Reinforcement Learning

Model-Based Reinforcement Learning

Model-based, PAC-MDP, sample complexity, exploration/exploitation, RMAX, E3, Bayes-optimal, Bayesian RL, model learning

Vien Ngo
MLR, University of Stuttgart
Outline

- Model-Based RL
- Exploration/Exploitation
- PAC-MDP
- $E^3$, RMAX
- Bayes optimal: Bayesian model-based RL
RL Approaches

experience data
\[ D = \{(s, a, r, s')_t\}_{t=0}^T \]

- learn model
  \[ P(s'|s, a), \quad R(s, a) \]
  - dynamic prog.
    \[ V(s) \]
    - policy
      \[ \pi(s) \]

- learn value fct.
  \[ V(s) \]
  - policy
    \[ \pi(s) \]

- optimize policy
  \[ \pi(s) \]

Model-based RL

Model-free RL

Policy Search
Model-based RL

- Model learning:
  Given data $D = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{H}$ estimate $P(s'|a, s)$ and $P(r|a, s)$

  - discrete state-action: $\hat{P}(s'|a, s) = \frac{\#(s', a, s)}{\#(a, s)}$
  - continuous state-action: $\hat{P}(s'|a, s) = \mathcal{N}(s' \mid \phi(s, a)^\top \beta, \Sigma)$

  estimate parameters $\beta$ (and perhaps $\Sigma$) as for regression (including non-linear features, regularization, cross-validation!)

- Planning:
  - discrete state-action: Dynamic Programming with the estimated model
  - continuous state-action: Differential Dynamic Programming, Planning-by-Inference, etc.
Exploration vs. Exploitation
Exploration: Motivation

Start: many potential paths
Exploration: Motivation

Continue using this path?
Try out alternatives?
Exploration: Motivation
Exploration: Motivation
Reinforcement learning

- In reinforcement learning (RL), the agent starts to act **without a model** of the environment.
- The agent has to **learn from its experience** what to do in order to fulfill tasks and achieve high rewards.
- RL algorithms we have seen thus far: Q-learning, TD-learning, policy gradient, ....
Reinforcement learning

- In reinforcement learning (RL), the agent starts to act **without a model** of the environment.
- The agent has to **learn from its experience** what to do to in order to fulfill tasks and achieve high rewards.
- RL algorithms we have seen thus far: Q-learning, TD-learning, policy gradient, ....

- Note the difference to the problem of adapting the behavior based on a given model (also called planning / solving an MDP / calculating optimal state and action values). This is a computational subproblem in model-based RL.
- Planning algorithms we have seen thus far: value iteration, policy iteration.
Exploration / Exploitation

- In contrast to supervised learning, in RL the *data used for learning depend on the agent*.

- Two different types of behavior:
  - *exploration*: behave with the goal to learn as much as possible
  - *exploitation*: behave with the goal of getting as much reward as possible

- Challenge in exploration: which actions will lead to the most important learning progress with respect to the goal?
  → exploration as fundamental intelligent behavior
Recall Markov Decision Processes

\[
P(s_0:T, a_0:T, r_0:T; \pi) = \\
P(s_0)P(a_0|s_0; \pi)P(r_0|a_0, s_0) \prod_{t=1}^{T} P(s_t|a_{t-1}, s_{t-1})P(a_t|s_t; \pi)P(r_t|a_t, s_t)
\]

– world’s initial state distribution \(P(s_0)\)
– world’s transition probabilities \(P(s_{t+1} | a_t, s_t)\)
– world’s reward probabilities \(P(r_t | a_t, s_t)\)
– agent’s policy \(\pi(a_t | s_t)\) (or deterministic \(a_t = \pi(s_t)\))
– discount parameter \(\gamma\) for future rewards

– two different sources of uncertainty: the world itself (not controlled by the agent) vs. the policy (controlled by the agent)
Exploration-exploitation tradeoff

- **Goal** of reinforcement learning agent: 
  maximize future rewards $E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0; \pi\right]$

- However, the agent does *not know* the transition parameters $P(s_{t+1} \mid a_t, s_t)$ and reward parameters $P(r_t \mid a_t, s_t)$ of the MDP.

- Rather, the agent needs to learn from its experience $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$ which actions will lead to high rewards.
**Exploration-exploitation tradeoff**

- **Goal** of reinforcement learning agent: maximize future rewards $E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0; \pi\right]$

- However, the agent does *not know* the transition parameters $P(s_{t+1} \mid a_t, s_t)$ and reward parameters $P(r_t \mid a_t, s_t)$ of the MDP.

- Rather, the agent needs to learn from its experience $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$ which actions will lead to high rewards.

- **Exploration-exploitation tradeoff**: Which policy $\pi(a_t \mid s_t)$ for action selection shall the agent follow so that it does not miss the high-reward states, but does not spend too much time in low-reward states, either?
  - Exploitation: Prefer actions $a_t$ which have led to reward before?
  - Exploration: Or rather take actions to learn more about the unknown MDP parameters and potentially find states with higher reward?
Notion of Optimality
Sample Complexity

- Let $M$ be an MDP with $N$ states, $K$ actions, discount factor $\gamma \in [0, 1)$ and a maximal reward $R_{max} > 0$.
- Let $A$ be an algorithm (that is, a reinforcement learning agent) that acts in the environment, resulting in $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$.
- Let $V^A_{t,M} = E[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \mid s_0, a_0, r_0 \ldots s_{t-1}, a_{t-1}, r_{t-1}, s_t]$.
- $V^*$ is the value function of the optimal policy.
- Define an accuracy threshold $\epsilon$: $\|\hat{V} - V^*\| \leq \epsilon$
• **Definition:** Let $\epsilon > 0$ be a prescribed accuracy and $\delta > 0$ be an allowed probability of failure. The expression $\eta(\epsilon, \delta, N, K, \gamma, R_{max})$ is a **sample complexity** bound for algorithm $A$ if independently of the choice of $s_0$, with probability at least $1 - \delta$, the number of timesteps such that $V_{t,M}^A < V^*(s_t) - \epsilon$ is at most $\eta(\epsilon, \delta, N, K, \gamma, R_{max})$. (Kakade, 2003)
Efficient exploration

- An algorithm with sample complexity that is polynomial in $1/\epsilon$, $\log(1/\delta)$, $N$, $K$, $1/(1 - \gamma)$, $R_{\text{max}}$ is called PAC-MDP (probably approximately correct in MDPs).
Exploration strategies
Exploration strategies

- The exploration strategy is reflected in the policy $\pi(s)$.
- In the following, assume we have estimates $\hat{Q}(s, a)$.

- **greedy** (only exploit): $\pi(s) = \arg\max_a \hat{Q}(s, a)$
  - problem: learned model not the same as the true environment
  - Without exploration, agent is likely to miss high rewards.

- **random**: choose action $a$ with probability $1/\{\#actions\}$
  - problem: ignores value estimates and thus rewards
Exploration strategies (continued)

- **\( \epsilon \)-greedy:** 
  \[ 
  \pi(s) = \begin{cases} 
  \arg\max_a \hat{Q}(s, a) & \text{with probability } 1 - \epsilon \\
  \text{random action} & \text{with probability } \epsilon 
  \end{cases} 
  \]
  
  - most popular method
  
  - Converges to the optimal value function with probability 1 (all paths will be visited sooner or later), if the exploration rate diminishes according to an appropriate schedule.
  
  - problem: *sample complexity exponential in the number of states*

- **Boltzmann:** choose action \( a \) with probability
  \[ 
  \frac{\exp(\hat{Q}(s, a)/T)}{\sum_a \exp(\hat{Q}(s, a)/T)} 
  \]
  
  - temperature \( T \) controls amount of exploration
  
  - problem again: sample complexity exponential in the number of states
Exploration strategies (continued)

- Other heuristics for exploration:
  - minimize variance of action value estimates
  - optimistic initial values ("optimism in the face of uncertainty")
  - state bonuses: frequency, recency, error etc.
  → problem again: sample complexity exponential in the number of states

- **Bayesian RL**: optimal exploration strategy
  - distribution over MDP models (i.e., the parameters of the MDP)
  - posterior distribution updated after each new observation
  - exploration strategy minimizes uncertainty of parameters
  - Bayes-optimal solution to the exploration-exploitation tradeoff (i.e., no other policy is better in expectation w.r.t. prior distribution over MDPs)
  - only tractable for very simple problems

- $E^3$ and $R^{max}$: principled approach to the exploration-exploitation tradeoff with polynomial sample complexity
PAC-MDP Algorithms

- Explicit-Exploit-or-Explore (E3)
- RMAX
PAC-MDP Algorithms

- Explicit-Exploit-or-Explore (E3)
- RMAX

- Common Instuition: optimism in the face of uncertainty (i.e. if faced the option of certain and uncertain reward regions, try the UNCERTAIN reward region)
Explicit-Exploit-or-Explore (E3) algorithm
Explicit-Exploit-or-Explore (E3) algorithm

Kearns and Singh (2002)

- Model-based approach with polynomial sample complexity (PAC-MDP)
  - uses optimism in the face of uncertainty
  - assumes knowledge of maximum reward
- Maintains *counts for states and actions* to quantify confidence in model estimates
  - A state $s$ is *known* if all actions in $s$ have been sufficiently often executed.
Explicit-Exploit-or-Explore (E3) algorithm

Kearns and Singh (2002)

- Model-based approach with polynomial sample complexity (PAC-MDP)
  - uses optimism in the face of uncertainty
  - assumes knowledge of maximum reward
- Maintains counts for states and actions to quantify confidence in model estimates
  - A state $s$ is known if all actions in $s$ have been sufficiently often executed.
- From observed data, E3 constructs two MDPs:
  - $\text{MDP}_{\text{known}}$: includes known states with (approximately exact) estimates of $P(s_{t+1} \mid a_t, s_t)$ and $P(r_t \mid a_t, s_t)$
    $\rightarrow$ model which captures what you know
  - $\text{MDP}_{\text{unknown}}$: $\text{MDP}_{\text{known}}$ + special state $s'$ where the agent receives maximum reward
    $\rightarrow$ model which drives exploration
**E3 sketch**

**Input:** State $s$

**Output:** Action $a$

1. **If $s$ is known then**
   - Plan in $\text{MDP}_{\text{known}}$  
     - Sufficiently accurate model estimates
   - If resulting plan has value above some threshold then
     - **Return** first action of plan  
     - Exploitation
   - Else
     - Plan in $\text{MDP}_{\text{unknown}}$
     - **Return** first action of plan  
     - Planned exploration
2. Else
   - **Return** action with the least observations in $s$  
   - Direct exploration
E3 example

S. Singh (Tutorial 2005)
E3 example

S. Singh (Tutorial 2005)
E3 example

$M$: true known state MDP  
$\hat{M}$: estimated known state MDP

S. Singh (Tutorial 2005)
Implementation Setting

- $T$ is the time horizon.
- $G^T_{\text{max}}$ is the maximum $T$-step return. (discounted case $G^T_{\text{max}} \leq TR_{\text{max}}$).
- A state is known if it was visited $O\left(\frac{NTG^T_{\text{max}}/\epsilon}{4Var_{\text{max}} \log(1/\delta)}\right)$ times. ($Var_{\text{max}}$ is the maximum variance of the random payoffs over all states).
- For the exploration/exploitation choice at known states: It’s assumed to be given the optimal value function $V^*$. If $\hat{V}$ obtained from the $MDP_{\text{known}} > (V^* - \epsilon)$ then do exploitation.
RMAX Algorithm
RMAX

- R-MAX solves only one unique model (don’t separate $\text{MDP}_{\text{known}}$ and $\text{MDP}_{\text{unknown}}$) and therefore implicitly explores or exploits.
- The R-MAX and E3 algorithms were able to achieve roughly the same level of performance (Strehl’s thesis).
RMAX

- R-MAX solves only one unique model (don’t separate MDP\textsubscript{known} and MDP\textsubscript{unknown}) and therefore implicitly explores or exploits.
- The R-MAX and E3 algorithms were able to achieve roughly the same level of performance (Strehl’s thesis).
- RMAX builds an approximate MDP based on reward function

\[ R(s, a) = \begin{cases} \hat{R}(s, a) & \text{if (s,a) known} \\ R_{\text{max}} & \text{otherwise} \end{cases} \]
RMAX’s sketch

Initialize all counter \( n(s, a) = 0 \), \( n(s, a, s') = 0 \).

Initialize \( \hat{T}(s'|s, a) = I_{s=s'}, \hat{R}(s, a) = R_{\text{max}} \)

while (1) do

Compute policy \( \pi_t \) using MDP model of (\( \hat{T}, \hat{R} \)).

Choose \( a = \pi_t(s) \), observe \( s', r \).

\( n(s, a) = n(s, a) + 1 \)

\( r(s, a) = r(s, a) + r \)

\( n(s, a, s') = n(s, a, s') + 1 \)

if \( n(s, a) = m \) then

Update \( \hat{T}(\cdot|s, a) = n(s, a, \cdot)/n(s, a) \), and \( \hat{R}(s, a) = r(s, a)/n(s, a) \).
RMAX Analysis

• The general PAC-MDP theorem does not easily adapt to the analysis of E3 because of E3’s use of two internal models (Original analysis depends on horizon and mixing time).

• (Upper bound) There exists $m = O\left(\frac{NT^2}{\epsilon^2} \ln^2 \frac{NK}{\delta}\right)$, then with probability of at least $1 - \delta$, $V(s_t) \geq V^*(s_t) - \epsilon$ is true for all but

$$O\left(\frac{N^2KT^3}{\epsilon^3} \ln^2 \frac{NK}{\delta}\right)$$

where $N$ is the number of states.

• For discounted case: $T = \frac{-\log \epsilon}{1-\gamma}$
Limitations of E3 and RMAX

- E3/RMAX is called “efficient” because its sample complexity scales only polynomially in the number of states.
- In natural environments, however, this number of states is enormous: it is exponential in the number of objects in the environment.
- Hence E3/RMAX scales exponentially in the number of objects.

- Generalization over states and actions is crucial for exploration.
KWIK

KWIK (Known What It Knows): a supervised-learning model
KWIK

KWIK (Known What It Knows): a supervised-learning model

- Input set: $X$
- Output set: $Y$
- Observation set: $Z$
- Hypothesis class: $H : X \rightarrow Y$
- Target function: $h^* \in H$
- Special symbol: ? (I don’t know)
Given: $\epsilon, \delta, H$

Env: Pick $h^* \in H$ secretly & adversarially

Env: Pick $x$ adversarially

Learner

Learning succeeds if

- W/ prob. $1-\delta$, all predictions are correct
  - $|\hat{y} - h^*(x)| \leq \epsilon$
- Total #? is small
  - at most $\text{poly}(1/\delta,1/\delta,\text{dim}(H))$

Observe $y=h^*(x)$ [deterministic] or measurement $z$ [stochastic where $E[z]=h^*(x)$]
Example: Coin Learning

- Predict \( Pr(\text{head}) \in [0, 1] \) for a coin
  - from many observations: head or tail

- Algorithm
  - Predict \( ? \) the first \( O(1/\epsilon^2 \log(1/\delta)) \) times
  - Use empirical estimate afterwards
  - The bound follows from Hoeffdings bound \( O(1/\epsilon^2 \log(1/\delta)) \)

Li et. al. ICML 2008.
KWIK-RMAX

- $T(s'|s, a)$ and $R(s, a)$ are TWIK-learned.
Conclusions

- RL agents need to solve the exploration-exploitation tradeoff.
- Sample complexity measures the required number of explorative actions of an algorithm.
- Ideas for driving exploration: random actions, optimism in the face of uncertainty, maximizing learning progress and information gain.
References