MULTI-AGENT REINFORCEMENT LEARNING

Jonatan Milewski

Outline

- Introduction
- Reinforcement Learning
  - Q-Learning
- Multi-Agent Reinforcement Learning
  - Minimax Q-Learning
  - Q-Learning in general-sum games
- Conclusion

Introduction

- Context: Repeated Games & Stochastic Games
- Want to learn the best strategy against the opponent(s)
- Might not know all the payoff values beforehand
- Might not know the transition probabilities between states (in a stochastic game)
- Can use Reinforcement Learning!

Reinforcement Learning

- Inspired by behaviorist psychology
- Learn by interacting with the environment
- Trial-and-error approach
- Positive feedback encourages given behavior
- Negative feedback discourages given behavior
- Balance between exploration and exploitation
- Long-term payoff

Q-Learning

- Environment consists of states
- From each state agent can choose an action
- Each action has an associated reward
- After performing action, agent moves to another state (maybe)

Q-Learning

- Each state-action pair has a corresponding Q-value: represents expected cumulative payoff from performing action in the given state
- Update Q each time:

Learning rate (between 0 and 1)

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_a Q(s',a))$$

Discount factor (between 0 and 1)

- Goal: Find “optimal policy” i.e. actions that maximize V(s)

$$V(s) \leftarrow \max_a Q(s,a)$$

Q-Learning

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_a Q(s',a))$$

- How are actions chosen?
  - Randomly, with probability *explor* exploration
  - According to max Q(s,a) with probability *1 – explor* exploitation

Q-Learning

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a} Q(s', a))$$

- How are actions chosen?
  - Randomly, with probability *explor* exploration
  - According to max Q(s,a) with probability *1 – explor* exploitation

- What's a good learning rate (alpha) ?
  - 1/k
  - 0.1
  - …

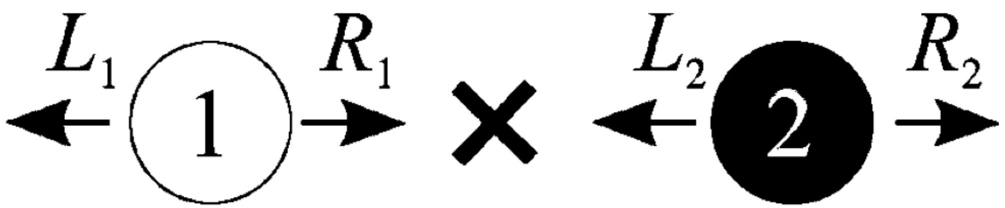
Outline

- Introduction
- Reinforcement Learning
  - Q-Learning
- Multi-Agent Reinforcement Learning
  - Minimax Q-Learning
  - Q-Learning in general-sum games
- Conclusion

Play This Game Repeatedly:

Player 2

Player 1		**L2**	**R2**
	L1	**0,0**	**1,-1**
	R1	**-10,10**	**10,-10**



Q-Learning in Zero-Sum Stochastic Games

- Naïve approach: apply Q-learning directly
  - Might not work well against a good opponent with a complex strategy
  - No guarantee of convergence
- Better approach: play MaxMin and converge to Nash

Minimax Q-Learning

- Q-values are over joint actions: Q(s,a,o)
  - s = state
  - a = your action
  - o = action of the opponent
- Instead of playing action with highest Q(s,a,o), play MaxMin

$$Q(s,a,o) \leftarrow (1-\alpha)Q(s,a,o) + \alpha(r + \gamma V(s'))$$

$$V(s) \leftarrow \max_{\pi_s} \min_{o} \sum_{a} Q(s,a,o)\pi_s(a)$$

probability of playing *a* when following strategy π_s

Minimax Q-Learning

- How are actions chosen?
  - At the beginning set π_s to select actions uniformly at random for each state
  - Before each step:
    - Play random action with probability *explor*
    - Play according to π_s with probability *1 – explor*
  - After each step:
    - Update π_s to the MaxMin strategy (based on Q(s,a,o))

Minimax Q-Learning

- Does it work?
  - Performs better than naïve Q-learning
  - Guarantees convergence to Nash equilibrium (under certain conditions)
  - No guarantee of rate of convergence ☹

Q-Learning in General-Sum Games

- A much harder problem
- Nash Q-Learning:

$$Q(s, a_1, ..., a_n) \leftarrow (1-\alpha)Q(s, a_1, ..., a_n) + \alpha(R + \gamma NashV(s'))$$

- *NashV(s)* is the payoff value from computing a Nash equilibrium
- Must keep track of all players' Q-values to compute *NashV(s)*
- Assumes all players play the same Nash equilibrium

Belief-based Reinforcement Learning

$$Q_{t+1}(s_t, a_t) \leftarrow (1 - \alpha)Q_t(s_t, a_t) + \alpha_t(r(s_t, a_t) + \beta V_t(s_{t+1}))$$

$$V_t(s) \leftarrow \max_{a_i} \sum_{a_{-i} \subset A_{-i}} Q_t(s, (a_i, a_{-i}))Pr_i(a_{-i})$$

- Uses some beliefs about opponents' strategy to calculate V(s)
- Ideally beliefs are updated after each move

Other approaches

- A Nash equilibrium is not always the “best” way to play
- Can use other solution concepts:
  - Correlated equilibrium
  - Pareto-optimality
  - Regret
  - …
- Methods developed for specific kinds of games
  - E.g. “coordination games” (Battle of the Sexes)

Conclusion

- Reinforcement learning can be useful in learning strategies in stochastic games
- It is not necessary to know the payoff matrix and transition probabilities beforehand
- Many methods' success depends on the accuracy of assumptions about other players' strategies

THANK YOU

Jonatan Milewski

References

- Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations,* Cambridge University Press, 2009.
- M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning.” *ICML.* Vol. 94. 1994.
- A. Nowe, P. Vrancx, Y.-M. De Hauwere, “Game Theory and Multi-agent Reinforcement Learning.”, *Reinforcement Learning*, ALO 12, pp. 441–470.
- J. Hu, M. P. Wellman. “Nash Q-learning for general-sum stochastic games.” *The Journal of Machine Learning Research 4* (2003): 1039-1069.
- L. Busoniu, R. Babuska and B. De Schutter. “A comprehensive survey of multiagent reinforcement learning.” *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE transactions on* 38.2 (2008): 156-172.
- D. Bloembergen and D. Hennes. “Fundamentals of multi-agent reinforcement learning.” *AAMAS MARL Tutorial*. 2013.