ON-POLICY CONCURRENT LEARNING

Vlad Chiriacescu, Wei Li, Sean Hicks
December 6, 2011

Overview

Introduction

- Problem:
  Off-policy Q-learning methods do not scale well in multiagent environments
- Solution:
  Design of on-policy Q-learning methods that are scalable and efficient

Introduction

- The reinforcement learning paradigm provides techniques using which an individual agent can optimize its environmental payoff.
- Standard reinforcement learning techniques like Q-learning are not guaranteed to converge in a multi-agent environment. (due to the non-stationary nature of the environment)
- SARSA, an on-policy version of Q-learning, with function approximation has been demonstrated to converge to a bounded region at worst (Gordon 2000).

Introduction

In this paper:

- Presenting SARSA Q-learning for competitive and general-sum domains
- Proving convergence to minimax and Nash equilibrium value-functions in the limit, under appropriate assumptions.
- Showing experimentally that the new method can not only learn better policies in competitive domains, but can also learn faster in general-sum domains.

Multi-agent Q-learning

- A Markov Decision Process (MDP) is a quadruple {S, A, T, R}, where S is the set of states, A is the set of actions, T is the transition function, $T: S \times A \rightarrow PD(S)$, PD being a probability distribution and R is the reward function $R: S \times A \rightarrow \Re$.

$$v(s, \pi_i) = \sum_{t=0}^{\infty} \gamma^t E(r_t^i | \pi_i, \pi_{-i}, s_0 = s)$$

s_0 = initial joint state
r_t^i = reward of the ith agent at time t
γ = discount factor

Multi-agent Q-learning

- A bimatrix game is given by a pair of matrices, (M_1 and M_2), where the payoff of the kth agent for the joint action (a_1, a_2) is given by the entry $M_k(a_1, a_2), \forall (a_1, a_2) \in A_1 \times A_2,\ k = 1,2$
- A mixed-strategy Nash Equilibrium for a bimatrix game (M_1, M_2) is pair of probability vectors (π_1^*, π_2^*) such that

$$\pi_1^{*T} M_1 \pi_2^* \geq \pi_1^T M_1 \pi_2^*, \quad \forall \pi_1 \in PD(A_1).$$
$$\pi_1^{*T} M_2 \pi_2^* \geq \pi_1^{*T} M_2 \pi_2, \quad \forall \pi_2 \in PD(A_2).$$

PD(A_i) = set of probability-distributions over the ith agent's action space

Multi-agent Q-learning

- Q-learning for an individual learner

$$Q^{t+1}(s_t, a_t) = (1 - \alpha_t)Q^t(s_t, a_t) + \alpha_t[r_t + \gamma v^t(s_{t+1})]$$
$$v^t(s_{t+1}) = \max_a Q^t(s_{t+1}, a)$$

- Minimax-Q algorithm for simultaneous-move zero-sum games

$$v_1^t(s_{t+1}) = \max_{\pi \in PD(A)} \min_{o \in O} \pi^T Q_1^t(s_{t+1}, \cdot, o)$$

- For general-sum games, each agent observes the other agent's actions and rewards and maintains separate action values for each of them in addition to its own

$$v_1^t(s_{t+1}) = \pi_1^*(s_{t+1})^T Q_1^t(s_{t+1}, \cdot, \cdot)\pi_2^*(s_{t+1})$$

On-policy concurrent Q-learning

- Off-policy learning - the update of Q depends on V which relies on actions that are not taken
- On-policy learning - Q depend on the actual learning policy followed
- Off-policy algorithms separate control from exploration but on-policy methods are superior in control and prediction problems
- SARSA (on-policy method) converges to a stable Q value while the classic Q-learning diverges

On-policy concurrent Q-learning

- Minimax-SARSA learning
  - classic SARSA update rule (in simple Q-learning scenario):

$$v^t(s_{t+1}) = Q^t(s_{t+1}, a_{t+1})$$

  - minimax-SARSA update rule (in multi-agent minimax-Q setting):

$$Q_1^{t+1}(s_t, a_t, o_t) = (1 - \alpha_t)Q_1^t(s_t, a_t, o_t) + \alpha_t[r_t^1 + \gamma Q_1^t(s_{t+1}, a_{t+1}, o_{t+1})]$$
$$v_1^t(s_{t+1}) = Q_1^t(s_{t+1}, a_{t+1}, o_{t+1})$$
$$Q_1^*(s_t, a_t, o_t) = R_1(s_t, a_t, o_t) + E_y[\max_{\pi_1} \min_o \pi_1^T Q_1^*(y, \cdot, o)]$$

- paper shows that minimax-SARSA learning converges to minimax-Q values if agents actions follow MLIE scheme (Minimax in the limit with infinite exploration)

On-policy concurrent Q-learning

- Nash-SARSA learning
  - extends SARSA technique to Nash learning in general-sum domains

$$Q_1^*(s_t, a_t, o_t) = R_1(s_t, a_t, o_t) + E_y[\pi_1^{*T} Q_1^*(y, \cdot, \cdot)\pi_2^*]$$

  - similar strategy for convergence proof as in minimax-SARSA
  - converges to Nash equilibrium if actions follow NELIE strategy (Nash equilibrium in the limit with infinite exploration)
  - restrictions for obtaining convergence limit the applicability of the method

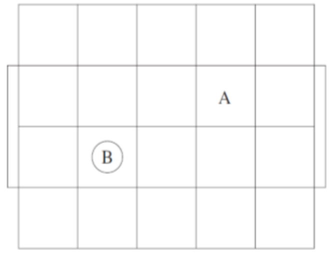
On-policy concurrent Q-learning

- Minimax-Q(λ)
  - Uses Time Difference (TD) estimators
    - Combines Monte-Carlo and dynamic programming
    - λ represents the trace decay parameter. Higher λ means longer lasting traces (e.g. rewards that come from states and actions which are further away)
  - Used in this paper to provide comparison to other methods

Experiments in competitive domain

- Soccer game
  - 4x5 grid
  - Two agents (A and B) try to score goal
    - Can move up, down, left, right, or stay put
    - If a moving agent hits a stationary agent who has the ball, the moving agent gets the ball
    - Reward +1 for goal, -1 for opponent scoring or self goal (zero sum game)
    - Resets once goal is scored

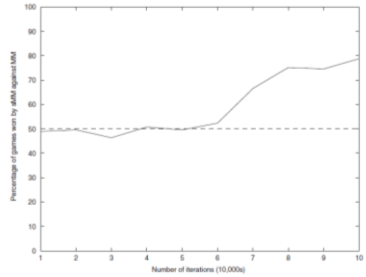
Experiments in competitive domain



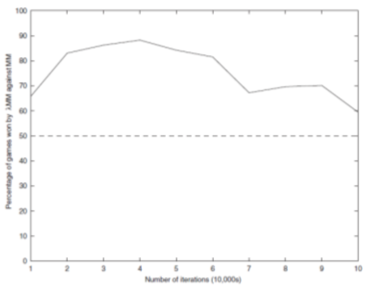
Experiments in competitive domain

- Soccer game
  - Conducted i x 10,000 iterations (i = 1 to 10)
    - Minimax-SARSA vs. Ordinary Minimax
    - Minimax-Q(λ) vs. Ordinary Minimax
    - Minimax-Q(λ) vs. Minimax-SARSA
  - Exploration probability = 0.2
  - Decay-factor for learning = 0.999954
  - λ = 0.7

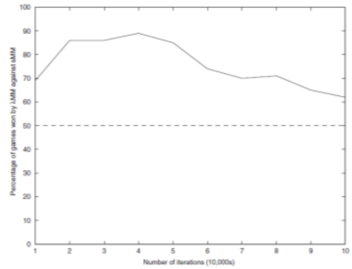
Experiments in competitive domain



Experiments in competitive domain



Experiments in competitive domain



Experiments in competitive domain

- Results
  - Ordinary minimax-Q initially dominates, but SARSA catches up and outperforms
  - Q(λ) initially outperforms minimax, but loses edge as minimax learns better progressively (also seen vs SARSA)
  - SARSA performs better than minimax because minimax is pessimistic in nature and SARSA backpropagates information more expeditiously
  - Results are far from convergence, where all algorithms would perform equally well

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

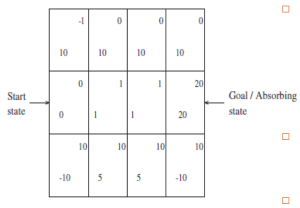
Experiments in general-sum domain

- A domain where the rationale of the assumption of minimizing policy of the opponent is to guarantee a minimum security level to the learner, instead of maximizing the reward of the opponent itself as in the zero-sum interpretation.
- A general-sum problem called 'tightly coupled navigation' is introduced because of the purpose of experimentation.

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

Experiments in general-sum domain

- Tightly coupled navigation



The tightly coupled navigation domain

- 4*3 grid world
- The values in the lower left corners of each cell is the reward to agent 1 for reaching the state corresponding to that cell. (The upper right corner is for agent 2)
- Two agents are tightly coupled as they must always reach in the same cell if they want to get the reward
- Three moving actions: up, down & right

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

Experiments in general-sum domain

A realistic scenario

- Two men carrying a piece of heavy furniture
- Furniture moves in a given direction if both the agents move in that direction
- Moves not coordinated by explicit communication, but each man observes the moves and the subsequent situation of the other.
- Since they are tightly coupled, they must strike a compromise and find an intermediate path that both can be maximally satisfied with.

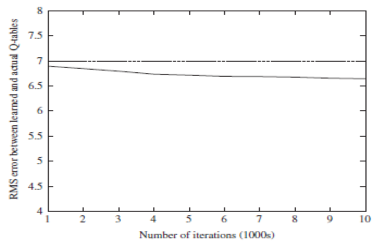
Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

Experiments in general-sum domain

- Minimax-Q vs. Minimax-SARSA
- For each iteration (from start state to goal state)
  - Setting the exploration probabilities for the agents: 0.2
  - Varying the probability of reward-generation from 0 to 0.5 to 1.0
- Exact minimax action values computed off-line
- Plot of average RMS deviation of the learned action values for every 1000 training-iterations
- Total training of 10, 000 iterations.

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

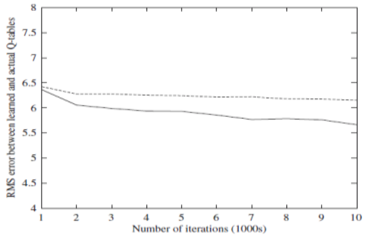
Experiments in general-sum domain



Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 0.

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

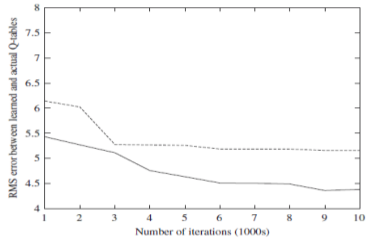
Experiments in general-sum domain



Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 0.5.

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

Experiments in general-sum domain



Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 1.0.

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

Experiments in general-sum domain

Result Statistics:

- The minimax-SARSA algorithm always approaches minimax values faster than the ordinary minimax-Q algorithm
- Error in all cases decreases monotonically, suggesting that both algorithms will eventually converge
- Error-levels fall with increasing probability of reward-generation

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

Conclusions

- Both the SARSA and Q(λ) versions of minimax-Q learn better policies early on than Littman's minimax-Q algorithm
- A combination of minimax-SARSA and Q(λ), minimax-SARSA(λ), would probably be more efficient than either of the two, by naturally combining their disjoint areas of expediency

Introduction | Multi-agent Q-learning | On-policy Q-learning | Competitive Domain | General Sum | Conclusions

Praises

- SARSA techniques (minimax and Nash) pave the way for Q-learning in large multiagent environments where maintaining table-based representations is intractable
- For time-dependent applications, the ability of minimax Q(λ) to learn good policies early on is a clear advantage

Critiques

- Only symmetric training used for both domain types (match-up between two agents of the same type)
- Convergence proof for minimax-Q(λ) not given; eventual restrictions for convergence not mentioned
- For general sum, minimax Q(λ) not used to compare with ordinary minimax and minimax-SARSA
- Comparison between ordinary Nash method and Nash-SARSA not given
- Time-dependent applications in general-sum domains could be well served by Nash-Q(λ) but this method is not developed in this paper

References

- Bikramjit Banerjee, Sandip Sen and Jing Peng, **"On-policy concurrent reinforcement learning"**, Journal of Experimental and Theoretical Artificial Intelligence, Vol. 16, No. 4, 2004
- Michael Littman, **"Markov games as a framework for multiagent reinforcement learning"**, retrieved from http://students.cs.byu.edu/~cs670ta/Fall2009/Minimax/QLearning.pdf

Questions?