Reinforcement Learning

Multi-Agent Reinforcement Learning II

Stefano V. Albrecht, Michael Herrmann

12 March 2024

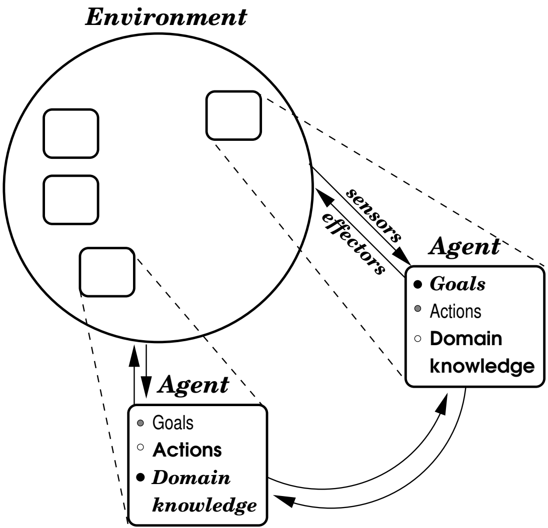
THE UNIVERSITY of EDINBURGH
informatics

Lecture Outline

- Independent learning
- Joint action learning
- Game-theoretic RL
- Opponent modelling RL
- Learning in mixed groups

Recap: Multi-Agent Systems

- Multiple agents interact in shared environment
- Each agent with own observations, actions, goals, …
- Agents must coordinate actions to achieve their goals



Multi-Agent Learning

Last time we discussed:

- Models of multi-agent interaction

  $\Rightarrow$ Repeated games, Stochastic games

- Solution concepts for games

  $\Rightarrow$ For common rewards: maximise expected return (like MDP)

  $\Rightarrow$ Zero-sum/general rewards: minimax, Nash equilibrium, Pareto, welfare, ...

Now: multi-agent learning

- Can agents *learn* to solve game through repeated interactions?

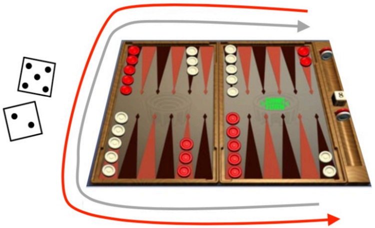
Independent Learning

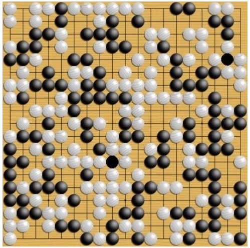
Basic approach: independent learning (IL)

- Each agent uses a single-agent RL algorithm (e.g. Q-learning)
- Treat game like MDP, agents do not model other agents

IL can be successful:

- TD-Gammon used IL,
  beat Backgammon champion
- AlphaGo used IL,
  beat Go champion





Independent Learning

Independent Q-Learning (we control agent i):

Initialise: $Q_i(s, a_i) = 0$ for all $s \in S, a_i \in A_i$

repeat:

  Observe current state s

  With probability ϵ: choose random action a_i

  Else: choose greedy action $a_i \in \arg\max_{a_i} Q_i(s, a_i)$

  Observe own reward r_i and next state s'

  $Q_i(s, a_i) \leftarrow Q_i(s, a_i) + \alpha \left[r_i + \gamma \max_{a_i'} Q_i(s', a_i') - Q_i(s, a) \right]$

Independent Learning

Problem with IL: high variance in updates

- Independent Q-learners: each agent i maintains Q-table $Q_i(s, a_i)$
- After reward $r_i = u_i(s, a_1, ..., a_n)$, update $Q_i(s, a_i)$ toward $r_i + \gamma \max_{a_i'} Q_i(s', a_i')$

Repeated RPS:

- If $(a_1, a_2) = (R, S)$, then $r_1 = +1$
- If $(a_1, a_2) = (R, P)$, then $r_1 = -1$

$\Rightarrow$ Agent 1 cannot tell when reward is $+1/-1$!

(unless we add actions to state; why?)

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

Joint Action Learning

Reduce variance by learning values for joint actions: $Q_i(s, a_1, ..., a_n)$

- Now can differentiate between $+1/-1$ rewards
- Space requirement is exponential in agents, $O(|A_1 \times \cdots \times A_n|)$
- Use function approximation to compress and generalise

Joint Action Learning

Reduce variance by learning values for joint actions: $Q_i(s, a_1, ..., a_n)$

- Now can differentiate between $+1/-1$ rewards
- Space requirement is exponential in agents, $O(|A_1 \times \cdots \times A_n|)$
- Use function approximation to compress and generalise

But: $Q_i(s, a_1, ..., a_n)$ alone is no longer enough to find best action for i

- How to evaluate $\max_{a_i} Q_i(s, a_1, ..., a_n)$?

  $\Rightarrow$ Best action depends on actions of other agents!

How to select action from Q_i? How to update Q_i?

Game-Theoretic Reinforcement Learning

Joint action Q-tables define normal-form game:

- Agent i stores a Q-table Q_j for every agent $j \in N$ (assumes agent can observe all agents' actions and rewards)
- Reward functions for normal-form game in state s are $u_j(a_1, ..., a_n) = Q_j(s, a_1, ..., a_n)$

We can *solve* the normal-form game defined by

$$\Gamma_s \doteq \left(u_1 = Q_1(s), \cdots, u_n = Q_n(s)\right)$$

Solution of Γ_s is a policy profile $(\pi_1, \ldots, \pi_n)$ with certain properties (e.g. NE)

$\Rightarrow$ Use π_i to select action for agent i

Value of Γ_s to agent j is expected reward under solution $(\pi_1, \ldots, \pi_n)$

$$Val_j(\Gamma_s) = \sum_{a \in A} u_j(a) \prod_{k \in N} \pi_k(a_k)$$

Now:

$\Rightarrow$ Update Q_j towards target: $r_j + \gamma \, Val_j(\Gamma_{s'})$

Joint Action Learning with Game Theory

JAL-GT (we control agent i):

Initialise: $Q_j(s,a) = 0$ for all $j \in N$ and $s \in S, a \in A$
repeat:
  Observe current state s
  With probability ϵ: choose random action a_i
  Else: solve Γ_s to get policies $(\pi_1, ..., \pi_n)$, then sample action $a_i \sim \pi_i(s)$
  Observe joint action $a = (a_1, ..., a_n)$, rewards r_j for all j, and next state s'
  for each j **do**
    $Q_j(s,a) \leftarrow Q_j(s,a) + \alpha \left[r_j + \gamma\, Val_j(\Gamma_{s'}) - Q_j(s,a) \right]$

Minimax-Q, Nash-Q, CE-Q

Minimax-Q uses minimax solution (Littman, 1994)

- Converges to unique value in two-player zero-sum games

  $\Rightarrow$ Any such game has unique minimax value

- Minimax profile can be computed with linear programming (LP)

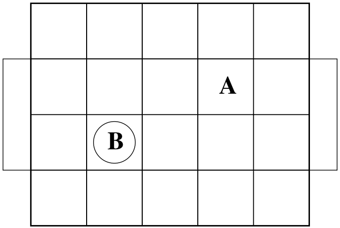
Nash-Q uses Nash equilibrium (Hu and Wellman, 2003)
CE-Q uses correlated equilibrium (Greenwald and Hall, 2003)

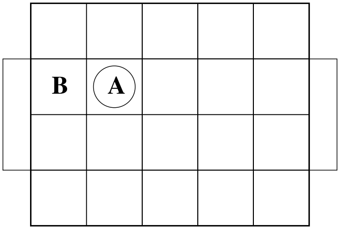
- Converges to equilibrium under highly restrictive conditions

  $\Rightarrow$ Problem: often no unique equilibrium value in general-reward games

- Compute CE with LP, compute NE with quadratic programming

Example: Minimax-Q in Grid Soccer (Littman, 1994)





- Episodes start in left state with random ball assignment
- Agent wins episode if it moves the ball into opponent goal
- Agent loses ball to opponent if it moves into opponent's location

Against *unknown* opponent, optimal policy must randomise (right state; why?)

Example: Minimax-Q in Grid Soccer (Littman, 1994)

	MR		MM		QR		QQ	
	% won	games	% won	games	% won	games	% won	games
vs. random								
vs. hand-built								
vs. MR-challenger								
vs. MM-challenger								
vs. QR-challenger								
vs. QQ-challenger								

Table 3: Results for policies trained by minimax-Q (MR and MM) and Q-learning (QR and QQ).

- MR: minimax-Q trained against random opponent
- MM: minimax-Q trained against minimax-Q
- QR: Q-learning trained against random opponent
- QQ: Q-learning trained against Q-learning (IL)
- “X-challenger” is optimal policy against final policy learned by X

Example: Minimax-Q in Grid Soccer (Littman, 1994)

	MR		MM		QR		QQ	
	% won	games	% won	games	% won	games	% won	games
vs. random	99.3	6500	99.3	7200				
vs. hand-built	48.1	4300	53.7	5300				
vs. MR-challenger	35.0	4300						
vs. MM-challenger			37.5	4400				
vs. QR-challenger								
vs. QQ-challenger								

Table 3: Results for policies trained by minimax-Q (MR and MM) and Q-learning (QR and QQ).

- Minimax-Q learns "safe" policy that works against any opponent

  $\Rightarrow$ Minimax policy guarantees minimum average 50% win

- Lower % win against challenger because MR/MM did not fully converge during training, so could be exploited by optimal challenger

Example: Minimax-Q in Grid Soccer (Littman, 1994)

	MR		MM		QR		QQ	
	% won	games	% won	games	% won	games	% won	games
vs. random	99.3	6500	99.3	7200	99.4	11300	99.5	8600
vs. hand-built	48.1	4300	53.7	5300	26.1	14300	76.3	3300
vs. MR-challenger	35.0	4300						
vs. MM-challenger			37.5	4400				
vs. QR-challenger					0.0	5500		
vs. QQ-challenger							0.0	1200

Table 3: Results for policies trained by minimax-Q (MR and MM) and Q-learning (QR and QQ).

- Q-learning optimises against specific opponent, can learn strong performance
- **Problem:** overfits to opponent, does not generalise well to other opponents

  $\Rightarrow$ Challenger exploits deterministic Q-learning policies

Opponent Modelling & Best Response

Game theory solutions are normative: they *prescribe* how agents *should* behave

- E.g. minimax assumes worst-case opponent
- E.g. NE assumes agents are perfect rational optimisers

$\Rightarrow$ What if agents don't behave as prescribed by solution?

Opponent Modelling & Best Response

Game theory solutions are normative: they *prescribe* how agents *should* behave

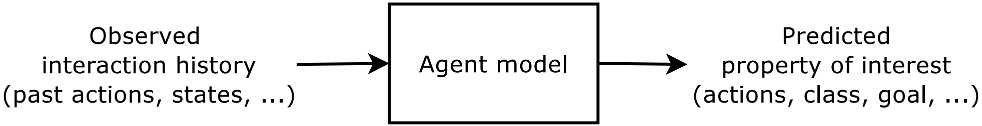
- E.g. minimax assumes worst-case opponent
- E.g. NE assumes agents are perfect rational optimisers

  $\Rightarrow$ What if agents don't behave as prescribed by solution?

Other approach: **opponent modelling with best response**

- Learn models of other agents to predict their actions
- Compute optimal action (best response) against agent models

Opponent Modelling



Many kinds of opponent modelling:

- Policy reconstruction
- Type-based reasoning
- Classification
- Plan recognition
- Recursive reasoning
- Graphical methods
- Group modelling
- Implicit modelling

Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems. *Artificial Intelligence*, 2018

Policy Reconstruction

Policy reconstruction: learn model $\hat{\pi}_j \approx \pi_j$ from observations

Conditional action frequency:

$$\hat{\pi}_j(s, a_j) \propto \sum_{t:\, s^t = s} [a_j^t = a_j]_1$$

Many modifications possible → Ideas?

In general, can train model with supervised learning on pairs (s^t, a_j^t)

- E.g. decision tree, neural network, finite state machine, ...
- Model should support incremental updating

Best Response

Expected value of action a_i in state s against models $\hat{\pi}_j$ is

$$EV(s, a_i) = \sum_{a_{-i}} Q(s, a_i, a_{-i}) \prod_{j \neq i} \hat{\pi}_j(s, a_j)$$

Assumes independent agents (why?)

a_{-i} is action tuple for all agents except i

Best response is action with maximum expected value: $\arg\max_{a_i} EV(s, a_i)$

Use $EV(s, a_i)$ in place of Q-table for action selection and update targets

Joint Action Learning with Opponent Modelling

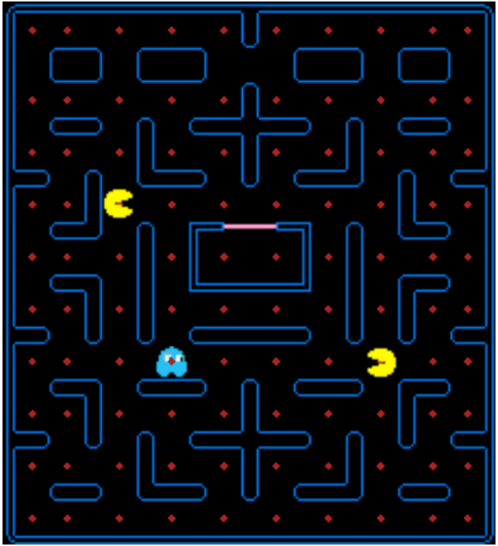
JAL-OM (we control agent i):

Initialise: $Q_i(s,a) = 0$ for all $s \in S, a \in A$; models $\hat{\pi}_j(s,\cdot) = \frac{1}{|A_j|}$ for $j \neq i$
repeat:
    Observe current state s
    With probability ϵ: choose random action a_i
    Else: choose best-response action $\arg\max_{a_i} EV(s, a_i)$
    Observe joint action $a = (a_1, ..., a_n)$, own reward r_i, and next state s'
    for each j **do**
        Update model $\hat{\pi}_j$ with new observations
    $Q_i(s,a) \leftarrow Q_i(s,a) + \alpha \left[r_i + \gamma \max_{a_i'} EV(s', a_i') - Q_i(s,a) \right]$

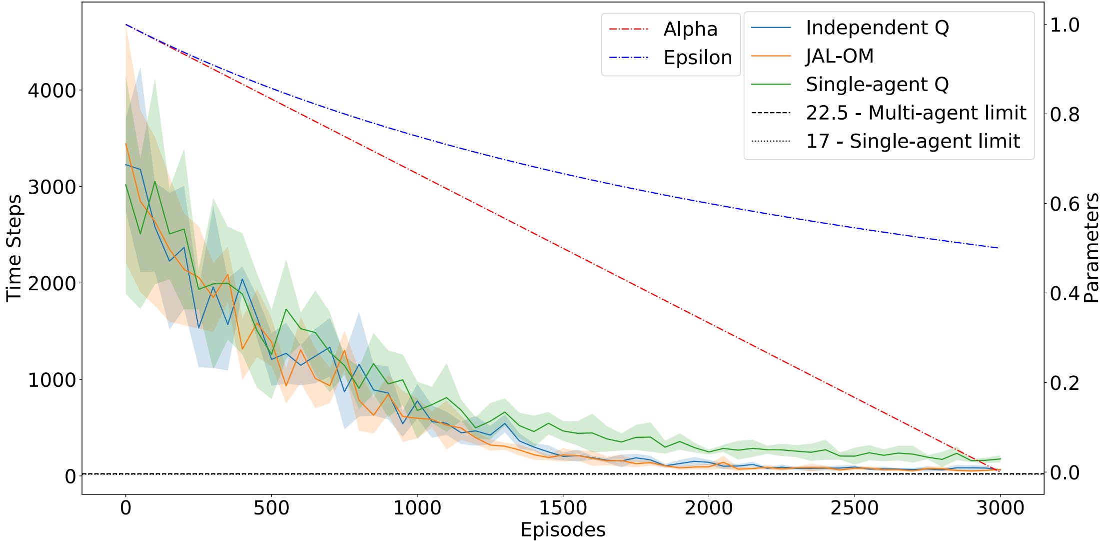
Example: Multi-Pacman

Pacmans must catch the ghost

- Actions: move up, down, left, right
- States: (P_1, P_2, G) = locations (red dot) of pacmans and ghost
- Ghost moves randomly
- Reward to both pacmans:
  $+1$ if ghost is caught, else 0 ($\gamma = 0.8$)



Example: Multi-Pacman – 10x10 Grid, 2 Agents, 1 Ghost

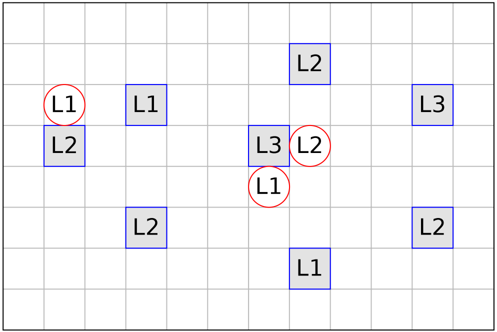


Video: learned JAL policies

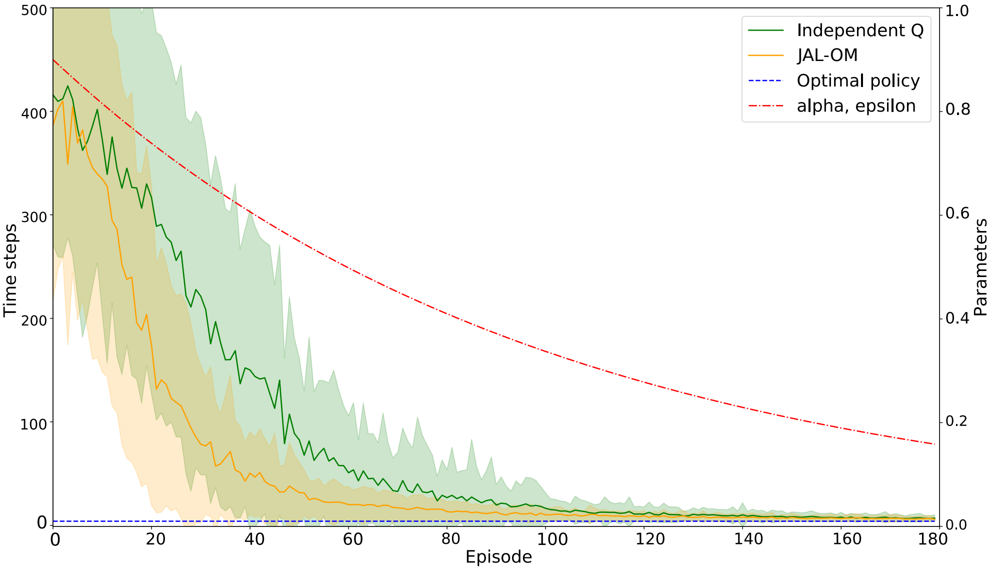
Example: Level-Based Foraging

Robots must collect items in minimal time

- Actions:
  - move up, down, left, right
  - try to load item
- Robots can load item if positioned next to item and sum of robots' levels $\geq$ item level
- Reward to robot *i*:
  - $+1$ if involved in successful loading
  - -1 if trying to move outside grid
  - 0 otherwise



Example: Level-Based Foraging – 5x5 Grid, 2 Agents, 1 Item



Learning in Mixed Groups

Standard mode of operation is self-play: all agents use same algorithm

Bonus question: how do algorithms perform in mixed groups?

Tested 5 algorithms in mixed learning groups:

- Nash-Q: game-theoretic RL
- JAL and CJAL: opponent modelling RL
- WoLF-PHC (Bowling and Veloso, 2002)
- Regret Matching (Hart and Mas-Colell, 2001)

Test criteria:

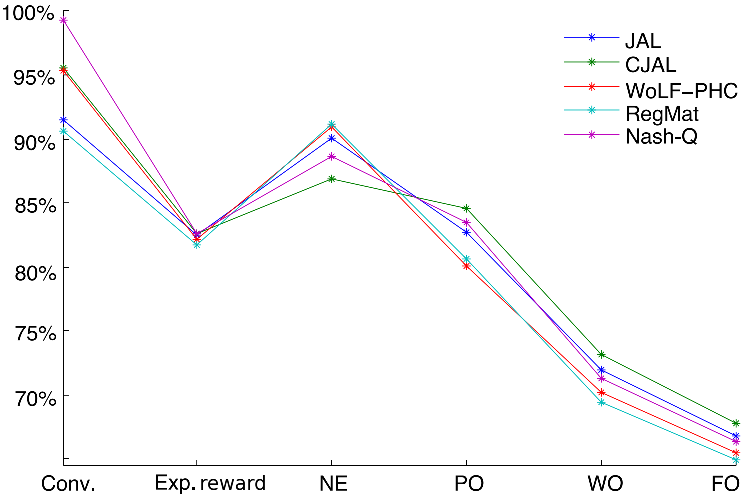
- Convergence rate
- Final expected rewards
- Social welfare/fairness
- Solution rates:
  - Nash equilibrium (NE)
  - Pareto-optimality (PO)
  - Welfare-optimality (WO)
  - Fairness-optimality (FO)

- Tested in 78 distinct, strictly ordinal 2×2 repeated games, e.g.

1,2	2,4
4,1	3,3

- Also tested in 500 random, strictly ordinal $2 \times 2 \times 2$ (3 agents) repeated games

Learning in Mixed Groups — No Clear Winner



100% is highest possible

No clear winner!

See (Albrecht and Ramamoorthy, 2012) for details

Reading (Optional)

- Useful summary: M. Bowling, M. Veloso (2000). An analysis of stochastic game theory for multiagent reinforcement learning. CMU-CS-00-165
- Survey on opponent modelling:
S. Albrecht, P. Stone (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence, 258:66–95
`https://arxiv.org/abs/1709.08071`
- Tutorial with more algorithms and recent developments:
S. Albrecht, P. Stone (2017). Multiagent Learning: Foundations and Recent Trends
`http://www.cs.utexas.edu/~larg/ijcai17_tutorial`

References

S. Albrecht and S. Ramamoorthy. Comparative evaluation of MAL algorithms in a diverse set of ad hoc team problems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 349–356, 2012.

M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

A. Greenwald and K. Hall. Correlated Q-learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 242–249, 2003.

S. Hart and A. Mas-Colell. A reinforcement procedure leading to correlated equilibrium. *Economic Essays: A Festschrift for Werner Hildenbrand*, pages 181–200, 2001.

J. Hu and M. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.