

Reinforcement Learning for Multi-Agent Systems

Robert Babuška and Lucian Buşoniu

*Delft Center for Systems and Control
Faculty of Mechanical Engineering
Delft University of Technology
The Netherlands*

e-mail: r.babuska@dcsc.tudelft.nl
http://dcsc.tudelft.nl/~rbabuska
tel: 015-27 85117

Outline

1. **Introduction**
 - multi-agent systems
 - learning paradigms
2. **Single-agent reinforcement learning**
 - Markov decision process, learning goal, policy
 - Bellman equation, optimality, solutions
 - Q-learning, actor-critic scheme
3. **Multi-agent reinforcement learning**
 - stochastic game, learning goal
 - tasks in multi-agent reinforcement learning
 - methods and algorithms
4. **Concluding remarks**

Multi-Agent Systems

- **Agent**: an entity that can perceive its environment through sensors and act upon it through actuators (Russell & Norvig, 2003)
- **Rational agent**: an agent that always tries to optimize an appropriate performance measure
- **Multi-agent system**: a group of agents that coexist and potentially interact with one another
- Human , software , robotic agents, ...

Learning: Rather than a priori design agent behaviors that perform well under all conditions → learn / adapt behaviors online.

Learning Systems

Can find solutions

- that are hard to determine a priori
- that are hard to program
- improve over time
- possibly in a (slowly) time-varying process

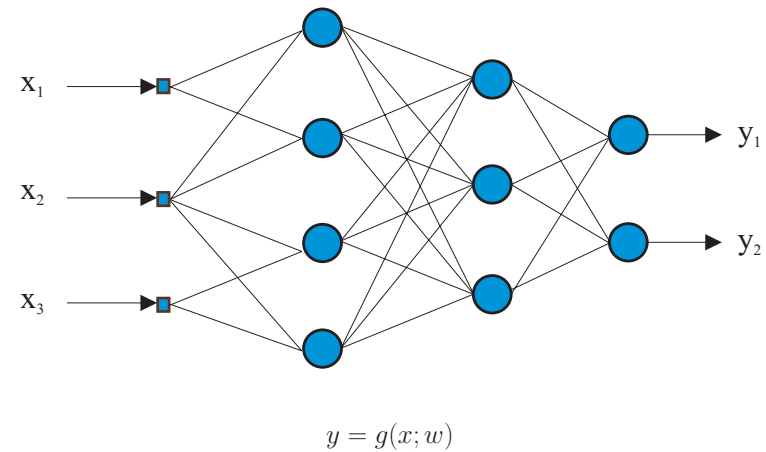
Learning – an essential feature of "intelligent" systems.

Learning Paradigms (1)

- **Supervised learning**

- Learning with teacher (from input–output sample pairs)
- Mainly used in modeling and identification
- Example: least-squares fitting, neural nets with back-propagation, operator cloning

Supervised Learning Example: Neural Network

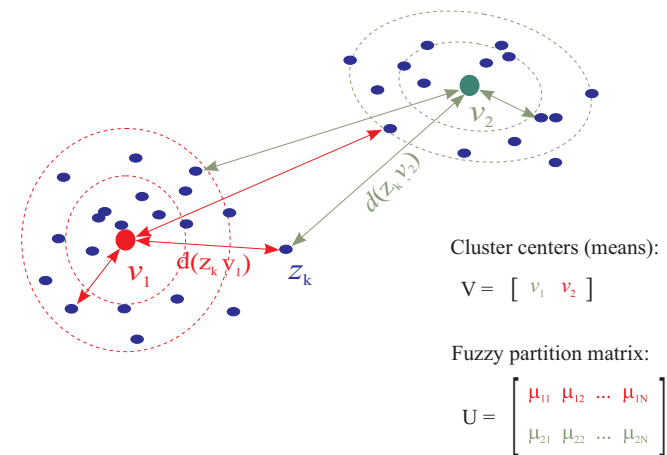


Learning Paradigms (2)

- **Unsupervised learning**

- Without teacher – desired output not used (not available)
- Discover structures in the data
- Example: clustering, self-organizing maps, classification

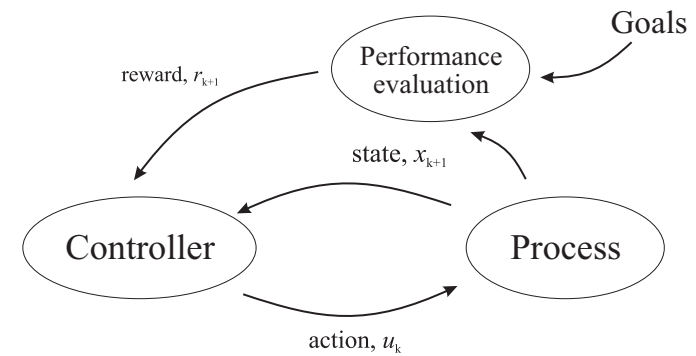
Unsupervised Learning Example: Clustering



Learning Paradigms (3)

- **Reinforcement learning**
 - Between supervised and unsupervised learning.
 - No input–output pairs: only inputs and reward.
 - Inspired by principles of human and animal learning.
 - Only mild assumptions on the process / environment.
 - A strategy can be learnt from scratch.

Reinforcement Learning



Basic Elements of Reinforcement Learning

- Model of the process – typically unknown.
- Reward function.
- Learning objective (cost function).
- Controller (policy, agent).
- Exploration.

Markov Decision Process

A **Markov decision process** is a tuple $\langle X, U, f, \rho \rangle$ where:

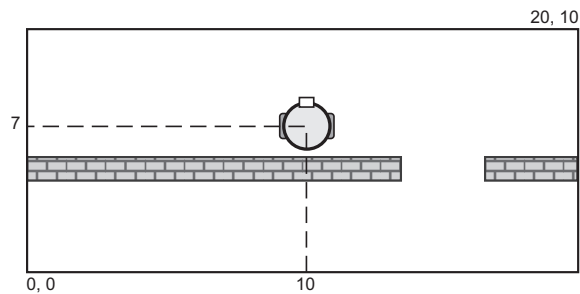
- X is the finite state space
- U is the finite action space
- $f : X \times U \rightarrow X$ is the state transition function
- $\rho : X \times U \rightarrow \mathbb{R}$ is the reward function

$$x_{k+1} = f(x_k, u_k) \quad (k - \text{discrete time step})$$

Note: stochastic formulation possible

Markov Decision Process - Example

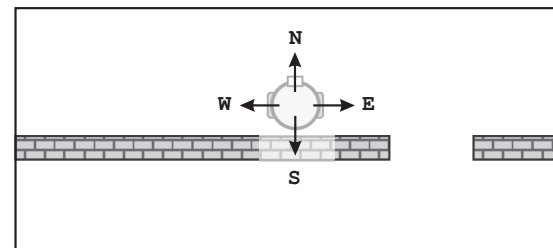
MDP ... $\langle X, U, f, \rho \rangle$



$$X = \{0, 1, \dots, 20\} \times \{0, 1, \dots, 10\}, \quad x_k = [10, 7]^T$$

Markov Decision Process - Example

MDP ... $\langle X, U, f, \rho \rangle$

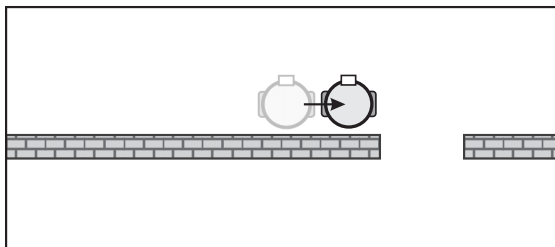


$$U = \{N, E, S, W\}$$

Markov Decision Process - Example

MDP ... $\langle X, U, f, \rho \rangle$

$$x_{k+1} = f(x_k, u_k)$$

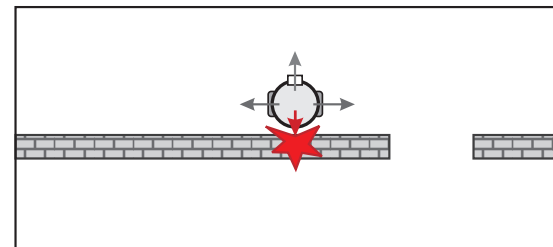


$$x_{k+1} = f([10, 7]^T, E) = [11, 7]^T$$

Markov Decision Process - Example

MDP ... $\langle X, U, f, \rho \rangle$

$$r_{k+1} = \rho(x_k, u_k)$$



$$\rho(x_k, W) = -1, \quad \rho(x_k, N) = -1, \quad \rho(x_k, E) = -1, \quad \rho(x_k, S) = -10$$

Learning Goal

Performance measure – **discounted return**:

$$R_k = \sum_{i=0}^{\infty} \gamma^i r_{k+i+1}$$

with:

$r_{k+1} = \rho(x_k, u_k)$... immediate reward

$\gamma \in (0, 1]$... discount factor (lookahead horizon)

finite trial vs. continuous learning

Learning goal: Find a control policy that maximizes the discounted return R_k at every time step k .

Learning the Control Policy

Policy defines what action should be taken in each state:

$$h : X \rightarrow U, \quad u = h(x)$$

There are various ways to learn the optimal policy.

Most straightforward: use the **Q-values**:

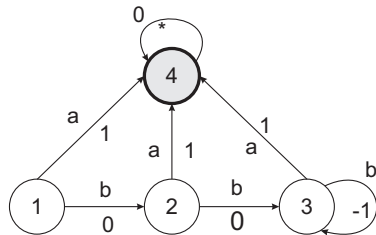
$$Q^h(x, u) = \sum_{i=0}^{\infty} \gamma^i r_{i+1}$$

where the agent starts in x , takes action u and then follows policy

$$h \quad (r_1 = \rho(x, u), \quad x' = f(x, u), \quad r_2 = \rho(x', h(x')), \dots)$$

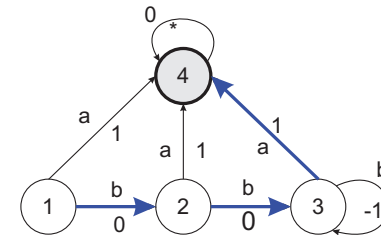
Greedy action: $u = \arg \max_{u' \in U} Q^h(x, u')$

Example: A Simple MDP



- $X = \{1, 2, 3, 4\}$; initial state: 1, goal state: 4
- $U = \{a, b\}$
- a moves to goal, reward 1; b in 1 and 2 moves right, no reward; b in 3 stays, punishment -1
- state 4 is absorbing, episode ends

Example: Policy and Q-Value



Policy: $h(1) = b, \quad h(2) = b, \quad h(3) = a, \quad \gamma = 0.9$

$$\begin{aligned} Q^h(1, b) &= \gamma^0 \rho(1, b) + \gamma^1 \cdot \rho(2, b) + \gamma^2 \cdot \rho(3, a) + \gamma^3 \cdot \rho(4, *) \\ &= 1 \cdot 0 + 0.9 \cdot 0 + 0.81 \cdot 1 + 0.729 \cdot 0 = 0.81 \end{aligned}$$

Recursive Computation of Q-values

$$\begin{aligned} Q^h(x, u) &= \sum_{i=0}^{\infty} \gamma^i r_{i+1} \\ &= r_1 + \gamma \sum_{i=0}^{\infty} \gamma^i r_{i+2} \\ &= \rho(x, u) + \gamma Q^h(x', u') \end{aligned}$$

$$Q^h(x, u) = \rho(x, u) + \gamma Q^h(x', u'), \quad x' = f(x, u), \quad u' = h(x')$$

This equation is called the **Bellman equation**.

Optimality

- Optimal value function:

$$Q^*(x, u) = \max_h Q^h(x, u)$$

- Optimal policy – greedy policy in Q^* :

$$h^*(x) = \arg \max_{u \in U} Q^*(x, u)$$

- Q^* satisfies the **Bellman optimality equation**:

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u' \in U} Q^*(x', u'), \quad x' = f(x, u)$$

Solution Techniques

Find Q^* , then $h^*(x) = \arg \max_{u \in U} Q^*(x, u)$

1. Model-based:

- known reward model ρ and state transition model f
- compute Q^* off-line (DP, value iteration)

2. Model-free:

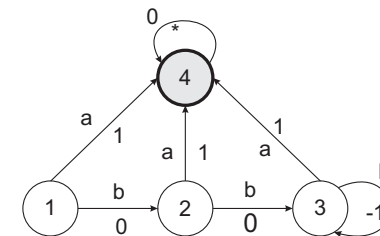
- unknown ρ and f
- learn Q^* online, by interaction with the process

3. Mixed:

- learn ρ and f models by interaction
- interleave (1) and (2)

Bellman Equation: Example

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u' \in U} Q^*(x', u'), \quad x' = f(x, u)$$

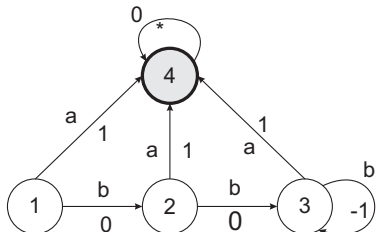


$$Q^*(3, a) = \rho(3, a) = 1$$

$$Q^*(3, b) = \rho(3, b) + \gamma \max\{Q^*(3, a), Q^*(3, b)\} = -1 + 0.9 \cdot 1 = -0.1$$

Bellman Equation: Example

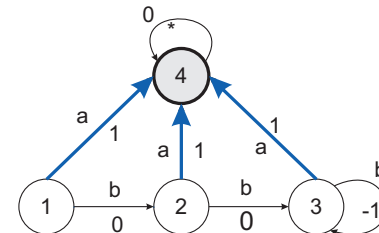
$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u' \in U} Q^*(x', u'), \quad x' = f(x, u)$$



$$Q^*(2, a) = \rho(2, a) = 1$$

$$Q^*(2, b) = \rho(2, b) + \gamma \max\{Q^*(3, a), Q^*(3, b)\} = 0 + 0.9 \cdot 1 = 0.9$$

Optimal Solution



$$Q^*, h^* :$$

$x \backslash u$	a	b
1	1	0.9
2	1	0.9
3	1	-0.1

u
a
a
a

Value Iteration

Turn the Bellman equation into assignment

$$Q(x, u) \leftarrow 0, \quad \forall x \in X, u \in U$$

repeat

for all $x \in X, u \in U$ **do**

$$x' \leftarrow f(x, u)$$

$$Q(x, u) \leftarrow \rho(x, u) + \gamma \max_{u' \in U} Q(x', u')$$

end for

until convergence

$$Q^* \leftarrow Q$$

$$h^*(x) \leftarrow \arg \max_{u \in U} Q^*(x, u), \quad \forall x \in X$$

- Convergence to Q^* guaranteed

Model-Free: Monte-Carlo

- Estimate returns without using ρ, f (not known):
 - run trials in real world, memorize returns for each pair (x, u)
 - compute the discounted return:

$$Q^h(x, u) = \sum_{i=0}^N \gamma^i r_{i+1}$$

- Disadvantages:
 - many tasks are not episodic
 - considerable experience in the world required – costly

Model-Free: Temporal Difference (TD)

- Use experience with the real process (like Monte Carlo)

- Improve value function at each step k :

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha \left[r_{k+1} + \gamma \max_{u' \in U} Q(x_{k+1}, u') - Q(x_k, u_k) \right]$$

- x_k, r_{k+1}, x_{k+1} observed, u_k actually taken action
- **Temporal difference**: measure of the estimation error – deviation from the Bellman equality $Q(x_k, u_k) = r_{k+1} + \gamma \max_{u' \in U} Q(x_{k+1}, u')$
- α is the learning rate

Exploration

- How to choose u ?
- Assume Q is optimal: $u = \arg \max_{u' \in U} Q^*(x, u')$ (greedy action)
- But Q is not (yet) optimal! We need to estimate Q^* !
- \Rightarrow sometimes **explore**:
 - with probability $(1-\varepsilon)$, choose greedy action $u = \arg \max_{u' \in U} Q(x, u')$
 - with probability ε , choose a random **exploratory** action u

Q-Learning

$$Q(x, u) \leftarrow 0, \quad \forall x \in X, u \in U$$

$$k \leftarrow 0$$

loop

observe state x_k

$$u_k \leftarrow \arg \max_{u' \in U} Q(x_k, u')$$

with probability ε , $u_k \leftarrow$ random action

apply u_k , observe r_{k+1} and x_{k+1}

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha \left[r_{k+1} + \gamma \max_{u' \in U} Q(x_{k+1}, u') - Q(x_k, u_k) \right]$$

end loop

- Q-learning converges to Q^*

User-Defined Parameters and Choices

- Discount factor γ (typically 0.9–0.99).
- Learning rate α (typically < 0.5).
- Exploration probability ε (typically < 0.5).
- State and action space discretization (ad hoc).
- Trial definition, immediate rewards (ad hoc).

Modifications, Extensions

- On-policy learning.
- Eligibility traces.
- Interpolation methods for continuous spaces.
- Actor-critic methods.
- Mixed model-based – model-free learning.
- Stochastic setting (probabilistic rewards and transitions).
- Multi-agent systems.

Basic Elements of Reinforcement Learning

- Model of the process (includes all agents).
- Reward functions (each agent may have its own).
- Learning objective (may be hard to define).
- Controllers (policies, agents).
- Exploration.

Multi-Agent Case: Stochastic Game

A **stochastic game** is a tuple $\langle A, X, \{U_i\}_{i \in A}, f, \{\rho_i\}_{i \in A} \rangle$, where:

- $A = 1, \dots, n$ is the set of agents
- X is the finite state space
- U_i is the finite set of agent's actions, $\mathbf{U} = \times_{i \in A} U_i$
- $f : X \times \mathbf{U} \times X \rightarrow [0, 1]$ is the state transition probability distr.
- $\rho_i : X \times \mathbf{U} \times X \rightarrow \mathbb{R}$ is the agent's reward function

State transitions and rewards depend on the joint action!

Consequences for Learning

- Optimal policy at any moment depends on policies of other agents.
- Other agents are also learning \rightarrow moving target.
- Need to take special care of convergence.
- Policies found may not be optimal against specific opponent's policies.
- Depending on the goal, different settings (tasks) possible.

Classification According to Tasks

- Cooperative tasks: $\rho_1 = \rho_2 = \dots = \rho_n$
- Competitive tasks: typically $\rho_1 = -\rho_2$ (cf. zero-sum games)
- Mixed tasks: general case (cf. general-sum games)

RL in MAS: Approaches

- **Ignore** the presence of other learning agents:
→ apply single-agent reinforcement learning
- **Be aware** of other learning agents:
→ guarantee convergence independently of other agents
- **Adapt** to (track) other agents:
→ strive for optimality / rationality

Agents may be heterogeneous: goals, assumptions, algorithms.

Single-Agent Learning

Assumption: Agents interfere weakly

Methods:

- Ignore other agents
- Consider other agents indirectly (through the reward signal)

Some applications reported: (Matarić, 1996; Crites and Barto, 1996)

Convergence not guaranteed.

Cooperative Multi-Agent Learning

Typical approach: use $Q(x, \mathbf{u})$, where

x – global world state, including all agents' states

\mathbf{u} – joint action of all agents (must be observed)

- Use Q-learning to find Q^*

$$Q_{k+1}(x_k, \mathbf{u}_k) = Q_k(x_k, \mathbf{u}_k) + \alpha [r_{k+1} + \gamma \max_{\mathbf{u}'} Q(x_{k+1}, \mathbf{u}') - Q_k(x_k, \mathbf{u}_k)]$$

- Greedy policy $\mathbf{u} = \arg \max_{\mathbf{u}' \in U} Q^h(x, \mathbf{u}')$

However: Agents may break ties differently!

Ensuring Convergence

- **Assume unique maxima** of Q (team Q-learning, Littman, 2001)
- **Coordination-free** methods – e.g., distributed Q-learning (Lauer and Riedmiller, 2000) – only for deterministic setting
- **Coordination-based** methods:
 - Direct coordination: social conventions, roles, negotiation (Boutilier, 1996)
 - Indirect coordination: learn models of other agents (Joint Action Learners, Claus and Boutilier, 1998)

Fully Competitive Tasks

Opponent-independent: solve the dynamic task stage-wise

$$h_{1,k}(x_k, \cdot) = \arg \mathbf{m}_1(Q_k, x_k)$$

$$Q_{k+1}(x_k, u_{1,k}, u_{2,k}) = Q_k(x_k, u_{1,k}, u_{2,k}) + \alpha [r_{k+1} + \gamma \mathbf{m}_1(Q_k, x_{k+1}) - Q_k(x_k, u_{1,k}, u_{2,k})]$$

where \mathbf{m}_1 is the minimax (worst-case) return of agent 1:

$$\mathbf{m}_1(Q, x) = \max_{h'_1(x, \cdot)} \min_{u_2} \sum_{u_1} h'_1(x, u_1) Q(x, u_1, u_2)$$

agent might do better if it has a model of the opponent's policy

Mixed Tasks

Opponent-independent: solve the dynamic task stage-wise

$$h_{i,k}(x, \cdot) = \text{solve}_i \{Q_{\cdot,k}(x_k, \cdot)\}$$

$$Q_{i,k+1}(x_k, \mathbf{u}_k) = Q_{i,k}(x_k, \mathbf{u}_k) + \alpha [r_{i,k+1} + \gamma \cdot \text{eval}_i \{Q_{\cdot,k}(x_{k+1}, \cdot)\} - Q_{i,k}(x_k, \mathbf{u}_k)]$$

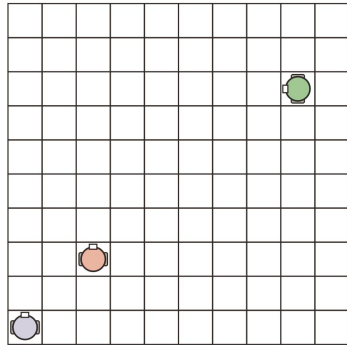
- **solve** – game-theoretic equilibrium in stage game
- **eval** – expected return for this equilibrium

Target convergence, but cannot exploit a specific opponent's policy

Limitations and Challenges

- Convergence vs. optimality / rationality
- Continuous domains:
 - discretization may not always work
 - function approximation - convergence not guaranteed
- Curse of dimensionality (exponential explosion of Q-tables):
 - computational problems (memory and speed)
 - slow convergence of learning

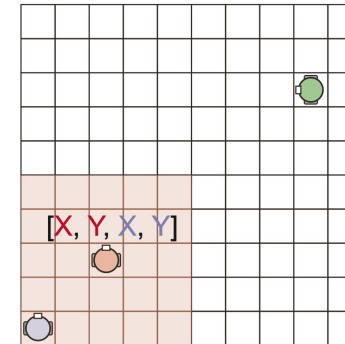
State Space Explosion



Q-table size: $100^3 \cdot 4^3 = 64$ million entries!
Large search space \Rightarrow slow learning speed

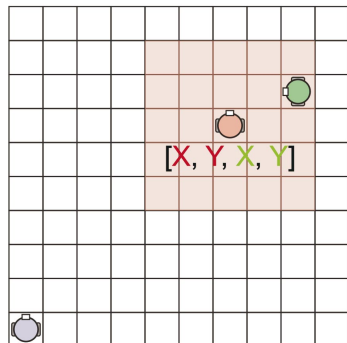
State Space Explosion

Other agents' positions most likely irrelevant to the red agent unless they are nearby:

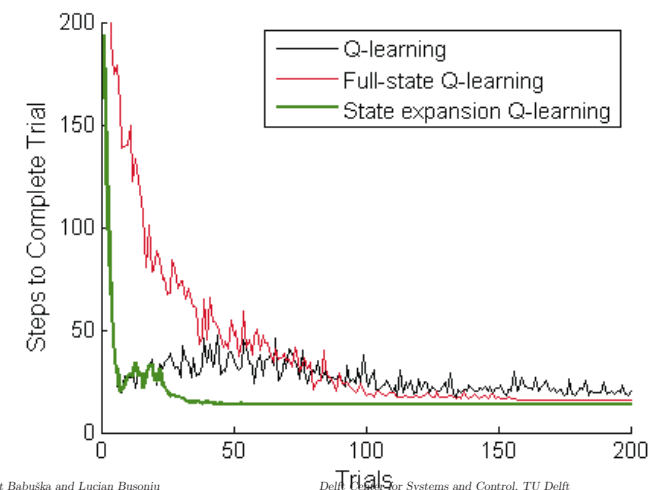


State Space Explosion

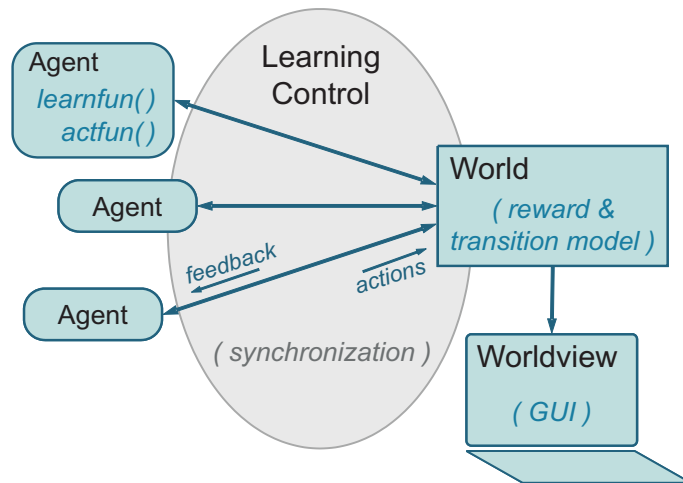
Other agents' positions most likely irrelevant to the red agent unless they are nearby:



MAS Learning: Rate of Convergence



Matlab Toolbox for RL MAS



Concluding Remarks

- **Single-agent reinforcement learning**
 - well developed for discrete state and action spaces
 - unresolved issues in continuous spaces
 - applicable to moderate-size problems
- **Multi-agent reinforcement learning**
 - much more intricate than single-agent
 - applicable to small-size problems
- **Learning in general**
 - not a panacea, the modeling / design problem only shifted
 - performance improvement not monotonous

References

- [1] Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *Proceedings Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK-96)*, pages 195–210, De Zeeuwse Stromen, The Netherlands.
- [2] Bowling, M. (2004). Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems 17 (NIPS-04)*, pages 209–216, Vancouver, Canada.
- [3] Bowling, M. and Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250.
- [4] Crites, R. H. and Barto, A. G. (1998). Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2–3):235–262.
- [5] Guestrin, C., Lagoudakis, M. G., and Parr, R. (2002). Coordinated reinforcement learning. In *Proceedings Nineteenth International Conference on Machine Learning (ICML-02)*, pages 227–234, Sydney, Australia.

- [6] Lauer, M. and Riedmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings Seventeenth International Conference on Machine Learning (ICML-00)*, pages 535–542, Stanford University, US.
- [7] Littman, M. L. (2001). Value-function reinforcement learning in Markov games. *Journal of Cognitive Systems Research*, 2:55–66.
- [8] Mataric, M. J. (1996). Learning in multi-robot systems. In Weiß, G. and Sen, S., editors, *Adaptation and Learning in Multi-Agent Systems*, pages 152–163. Springer Verlag.
- [9] Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434.
- [10] Powers, R. and Shoham, Y. (2004). New criteria and a new algorithm for learning in multi-agent systems. In *Advances in Neural Information Processing Systems 17 (NIPS-04)*, pages 1089–1096, Vancouver, Canada.
- [11] Wang, X. and Sandholm, T. (2002). Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *Advances in Neural Information Processing Systems 15 (NIPS-02)*, pages 1571–1578, Vancouver, Canada.