Reinforcement Learning

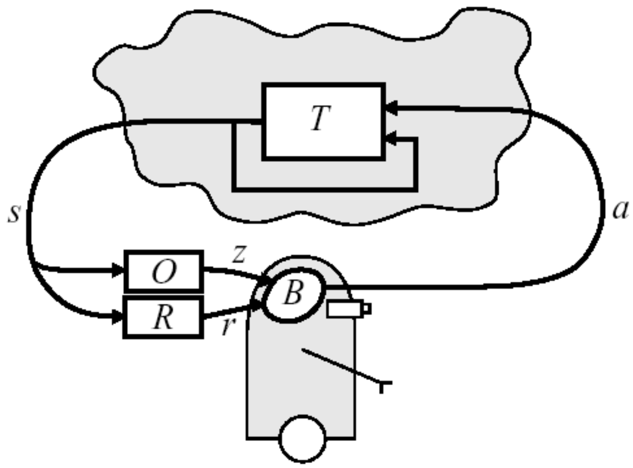
Yijia Zhao (yz4k@virginia.edu)

October 20, 2004

What Is Reinforcement Learning?

- Humans learn from interactions with the environment
- RL in AI addresses the question of how an autonomous agent learns to choose optimal actions through trial and error interactions
- Different from supervised learning

Standard RL Model



- T - the environment
- O - some observation of s, the current state of T
- R - reward from being in state s after choosing an action a
- Goal - learning to choose actions that maximize rewards

Example: Tic-Tac-Toe

- Game theory approach such as minimax requires all players to make the best moves throughout the game, which is not the case in practice.

Tic-Tac-Toe, RL Style

- $V(i)$ - latest estimate of probability of winning from that state i
- Most of the time find i with the largest $V(i)$ and choose the move that will lead to i
- Sometimes randomly choose a move
- Update $V(i)$ as we go
- $V(i)$ converges to the true probability of winning from state i

Single-agent RL Formulation

- Modeled as a Markov decision problem
- S - the set of game states
- A - the set of actions available
- A reward function $R : S \times A \rightarrow \mathcal{R}$
- A state transition function $T : S \times A \times S \rightarrow [0, 1]$, $T(s, a, s')$ is the probability of going from s to s' via action a
- A policy π is a deterministic function from S to A

Single-agent RL Formulation Cont.

- Let $\gamma \in [0, 1]$ be a discount rate
- Define value function

$$V^\pi(s) = \sum_{i=0}^{\infty} \gamma^i E_\pi[r_{t+i} | s_t = s]$$

the expected total discounted reward starting at state s following policy π

- We want the agent to learn an optimal policy π^* s.t.

$$\pi^* = \arg\max_\pi V^\pi(s) \quad \forall s \in S$$

- Write V^* instead of V^{π^*}

Bellman Equation and Dynamic Programming

- Bellman equation for value functions

$$
\begin{aligned}
V^*(s) &= \max_{\pi} E[r_t + \gamma \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} | s_t = s] \\
&= \max_{a} \{R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')\}
\end{aligned}
$$

- Bellman showed that π is optimal if and only if it satisfies the Bellman equation for all $s \in S$
- DL algorithms iteratively approximate value functions by updating the value functions with Bellman equations
- Drawbacks

Temporal-Difference Learning

- Temporal differences: adjust $V(s_t)$ for state s using k steps of observed rewards, followed by estimated value function for the final visited state

$$TD(1): \Delta V(s_t) = \alpha(r_t + r_{t+1} + \ldots + r_{t+k} + V(s_{t+k}) - V(s_t))$$

- Alternatively, we can update after just one step

$$TD(0): \Delta V(s_t) = \alpha(r_t + V(s_{t+1}) - V(s_t))$$

- $TD(1)$ is the telescoping sum of $TD(0)$'s!

TD(λ)

- For every state s

$$\Delta V(u) = \alpha(r + V(s') - V(s))e(u)$$

  Eligibility $e(s) = \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \delta(s, s_k)$ where $\delta(s, s_k) = 1$ if $s = s_k$ and 0 otherwise

- Set λ to 0, only the most recently visited state is updated
- Set λ to 1, all visited states are updated
- Best learning occurs at intermediate values

Action Value Function Q

- Define action value function

$$Q^{\pi}(s,a) = \sum_{i=0}^{\infty} \gamma^i E_{\pi}[r_{t+i} | s_t = s, a_t = a]$$

  the expected discounted cumulative reward starting at state s, taking action a, and following policy π thereafter

- Given $Q^*(s,a)$, π^* can be obtained by identifying the action that maximizes $Q^*(s,a)$

- Bellman equation for Q:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a' \in A} Q^*(s',a')$$

Q-learning (Watkins 1989)

- The most popular form of TD-learning
- Approximates $Q(s, a)$ instead of $V(s)$
- For 1-step Q-learning, the temporal difference is defined as

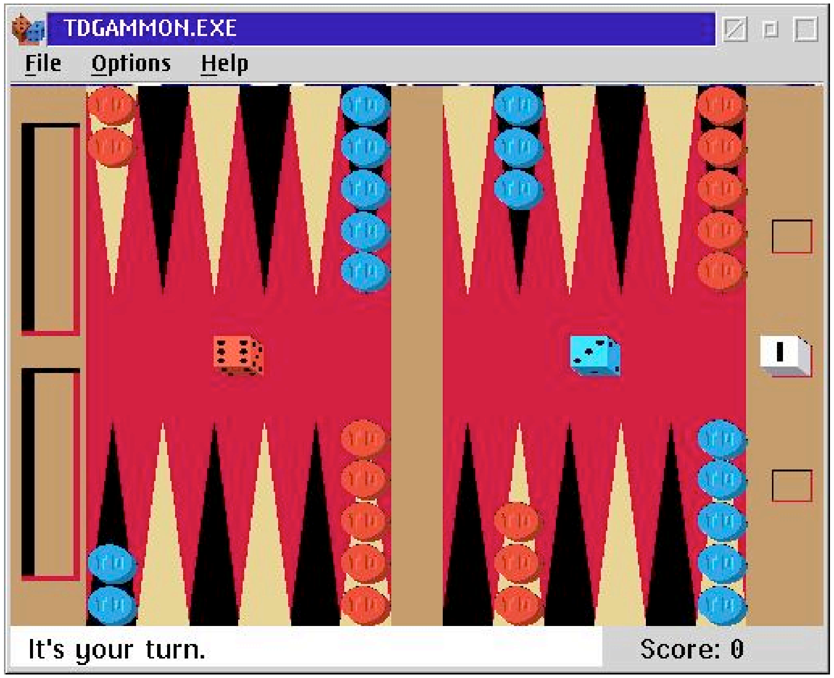
$$\Delta Q(s_t, a_t) = \alpha(r_t + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

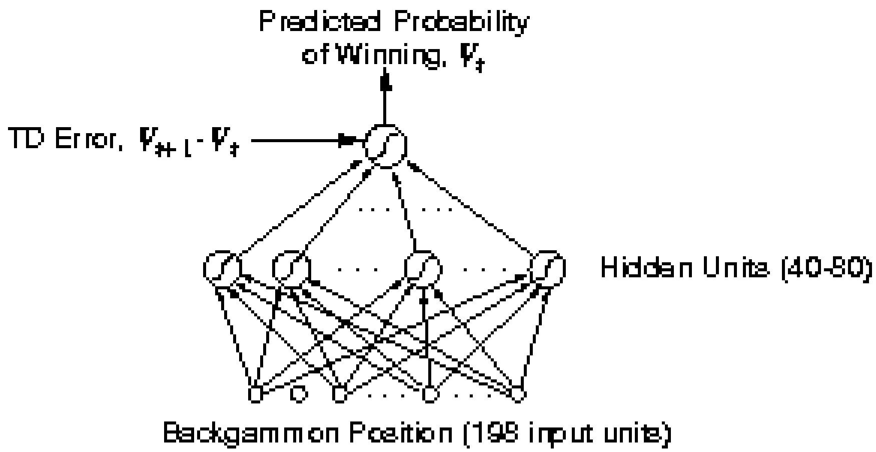
- Q-learning converges if learning rate decayed appropriately and all states and actions have been visited infinitely often (Watkins & Dayan 92; Tsitsiklis 94; Jaakkola et al. 94; Singh et al. 00)

The Exploration-Exploitation Tradeoff

- Explore - act to reveal information about environment
- Exploit - act to gain high expected reward
- Q-learning convergence depends only on infinite exploration, *i.e.* can't be greedy all the time
- Choose random action with prob $1 - \epsilon$
- Q-learning converges with any choice of ϵ

TD-Gammon (Gerry Tesauro)





Multi-agent Reinforcement Learning

- Single-agent RL treats other agents in the system as part of the environment
- Multi-agent RL explicitly takes other agents into consideration
- Use Markov games (stochastic games) as the theoretical framework
- Adopt solution concept from game theory
- Want all agents to learn policies such that their joint actions would eventually lead to an optimal solution

Two-Player Zero-Sum Markov Games

- Two agents with diametrically opposed goals
- A single reward function $R : S \times A \times O \rightarrow \mathcal{R}$, where O is the action set of the opponent
- A transition function $T : S \times A \times O \times S \rightarrow [0, 1]$
- Adopt the minimax strategy, *i.e.* behave so as to maximize the reward in the worst case
- An optimal policy π is defined as

$$\pi : S \times A \rightarrow [0, 1]$$

Minimax-Q Learning (Littman 1994)

- $Q^\pi(s, a, o)$ is the expected total discounted reward starting at state s, taking action a, the opponent chooses o, and following π thereafter
- $V^\pi(s)$ is the expected total discounted reward starting at state s and follow π thereafter

$$V^\pi(s) = \max_{\pi} \min_{o \in O} \sum_{a \in A} Q^\pi(s, a, o)\pi(s, a)$$

- Bellman equation

$$Q^\pi(s, a, o) = R(s, a, o) + \gamma \sum_{s' \in S} T(s, a, o, s')V^\pi(s')$$

Minimax-Q Learning Algorithm

Let $t = 1$
for all $s \in S, a \in A, o \in O$ **do**
  $Q(s, a, o) = 1$
  $V(s) = 1$
  $\pi(s, a) = \frac{1}{|A|}$
end for
loop
  With probability e, return an action uniformly at random
  Otherwise, if current state is s, return action a with $\pi(s, a)$
  Let s' be the next state via a and o the opponent's action
  $Q(s, a, o) = Q(s, a, o) + \alpha(R(s, a, o) + \gamma V(s') - Q(s, a, o))$
  Use linear programming to find

$$\pi(s, \cdot) = \arg\max_{\pi'(s,\cdot)} \min_{o' \in O} \sum_{a' \in A} \pi'(s, a')Q(s, a', o')$$

$$V(s) = \min o' \in O \sum_{a' \in A} \pi(s, a')Q(s, a', o')$$

  $\alpha = \alpha \times d$
  $t = t + 1$
end loop

General-Sum Markov Games

- N - the set of n agents
- S - the finite set of game states
- A^i - the action space of player i
- A transition function $T : S \times A^1 \times \ldots \times A^n \times S \rightarrow [0, 1]$
- Define a reward function for each agent $R^i : S \times A^1 \times \ldots \times A^n \rightarrow \mathcal{R}$
- π^i is the undeterministic policy of player i, $\pi^i : S \times A^i \rightarrow [0, 1]$

General-Sum Markov Games Cont.

- $V^i(s, \pi^1, \ldots, \pi^n)$ is the expected total discounted reward for player i with player j following π^j for all player i

$$V^i(s, \pi^1, \ldots, \pi^n) = \sum_{k=0}^{\infty} \gamma^k E(r^i_{t+k} | \pi^1, \ldots, \pi^n, s_t = s)$$

- $(\pi^1_*, \ldots, \pi^n_*)$ is a Nash equilibrium if and only if for all $s \in S$ and $i \in N$

$$V^i(s, \pi^1_*, \ldots, \pi^n_*) \geq V^i(s, \pi^1_*, \ldots, \pi^{i-1}_*, \pi^i, \pi^{i+1}_*, \ldots, \pi^n_*)$$

for all possible π^i.

Nash Q-Function

- Agent i's Nash Q-function is the sum of agent i's current reward plus its future rewards when all agents follow a joint Nash equilibrium strategy.

$$Q^i_*(s, a^1, \ldots, a^n) = R^i(s, a^1, \ldots, a^n)$$
$$+\gamma \sum_{s' \in S} T(s, a^1, \ldots, a^n, s') V^i(s', \pi^1_*, \ldots, \pi^n_*)$$

- Rather than updating Nash Q-function based on the agent's own maximum future reward as in the single-agent case, Nash Q-learning updates with future Nash equilibrium rewards.

- Each agent i must observe the other agents' rewards

Nash Q-Learning Algorithm

Agent i's expected total reward following selected Nash equilibrium at state s' is defined as

$$NashQ_t^i(s') = \sum_{all\ joint\ action\ profiles} [R^i(s', a^1, \ldots, a^n) \prod_{i \in N} \pi^i(s', a^i)]$$

Let t = 0
for all $s \in S, a^j \in A^j, j = 1, \ldots, n$ **do**
  $Q_t^j(s, a^1, \ldots, a^n) = 0$
end for
loop
  Choose action a_t^i
  Observe $r_t^1, \ldots, r_t^n; a_t^1, \ldots, a_t^n$, and $s_{t+1} = s'$
  for all $j = 1, \ldots, n$ **do**

$$\begin{aligned} Q_{t+1}^j(s, a_t^1, \ldots, a_t^n) &= Q_t^j(s, a_t^1, \ldots, a_t^n) \\ &+\alpha_t(r_t^j + \gamma NashQ_t^j(s') - Q_t^j(s, a_t^1, \ldots, a_t^n)) \end{aligned}$$

  end for
  $t = t + 1$
end loop

Convergence Requirements For Nash Q-Learning

- $(Q_t^1, \ldots, Q_t^n)$ converges to $(Q_*^1, \ldots, Q_*^n)$ if every state and action have been visited infinitely often and the learning rate α_t satisfies

  1. $0 \leq \alpha_t(s, a^1, \ldots, a^n) < 1$, $\sum_{t=0}^{\infty} \alpha_t(s, a^1, \ldots, a^n) = \infty$, $\sum_{t=0}^{\infty}(\alpha_t(s, a^1, \ldots, a^n))^2 < \infty$, and the latter two hold uniformly and with probability 1
  2. $\alpha_t(s, a^1, \ldots, a^n) = 0$ if $(s, a^1, \ldots, a^n) \neq (s_t, a^1, \ldots, a^n)$
  3. For every t and $s \in S$, there exists a joint action profile $(a^1, \ldots, a^n)$ such that it is a global optimal point or a saddle point for $(Q_t^1(s, \cdot), \ldots, Q_t^n(s, \cdot))$.

Limitations of Nash Q-Learning

- Aside from zero-sum or fully cooperative games, no general-sum game has been shown to satisfy convergence requirements for Nash Q-learning
- If multiple optimal NE exist, the algorithm needs an oracle to coordinate in order to converge to a NE
- Worse case running time is exponential

Other Learning Algorithms for MARL

- Friend-or-Foe-Q (Littman 2001) - divides the set of opponents into set of friends (use max) and set of foes (use minimax), easier to implement and converges to an optimal NE with less strict requirements
- Optimal Adaptive Learning (Wang & Sandholm 2002) - converges to an optimal NE for all Markov games where each agent receives the same expected reward (team Markov games)
- Polynomial Convergence (Brafman & Tennenholtz 2003) - showed that there exists a polynomial convergence algorithm for team Markov games

Open Questions in MARL

- A more widely applicable learning algorithm for general-sum Markov games
- Is NE the best solution concept?