Multiagent Reinforcement Learning

Table of contents

Model-free Reinforcement Learning

Markov Decision Processes

- a probabilistic finite state automaton
- each transition (s, a, s') has a reward $R^a_{s,s'}$
- $V^\pi(s)$ – expected discounted sum of rewards starting from state s
- $Q^\pi(s,\ a)$ – action value of a state under a policy π: expected discounted sum of rewards after action a from state s
- Bellman optimality equations:

$$V(s) = \max_{a \in A} \left(\sum_{s'} P^a_{s,s'} [R^a_{s,s'} + \gamma V(s')] \right)$$

- Reinforcement Learning (RL) approximates V without prior knowledge of probabilities and rewards

Temporal Difference Learning: Sarsa

- (a prediction method) subsequent predictions are correlated
- estimates both the expected value and the true V in

$$V^{\pi}(s) = E_{\pi}\left\{ \sum_{\tau=t+1} R_{\tau} \middle| s_t = s \right\} = E_{\pi}\{R_{t+1} + \gamma V^{\pi}(s_{t+1}) \,|s_t = s\}$$

- to find the action we need rather (state, action) pairs:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

- an ε-greedy strategy can update $\pi(s) \leftarrow \text{argmax}_a Q(s, a)$, but still choose random action with prob. ε (to assure exploration)

Q-Learning

- one-step Q-Learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \Big]$$

- aprpoximates the optimal action value Q regardless of the policy used
- enough that the policy assures exploration

RL in Homogeneous Multi-Agent Systems

...by Dynamically Merging Markov Decision Processes

"A Multiagent Reinforcement Learning by Dynamically Merging Markov Decision Processes", student paper by M. Ghavamzadeh and S. Mahadevan,

- when multiple similar agents work in an environment, optimal behavior is not automatically defined
- applying Q-Learning not efficient: state and action spaces grow exponentially with the number of agents
- use individually learned agent knowledge to find a multiagent solution (coordination skills)
- assume that decision-making is synchronous
- assume total reward = summation of individual rewards
- MultiAgent Policy LEarning (MAPLE) based on Temporal Differences learning and Q-learning
- multiagent semi-Markov Decision Process (SMDP), state space and action space are product spaces

Multiagent Homogeneous Semi-MDPs

- hierarchical methods – temporally extended actions (composite actions, closed-loop policies, micro-plans)
- options (a framework for extended actions): $o = (\pi, \beta, I)$
  - a policy $\pi: S \times A \to [0, 1]$,
  - a termination condition $\beta: S \to [0, 1]$,
  - an initiation set $I \subset S$.
- define a multiagent system with joint option set instead of joint action set
- need to define termination condition: e.g. when all options terminate
- state transitions of agents independent, reward of the system = sum of rewards

Planning algorithm

1. Initialize $\forall s \in S$:
2. $L_v(s) = \max_{i=1}^{N} V^{*,i}(s^i)$
3. $U_v(s) = \sum_{i=1}^{N} V^{*,i}(s^i)$
4. $A(s) = A$
5. Initialize s
6. **repeat**
    1. $L_v(s) \leftarrow \max_{a \in A(s)} \left(\sum_{s'} P^a_{s,s'}[R^a_{s,s'} + \gamma L_v(s')] \right)$
    2. $U_v(s) \leftarrow \max_{a \in A(s)} \left(\sum_{s'} P^a_{s,s'}[R^a_{s,s'} + \gamma U_v(s')] \right)$
    3. $A(s) \leftarrow \left\{ a \in A(s) \middle| \sum_{s'} P^a_{s,s'}[R^a_{s,s'} + \gamma U_v(s')] \geqslant \max_{b \in A(s)} \sum_{s'} P^a_{s,s'}[R^a_{s,s'} + \gamma L_v(s')] \right\}$
    4. $s \leftarrow s' \in S | \exists a \in A(s), P^a_{s,s'} > 0$
7. **until** algorithm converges

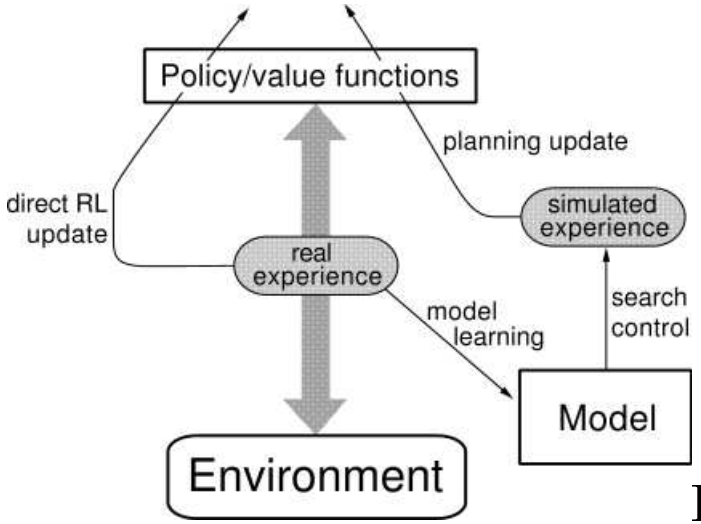
Learning Algorithm (MAPLE) [on next slide]

1. Initialize $\forall s \in S$:
2. $L_Q(s,a) = \max_{i=1}^{N} Q^{*,i}(s^i, a^i)$
3. $U_Q(s,a) = \sum_{i=1}^{N} Q^{*,i}(s^i, a^i)$
4. $A(s) = A$
5. **repeat**
    1. Initialize s
    2. **repeat**
        a. Choose a from $A(s)$ using policy derived from U_Q (e.g., ε-greedy)
        b. Take action a, observe reward R and state s'
        c. $L_Q(s,a) \leftarrow (1-\alpha)L_Q(s,a) + \alpha[R + \gamma \max_{a'} L_Q(s',a')]$
        d. $U_Q(s,a) \leftarrow (1-\alpha)L_Q(s,a) + \alpha[R + \gamma \max_{a'} L_Q(s',a')]$
        e. $A(s) \leftarrow \left\{ a \in A(s) \middle| U_Q(s,a) \geqslant \max_{b \in A(s)} L_Q(s,b) \right\}$
        f. $s \leftarrow s'$
    3. **until** s is terminal
6. **until** algorithm converges

Model-based Reinforcement Learning

"Reinforcement Learning: An Introduction", Richard S. Sutton and Andrew G. Barto 1997, Chapter 9

- model-based RL is like state-space planning; tries to approximate optimal value function for a current model with limited computation
- model $\rightarrow$ simulated experience $\overset{\text{backups}}{\longrightarrow}$ values $\rightarrow$ policy
- to enhance exploration, esp. with changing environment, add $\kappa\sqrt{n}$ to the reward of state that was visited n steps ago
- when planning, explore states that are expected to change value (and their predecessors)



Dyna Architecture

Prioritized Sweeping

Initialize $Q(s,a)$, Model(s,a) for all $s \in S, a \in A(s)$ and PQueue to empty. Loop:

1. $s \leftarrow$ current (nonterminal) state
2. $a \leftarrow$ policy(s,Q)
3. Execute action a; observe s', R
4. Model$(s,a) \leftarrow s', R$ (assume deterministic environment)
5. $p \leftarrow |R + \gamma \max_{a'} Q(s',a') - Q(s,a)|$
6. if $p > \theta$, then insert s,a into PQueue with priority p
7. Repeat N times, while PQueue is not empty:
    a. $s,a \leftarrow$ first(PQueue)
    b. $s', R \leftarrow$ Model(s,a)
    c. $Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    d. Repeat, for all $\bar{s}, \bar{a}$ predicted to lead to s:
        i. $\bar{R} \leftarrow$ predicted reward
        ii. $p \leftarrow \left|\bar{R} + \gamma \max_a Q(s,a) - Q(\bar{s},\bar{a})\right|$
        iii. if $p > \theta$ then insert $\bar{s}, \bar{a}$ into PQueue with priority p

More on Model-based RL

- full backup (like in dynamic programming):

$$Q(s,a) \leftarrow \sum_{s'} \hat{P}^a_{s,s'} \Big[\hat{R}^a_{s,s'} + \gamma \max_{a'} Q(s',a') \Big]$$

  vs. sample backup (like in Q-learning):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \Big[\hat{R}^a_{s,s'} + \gamma \max_{a'} Q(s',a') - Q(s,a) \Big]$$

- where eligibility methods update more backwards than one-step Q-learning, heuristic methods search more forwards to select better next action (the maximization goes deeper down the state-tree)

Bayesian Exploration

"Model based Bayesian Exploration", R. Dearden, N. Friedman, D. Andre 1999.

- Dirichlet priors for multinomial distributions (ξ is domain assumptions)

$$P(X^{N+1}=i|x^1,...,x^N,\xi)=\frac{\alpha_i+N_i}{\sum_j(\alpha_j+N_j)}$$

  where N_j is $\#\{x^k=j|k=1,...,N\}$.

- Priors for sparse multinomial distributions:

$$P(X^{N+1}=i|D)=\begin{cases}\frac{\alpha+N_i}{k^o\alpha+N}C(D,L) & \text{if } i\in\Sigma^o\\ \frac{1}{n-k^o}(1-C(D,L)) & \text{if } i\notin\Sigma^o\end{cases}$$

  where $C(D, L)$ is a scaling factor computed from a prior over the sizes of the set of feasible observations, and k^o is the number of observed symbols.

- Assumption of parameter independence:

$$\Pr(\theta|\mu)=\prod_s\prod_a\Pr(\theta^t_{s,a}|\mu)\Pr(\theta^R_{s,a}|\mu)$$

  is not destroyed by new evidence.

Q-Value distribution: exploiting the uncertainty

- value of information: balance expected gains from exploration against expected cost of suboptimal action
- suppose $q_{s,a}$ is the current estimate and $q^*_{s,a}$ is the true value of $Q(s, a)$, a_1 is the best action under q and a_2 is the second-best, then:

$$\text{Gain}_{s,a}(q^*_{s,a}) = \begin{cases} E[q_{s,a_2}] - q^*_{s,a} & \text{if } a = a_1, q^*_{s,a} < E[q_{s,a_2}] \\ q^*_{s,a} - E[q_{s,a_1}] & \text{if } a \neq a_1, q^*_{s,a} > E[q_{s,a_1}] \\ 0 & \text{otherwise} \end{cases}$$

- value of perfect information:

$$\text{VPI}(s,a) = \int_{-\infty}^{\infty} \text{Gain}_{s,a}(x)\Pr(q_{s,a} = x)dx$$

- choose the action that maximizes:

$$E[q_{s,a}] + \text{VPI}(s,a)$$

Estimating Q-Value Distributions

- sample several MDPs and compute sample Q-values from them by dynamic programming
- computing Q-values is expensive; keep weights for Q-samples and update weights when updating MDPs distribution instead of resampling; for a sample MDP M from distribution μ, when observed $\langle s, a, r, t \rangle$:

$$w_{\mu \circ \langle s,a,r,t \rangle} = \frac{\Pr(M \,|\, \mu \circ \langle s, a, r, t \rangle)}{\Pr(M \,|\, \mu)} w_\mu = \frac{\Pr(\langle s, a, r, t \rangle | M)}{\Pr(\langle s, a, r, t \rangle | \mu)} w_\mu$$

- for initial weights k, when the weights drop below $k_{\min}$, resample $k - k_{\min}$ new samples
- don't resample whole MDPs: "repair" – resample only around s, a, where the distribution changed; then correct the Q-value by generalized prioritized sweeping (performs several Bellman updates on states affected by the change, accounts for approximate settings)
- alternative approach is to represent distribution locally for each point s, a
- generalization: use the samples to estimate a distribution from a class of distributions (important esp. for the alternative approach: improves much)

Reinforcement Learning with Imitation

Heterogeneous Objectives but Homogeneous Actions

"Reinforcement Learning with Imitation in Heterogeneous Multi-Agent Systems", Bob Price and Craig Boutilier, 1999

- an IQ-learner uses the observations it has made of an expert agent to bias its exploration in promising directions
- no direct communication between the mentor (agent being imitated) and the observer or apprentice (agent imitating the other)
- agents may be heterogenous: may have different objectives and abilities
- the apprentice cannot directly observe the actions taken by the mentor, only the results
- assumption: the apprentice can evaluate its reward function at states the mentor has visited, but that it hasn't
- the apprentice projects its reward function on a trajectory it has observed the mentor follow, assuming it could follow this trajectory
- if the apprentice cannot follow the trajectory and agents are deterministic, the apprentice can sometimes fall in a loop

Augmented Bellman Equation

- two assumptions: the mentor implements a deterministic stationary policy π_m, which induces a Markov chain $\Pr_m(t|s) = \Pr_m(t|s, \pi_m(s))$
- for each action taken by the mentor, there exists an action $a \in A_o$ such that the distributions $\Pr_o(\cdot\,|s, a)$ and $\Pr_m(\cdot\,|s)$ are the same
- then the Bellman Equation can be rewritten as

$$V(s) = R_o(s) + \gamma \left\{ \max_{a \in A_o} \left\{ \sum_{t \in S} \Pr_o(t|s, a) V(t) \right\}, \sum_{t \in S} \Pr_m(t|s) V(t) \right\}$$

- compute model confidence e.g. by interval estimation, compare with the bottom of interval for $\Pr_m(t|s)$ to discredit regions not visited often by the mentor

Model-free IQ-Learning Algorithm

- instead of learning directly the model for MDP of the mentor, learn separate estimated state value for trajectory of the mentor

$$\hat{V}(s) \leftarrow (1-\alpha)\hat{V}(s) + \alpha\Big(R(s) + \beta^{\omega}\hat{V}(t)\Big)$$

  where ω reflects the ability of apprentice to duplicate the actions.

- when an apprentice makes a transition $\langle s,\ a,\ r,\ t\rangle$, in addition to Q-Learning it applies

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \beta\begin{cases} \hat{V}(t) & \text{if } \hat{V} \text{ visited } t \\ \max_{a'}\{Q(t,a')\} & \text{otherwise} \end{cases}\right)$$

- to diminish imitation with time, introduce $\gamma(t) = \frac{n-t}{n}$ for $t \leqslant n$; putting it together

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \beta \mathrm{FV}(t)), \text{ where}$$

$$\mathrm{FV}(t) = (1-\gamma)\max_{a}\{Q(t,a)\} + \gamma\begin{cases} \hat{V}(t) & \text{if } \hat{V} \text{ visited t} \\ \max_{a}\{Q(t,a)\} & \text{otherwise} \end{cases}$$

Bayesian Approach to Imitation

"A Bayesian Approach to Imitation in Reinforcement Learning" Bob Price and Craig Boutilier, 2003

- model-based RL method: the agent maintains an estimated MDP $\left\langle S, A_o, \widehat{R_o}, \hat{D} \right\rangle$ based on the set of experiences $\langle s, a, r, t \rangle$ obtained so far
- assume mentor has a stationary policy π_m
- Let H_o, A_o denote observer's state and action history, and H_m mentor's state history,

$$P(D|H_o, A_o, H_m) = \alpha \Pr(H_o|D, A_o) \Pr(H_m|D) P(D)$$

- in the case mentor's action in state s, π_m^s, is the same as observer's,

$$P(D^{s,a}|H_o^{s,a}, H_m^s, \pi_m^s = a) = \alpha \Pr(H_o^{s,a}|D^{s,a}) \Pr(H_m^s|D^{s,a}, \pi_m^s = a) P(D^{s,a})$$

- let $\boldsymbol{n}^{s,a}$ be the prior parameters for $P(D^{s,a})$, and $\boldsymbol{c}_o^{s,a}$ be the counts of observer transitions, and $\boldsymbol{c}_m^s$ be the counts of mentor transitions; then

$$\begin{aligned} &P(D^{s,a}|H_o^{s,a}, H_m^s) = \\ &= \Pr(\pi_m^s = a|H_o^{s,a}, H_m^s) P(D^{s,a}; \boldsymbol{n}^{s,a} + \boldsymbol{c}_o^{s,a} + \boldsymbol{c}_m^s) \\ &+ \Pr(\pi_m^s \neq a|H_o^{s,a}, H_m^s) P(D^{s,a}; \boldsymbol{n}^{s,a} + \boldsymbol{c}_o^{s,a}) \end{aligned}$$

- Update beliefs about mentor's policy:

$$\begin{aligned}
&\Pr(\pi_m|H_m, H_o) = \\
&= \alpha \Pr(H_m|\pi_m, H_o)\Pr(\pi_m|H_o) \\
&= \alpha \Pr(\pi_m) \int_{D \in \mathcal{D}} \Pr(H_m|\pi_m, D) P(D|H_o)
\end{aligned}$$

  The integral can be approximated by sampling models. We make parameter independence assumptions for the prior $\Pr(\pi_m)$.

- action selection respecting the "value of information"

Imitation with Heterogeneous Actions

Feasibility testing for agent actions

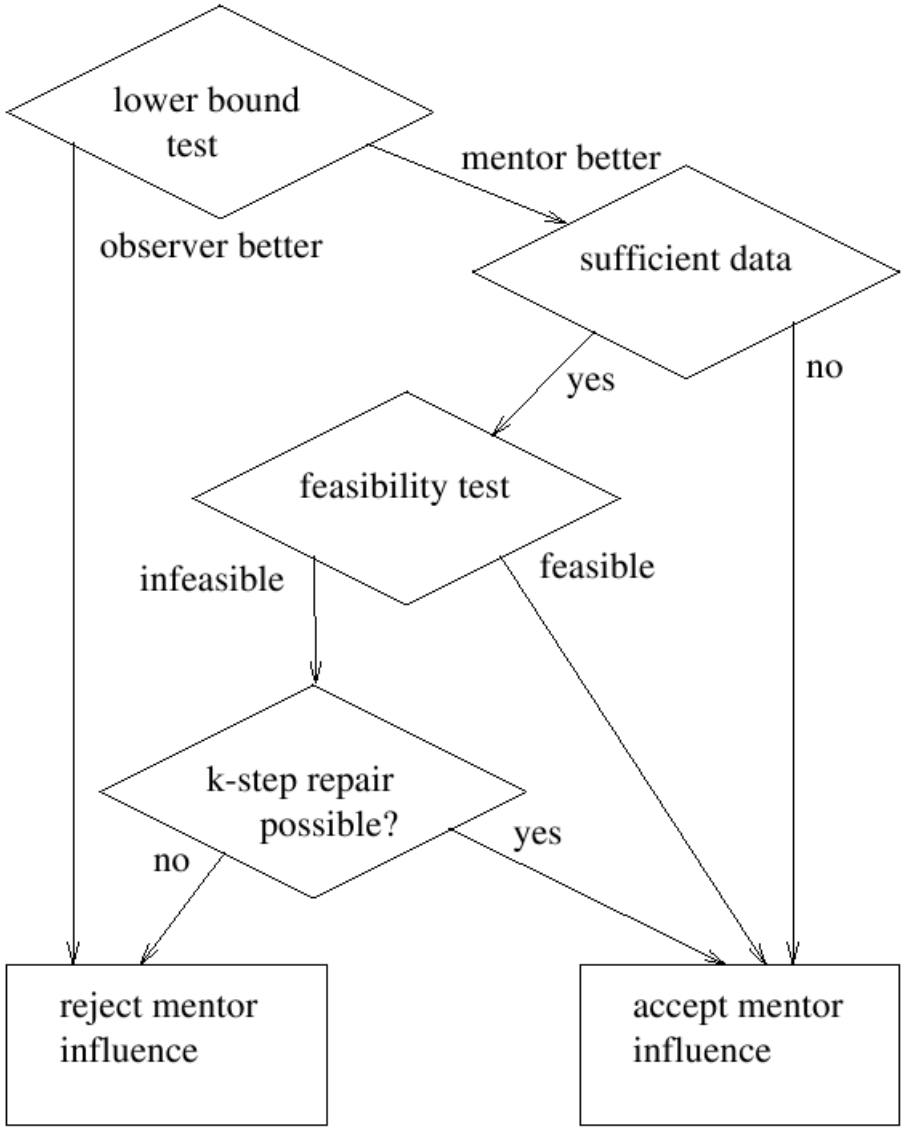
"Imitation and Reinforcement Learning in Agents with Heterogeneous Actions", Bob Price and Craig Boutilier, 2000

- action feasibility testing allows the learner to determine whether a specific mentor action can be duplicated
    - at a state, for each observer action, test if it gives the same transition probability as that observed in mentor from the state; if none observer action similar, unfeasible mentor action
- emergent ways that bypass (bridge) infeasible transitions $s \nrightarrow t$:
    - drop of the value of infeasible state t together with a small initial prior on states forces exploration of neighborhood of s

  but the effects of discounting and small priors: only short bridges explored

Feasibility testing for agent actions

- k-step repair, a learner attempts to determine whether it can approximate the mentor's trajectory
    - uses reachability analysis, caches the existence of a bridge
    - when infeasible state found, observer searches for a bridge of length at most k, following only feasible transitions
    - when a bridge is found, mentor's influence is ignored, because the value should flow through the existing bridge
    - the influence of infeasible transition is held for a fixed number of bridge-search trials (state re-entries), then suppressed thereafter
    - k-step repairability can measure the similarity of agents and be used to decide when repair attempts are worthwhile
    - feasibility test and k-step repair can be combined



Implicit imitation with feasibility tests. "k-step repair possible" here means that further repair attempts are possible (e.g. no bridge found so far, not too many attempts).

Imitation for Interacting Agents

"Accelerating Reinforcement Learning Through Imitation", Robert Roy Price, PhD. dissertation, 2002.

- learn in the joint state space (product of states of agents)
- Role Swapping: in cooperative environments, the observer can use the learned mentor's policy to augment action selection and converge to group policy
- general symmetric Markov games: needs action estimation because joint actions index the payoffs matrix, which represents the joint Q-value; little work in this field; (perhaps needs homogeneous action assumption)
- action planning in partially observable environments should deal with the tradeoff of improving the visibility of the mentor