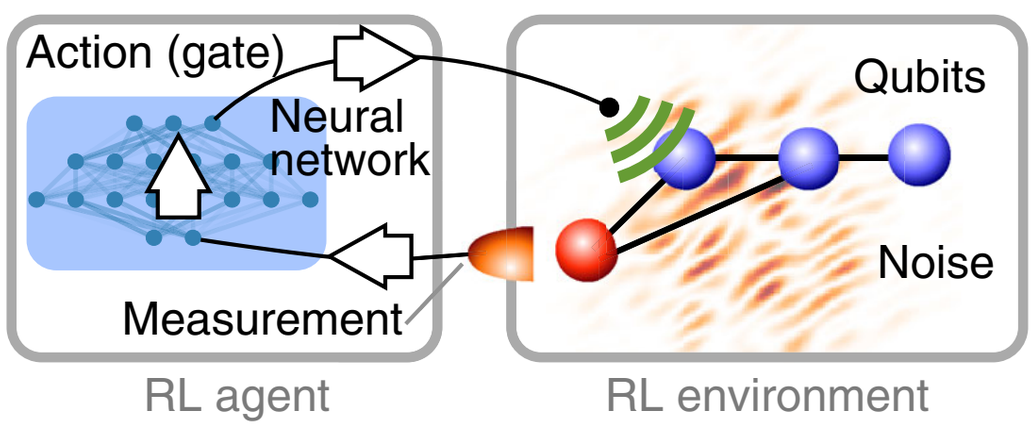


MB et al, PRX 8 031086 (2018)
Niu et al, npj 5 33 (2019)
Sivak et al, PRX 12, 011059 (2022)
Zhang, npj 9 85 (2019)
Yao et al, PRX 11 (3), 031070 (2021)
Porotti, Comm Phys 2 (2019)

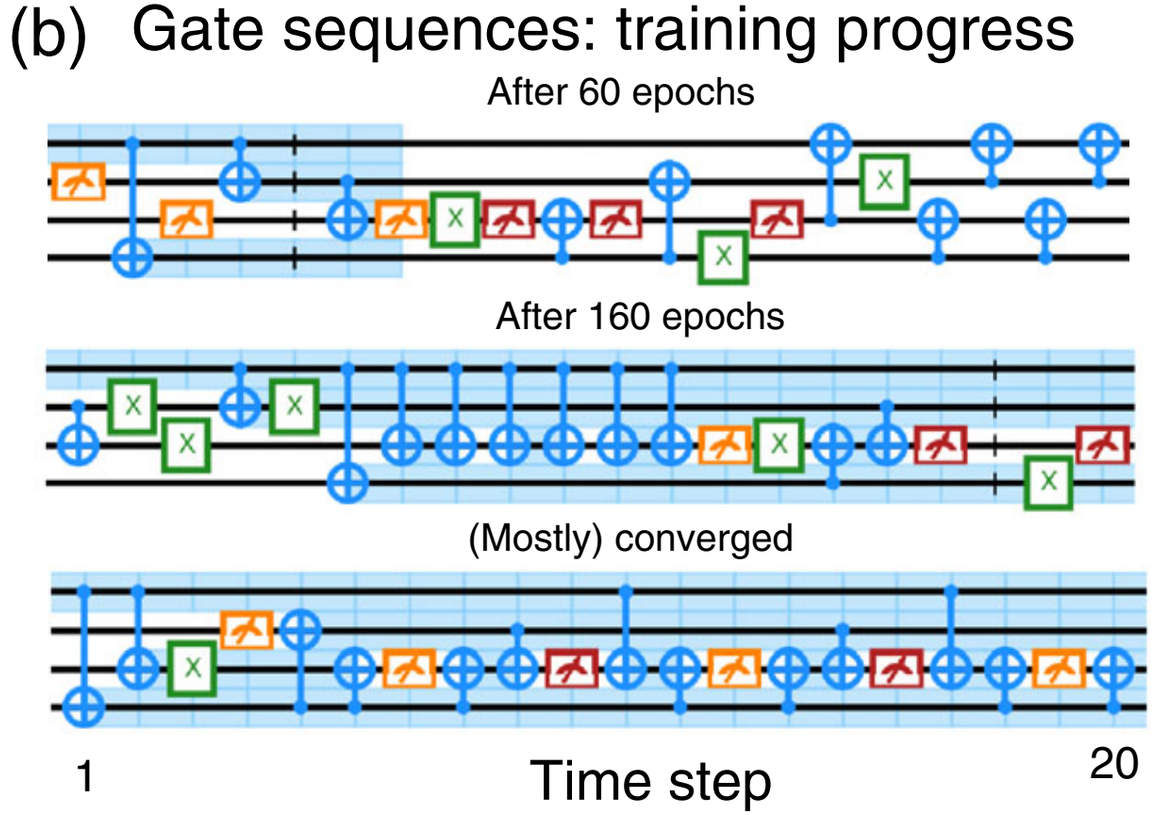
+ many more

→ **quantum error correction**

- **task:** find error correcting code that protects qubits from decoherence

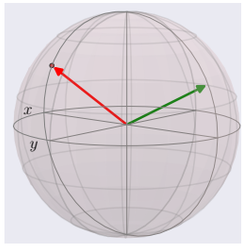


Fössel et al, PRX 8 031086 (2018)



Applications of RL in Quantum Physics

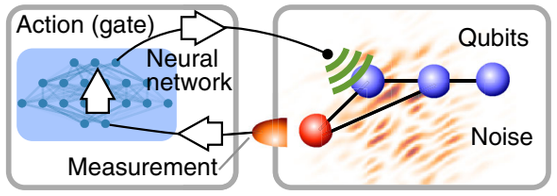
→ quantum control



MB et al, PRX 8 031086 (2018)
Niu et al, npj 5 33 (2019)
Sivak et al, PRX 12, 011059 (2022)
Zhang, npj 9 85 (2019)
Yao et al, PRX 11 (3), 031070 (2021)
Porotti, Comm Phys 2 (2019)

+ many more

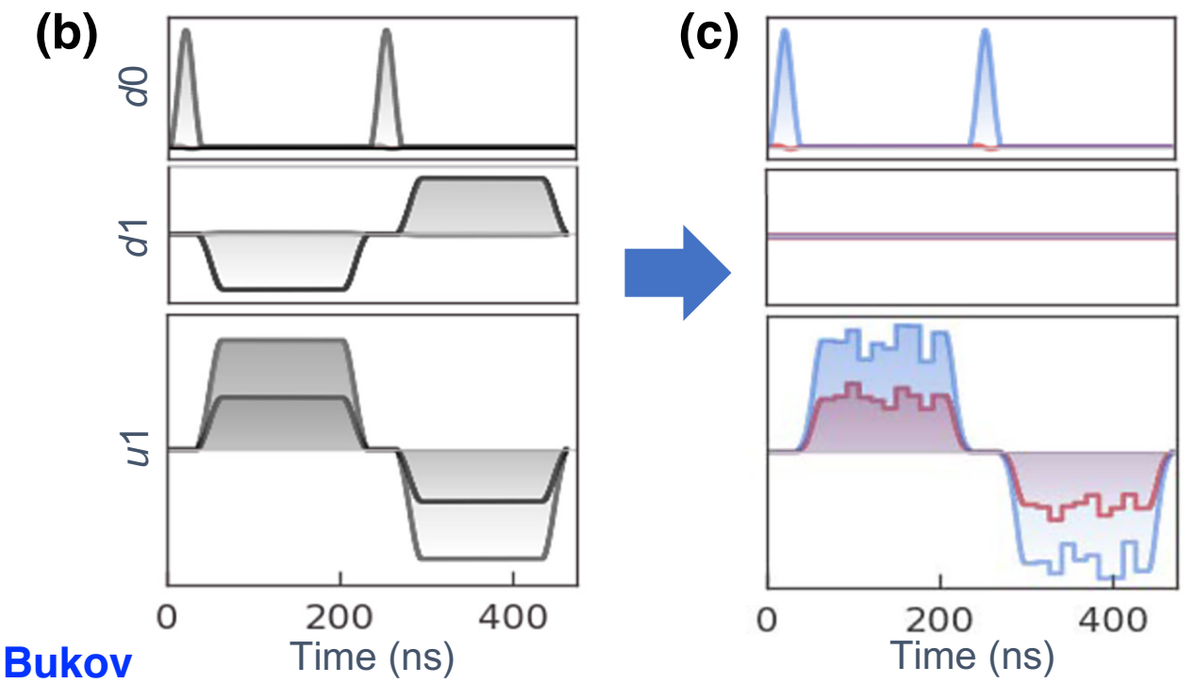
→ quantum error correction



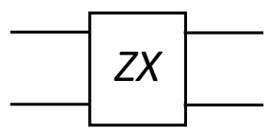
Fössel et al, PRX 8 031086 (2018)
Andreasson et al, Quantum 3 183 (2019)
Sweke et al, ML Sci Tech 2 025005 (2020)

→ quantum gates design

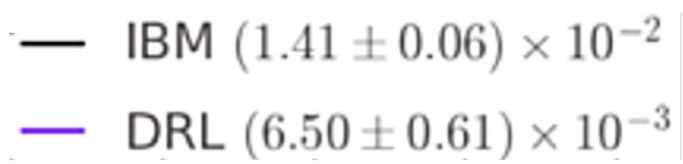
- **task:** find high-fidelity pulses that emulate gates on a quantum computer



Baum et al, PRX Quantum 2, 040324 (2021)



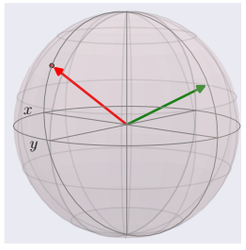
cross resonance gate



infidelity / error rate

Applications of RL in Quantum Physics

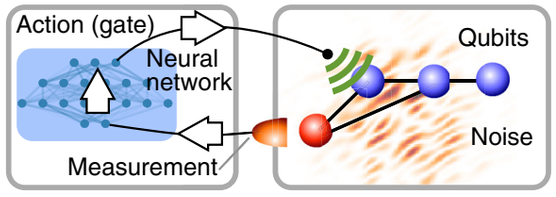
→ quantum control



MB et al, PRX 8 031086 (2018)
Niu et al, npj 5 33 (2019)
Sivak et al, PRX 12, 011059 (2022)
Zhang, npj 9 85 (2019)
Yao et al, PRX 11 (3), 031070 (2021)
Porotti, Comm Phys 2 (2019)

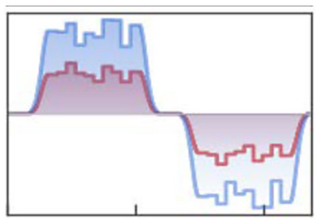
+ many more

→ quantum error correction



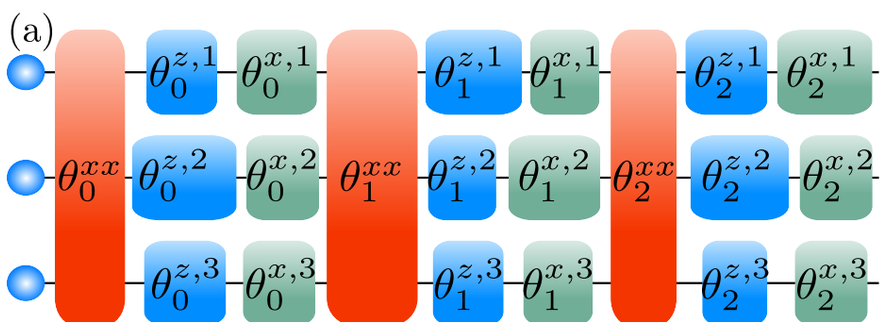
Fössel et al, PRX 8 031086 (2018)
Andreasson et al, Quantum 3 183 (2019)
Sweke et al, ML Sci Tech 2 025005 (2020)

→ quantum gates design

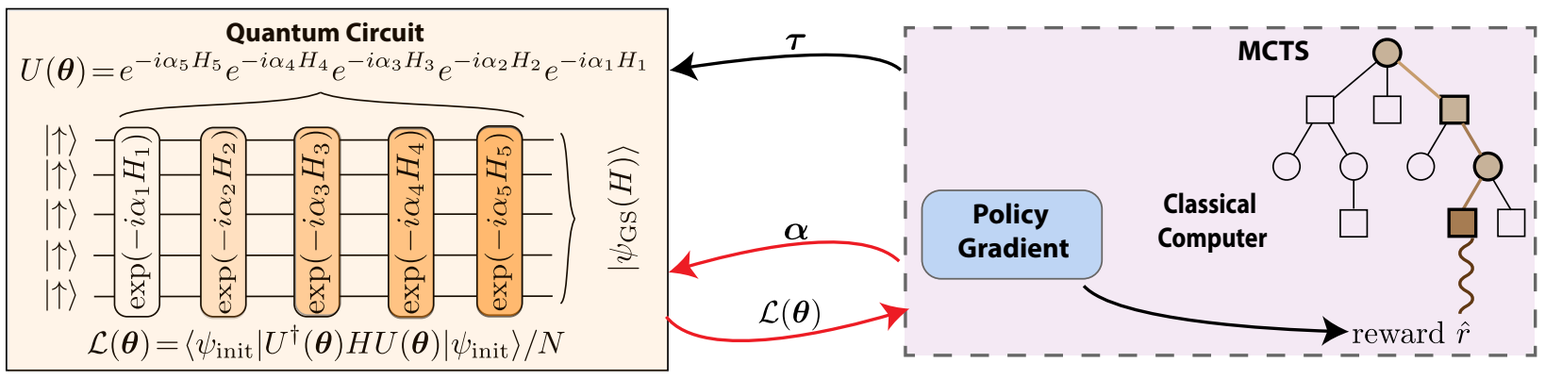


Baum et al, PRX Quantum 2, 040324 (2021)

→ variational quantum circuit design

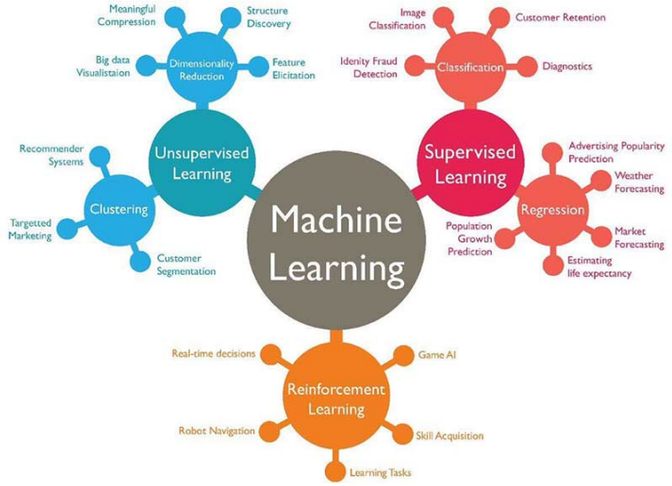


Bolens et al, PRL 2021

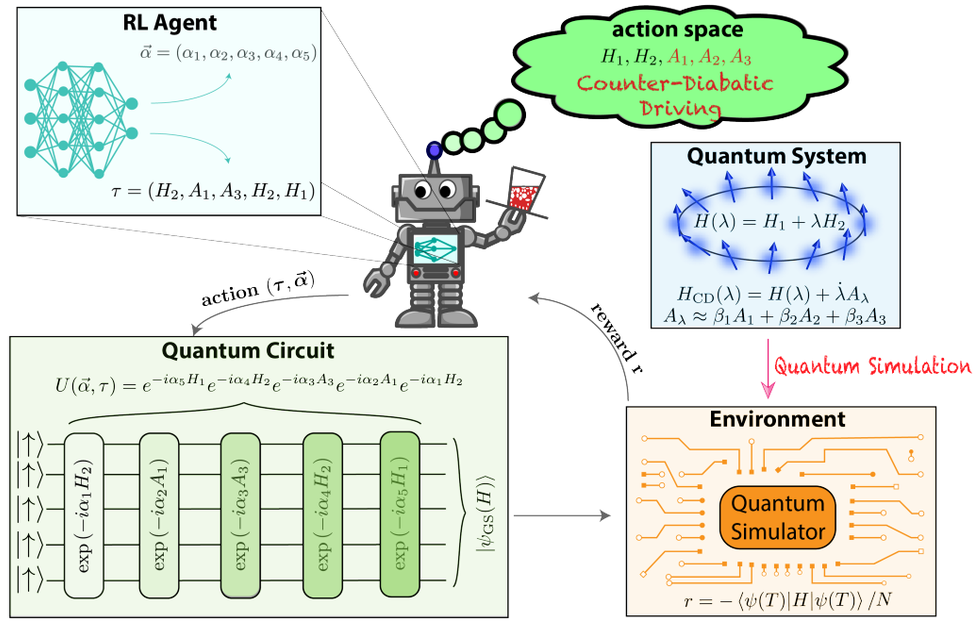


Yao et al, MSML 2019, 2021, 2022

Overview

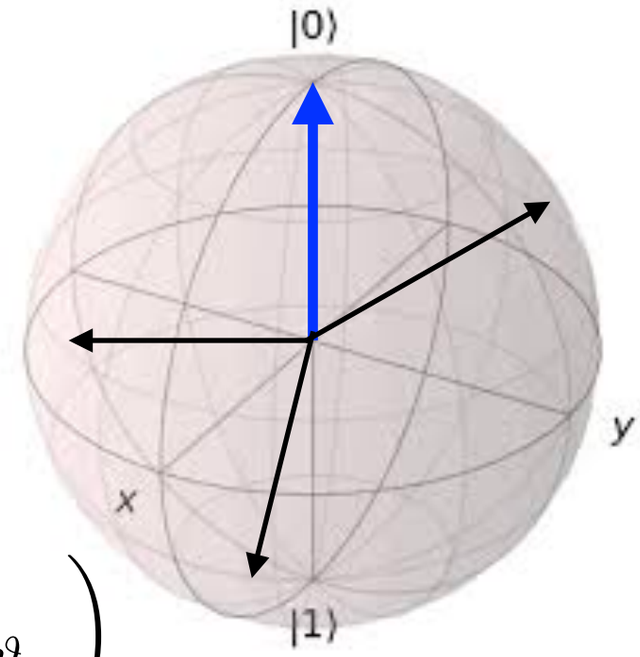


- What is Reinforcement Learning (RL)?
- RL framework in a nutshell
- RL algorithms
- Applications of RL in quantum physics
- Deep RL



RL for Quantum State Initialization

Bloch sphere



→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

- define **RL states**:

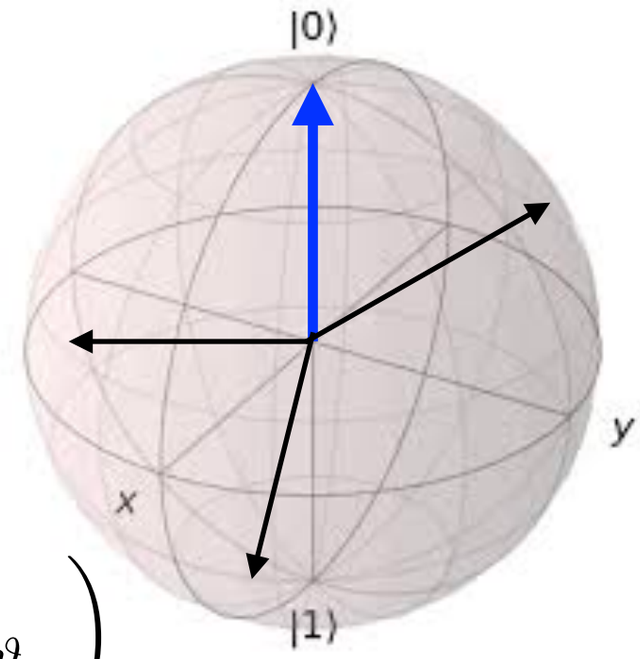
parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

RL state space: $\mathcal{S} = \{(\vartheta, \varphi)|\vartheta \in [0, \pi],\ \varphi \in [0, 2\pi)\}$

- define **RL actions**: $\mathcal{A} = \{I, U_\alpha|\alpha = x, y, z\}$ $\qquad |\psi'\rangle = U_\alpha|\psi\rangle$ $\qquad s \mapsto s'$
- define reward: $r_t = |\langle 0|\psi_t\rangle|^2$

RL for Quantum State Initialization

Bloch sphere



→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

- define **RL states**:

parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

RL state space: $\mathcal{S} = \{(\vartheta, \varphi)|\vartheta \in [0, \pi],\ \varphi \in [0, 2\pi)\}$

- define **RL actions**: $\mathcal{A} = \{I, U_\alpha|\alpha = x, y, z\}$ $\qquad |\psi'\rangle = U_\alpha|\psi\rangle$ $\qquad s \mapsto s'$
- define reward: $r_t = |\langle 0|\psi_t\rangle|^2$

problem: *continuous* state space has infinitely many configurations

RL with Function Approximation

→ **problem:** state space has exponentially many configurations $|\mathcal{A}|^{N_T}$

- can we estimate values of not yet encountered states?

RL with Function Approximation

→ **problem:** state space has exponentially many configurations $|\mathcal{A}|^{N_T}$

- can we estimate values of not yet encountered states?

→ YES, via inter- & extrapolation: parametrize the Q-function/policy

$$Q(s,a) \to Q_\theta(s,a) \qquad \pi(a|s) \to \pi_\theta(a|s)$$

variational parameters θ

RL with Function Approximation

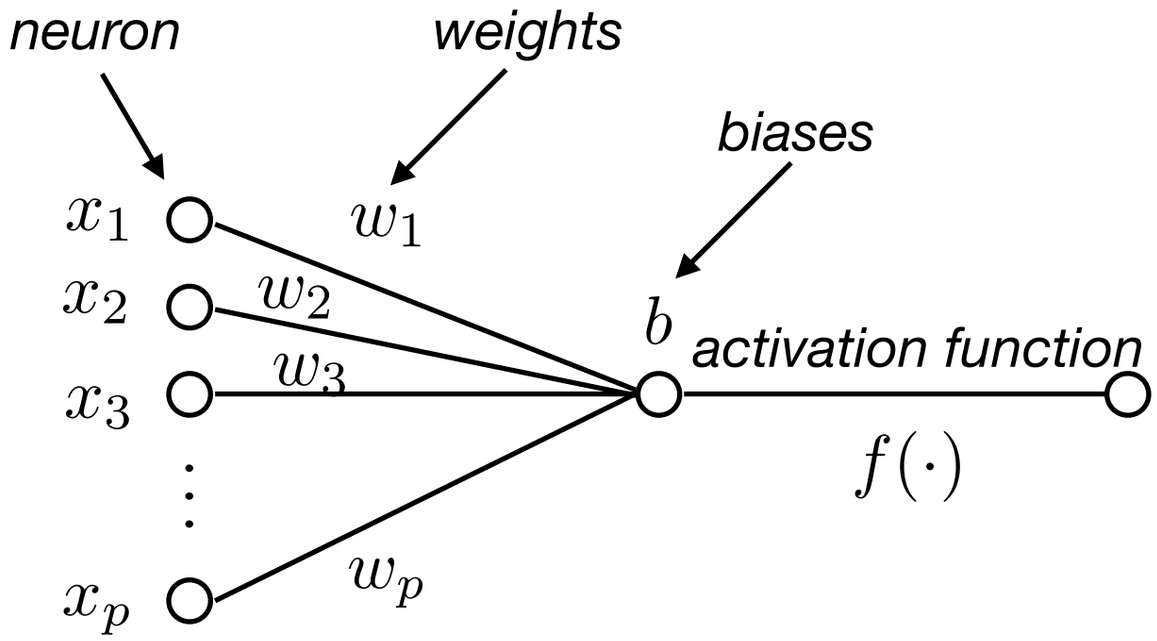
→ **problem:** state space has exponentially many configurations $|\mathcal{A}|^{N_T}$

- can we estimate values of not yet encountered states?

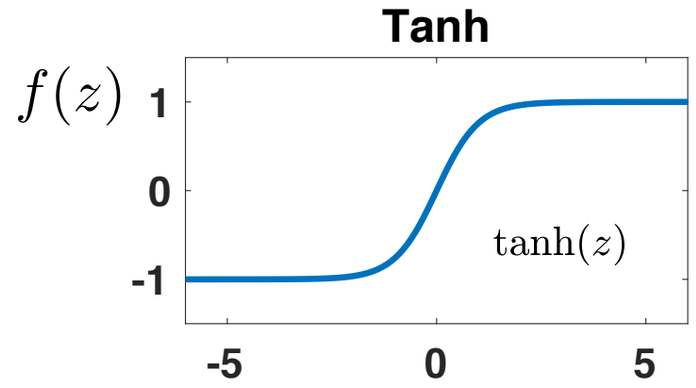
→ YES, via inter- & extrapolation: parametrize the Q-function/policy

$$Q(s,a) \to Q_\theta(s,a) \qquad \pi(a|s) \to \pi_\theta(a|s)$$

- typical approach: use deep neural network **(Deep RL)**



$$y = f(W \cdot x + b)$$



$$W = (w_1, w_2, \ldots, w_p) \qquad \theta = \{W, b\}$$

RL with Function Approximation

→ **problem:** state space has exponentially many configurations $|\mathcal{A}|^{N_T}$

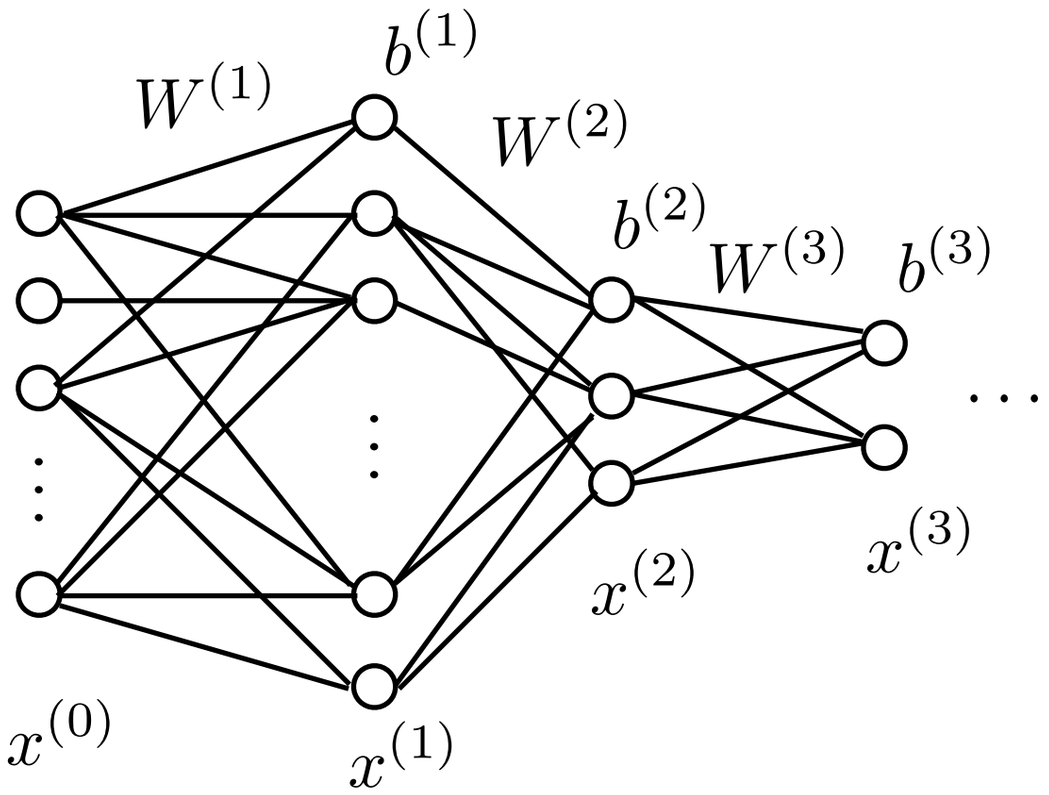
- can we estimate values of not yet encountered states?

→ YES, via inter- & extrapolation: parametrize the Q-function/policy

$$Q(s,a) \to Q_\theta(s,a) \qquad \pi(a|s) \to \pi_\theta(a|s)$$

$$\theta = \{W, b\}$$

- typical approach: use deep neural network **(Deep RL)**



$$x^{(0)} \quad \text{input layer}$$

$$x_i^{(1)} = f^{(1)}\left(W_{ij}^{(1)} x_j^{(0)} + b_i^{(1)}\right)$$

$$x_i^{(2)} = f^{(2)}\left(W_{ij}^{(2)} x_j^{(1)} + b_i^{(2)}\right)$$

$b_i^{(l)}$: bias vector of layer l

$W_{ij}^{(l)}$: weight matrix of layer l

$f^{(l)}$: activation function of layer l

RL for Quantum State Initialization

→ **quantum control:** two-level system

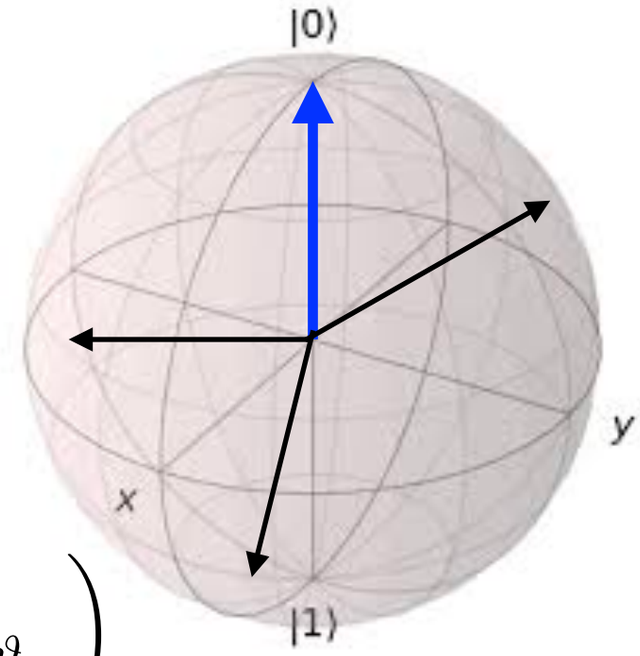
- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

- define **RL states**:

parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

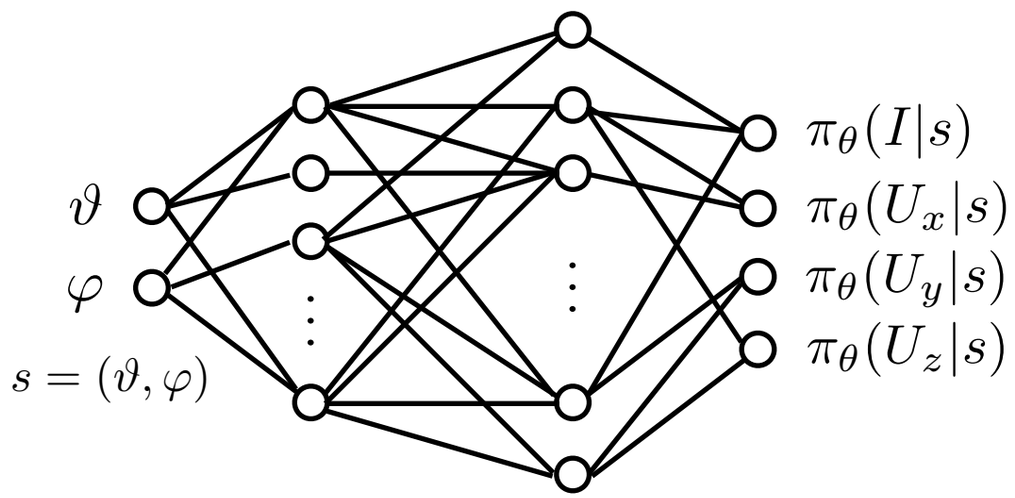
Bloch sphere



RL state space: $\mathcal{S} = \{(\vartheta, \varphi)|\vartheta \in [0, \pi],\ \varphi \in [0, 2\pi)\}$

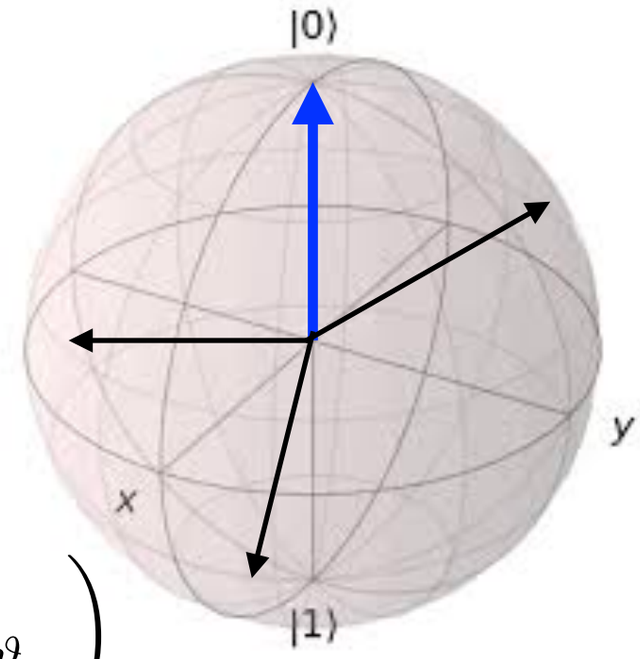
- define **RL actions**: $\mathcal{A} = \{I, U_\alpha|\alpha = x, y, z\}$
- define reward: $r_t = |\langle 0|\psi_t\rangle|^2$
- choose **RL algorithm:** policy gradient

$$\pi_\theta(a|s)$$



RL for Quantum State Initialization

Bloch sphere



→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

- define **RL states**:

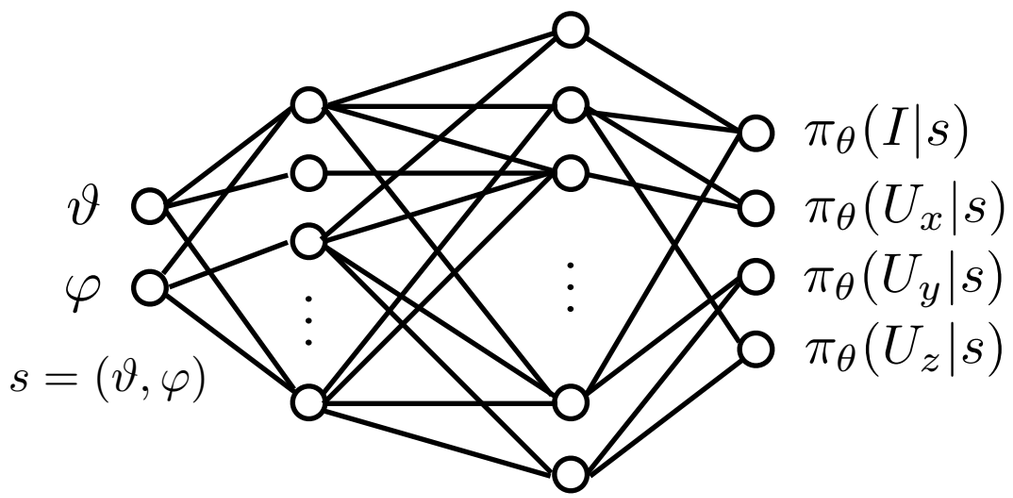
parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

RL state space: $\mathcal{S} = \{(\vartheta, \varphi)|\vartheta \in [0, \pi],\ \varphi \in [0, 2\pi)\}$

- define **RL actions**: $\mathcal{A} = \{I, U_\alpha|\alpha = x, y, z\}$
- define reward: $r_t = |\langle 0|\psi_t\rangle|^2$
- choose **RL algorithm:** policy gradient

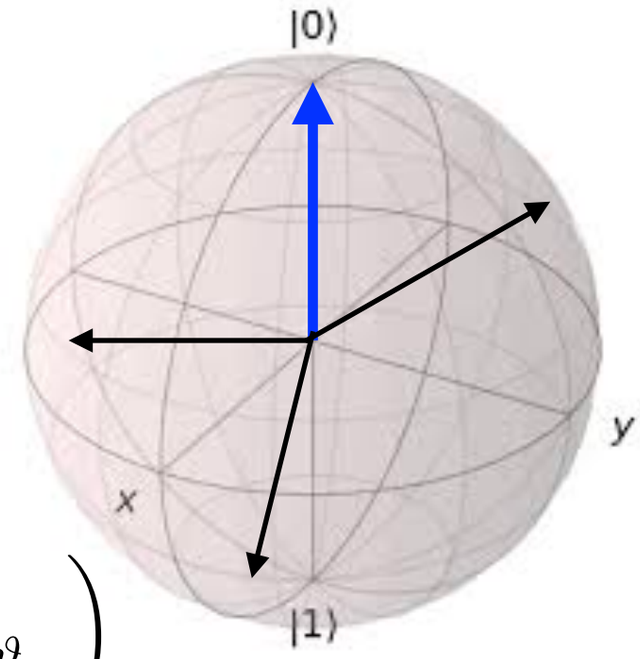
simulate trial trajectories τ_j

$$\nabla_\theta J(\theta) \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}} \nabla_\theta \log \pi_\theta(\tau_j) \sum_t R_t(\tau_j)$$



RL for Quantum State Initialization

Bloch sphere



→ **quantum control:** two-level system

- **task:** prepare $|0\rangle$ using infinitesimal rotations

$$U_\alpha = \exp\left(-i\frac{\delta t}{2}\sigma^\alpha\right) \qquad \alpha = x, y, z$$

- define **RL states**:

parametrization of state of system: $|\psi\rangle = \begin{pmatrix} \cos\frac{\vartheta}{2} \\ e^{i\varphi}\sin\frac{\vartheta}{2} \end{pmatrix}$

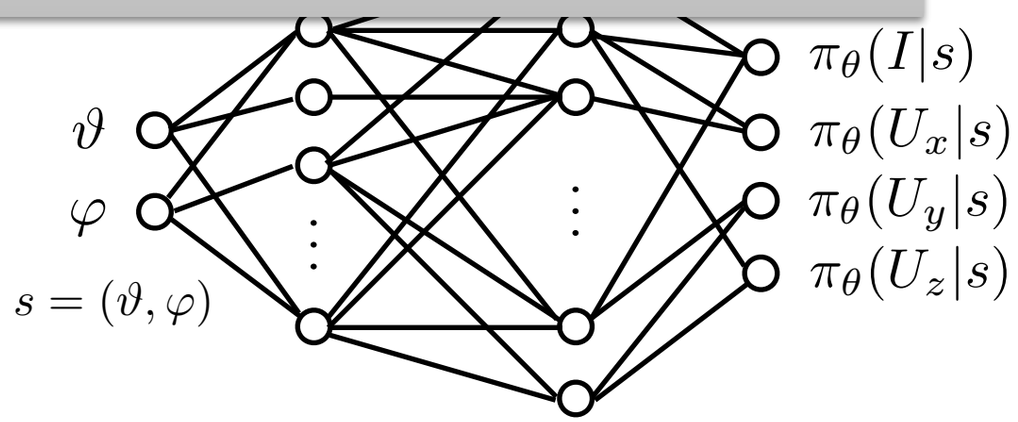
Check out Jupyter notebook for how this works in practice!

https://github.com/mgbukov/quant22_notebooks

- choose **RL algorithm:** policy gradient

simulate trial trajectories τ_j

$$\nabla_\theta J(\theta) \approx \frac{1}{N_{\mathrm{MC}}}\sum_{j=1}^{N_{\mathrm{MC}}} \nabla_\theta \log \pi_\theta(\tau_j) \sum_t R_t(\tau_j)$$



Thanks for your attention!

→ Useful Literature

M. Nielsen, *Neural Networks and Deep Learning* (online book)

Sutton and Barto, *Reinforcement Learning: an Introduction*, MIT press

S. Levine, *You Tube*, UC Berkeley (videos of lecture course)

M. Bukov, *lecture course,* Sofia University

http://quantum-dynamics.phys.uni-sofia.bg/teaching/WiSe_2020_RL_class/

€€€
European Commission
Horizon 2020
European Union funding
for Research & Innovation

Check out Jupyter notebook for how this works in practice!

https://github.com/mgbukov/quant22_notebooks

Marin Bukov

web: *www.pks.mpg.de/nqd*