Lecture 11 - Planning & Learning with Tabular Methods

Instructor: GuangBing Yang, PhD

Introduction to Planning & Learning with Tabular Methods

- Introduce a unified view of RL methods that work for either model-based or model-free approaches
- Model-based methods include dynamic programming, Markov decision process, and heuristic search like multi-armed bandits.
- Model-free methods include Monte Carlo and temporal-difference methods.
- Model-based methods rely on ***planning***,
- model-free methods rely on ***learning***

Introduction to Planning & Learning with Tabular Methods

- Similarities between two:
  - Computation of value function
  - Looking ahead to future events
  - Computing backed-up value
  - Update target for an approximate value function

Models and Planning

- A model of environment is defined as what an agent can use to predict the responses of environment
- Given a state and an action, a model produces a prediction of the value of next state and next reward
- In other words, model gives the prediction of the next state and reward, and
- Agent gives actions
- If the model is stochastic, the next states and rewards present as probability of occurring

Models and Planning

- Models produce a description of all possibilities and their probabilities; these are called *distribution models*
- Models produce just one of the possibilities, sampled according to the probabilities; these are called *sample models*
- A model in dynamic programming--estimates of the MDP's dynamics, p(s', r | s, a)--is a *distribution model*
- Distribution models are stronger than sample models because they can always be used to produce samples.
- However, sample models are easier to obtain

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 6 2021

Models and Planning

- The term of planning refers to any computational process improving policy via a model of environment.
- Two different planning definitions in AI:
  - **State-space planning**
  - **Plan-space planning**

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 6 2021

Models and Planning

- There are two basic ideas:

1. all state-space planning methods need to compute value functions in order to improve the policy
2. by updating or backing up the simulated experience, compute the value functions

- The common structure is given as:
- $$\text{model} \longrightarrow \text{simulated experience} \xrightarrow{\text{backups}} \text{values} \longrightarrow \text{policy}$$

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 6 2021

Random-sample one-step tabular Q-planning

- Learning methods require only experience as input.
- Those experiences can be real ones or simulated.
- The random-sample one-step tabular Q-planning is an example of a planning method based on one-step tabular Q-learning and on random samples from a sample model.

Random-sample one-step tabular Q-planning

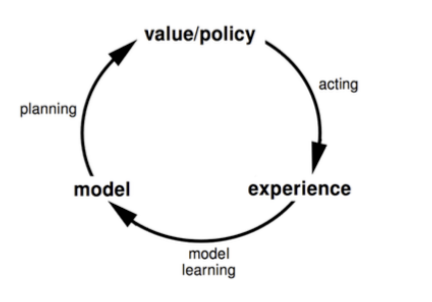
Loop forever:
1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(S)$, at random
2. Send S, A to a sample model, and obtain
   a sample next reward, R, and a sample next state, S'
3. Apply one-step tabular Q-learning to S, A, R, S':
   $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 6 2021

Model-based RL

- Previously, a policy was learnt directly from experience, as well as
- value functions
- Now, rather than learn a policy, learn a model directly from experience, and
- use planning to construct a value function or policy rather than from experience directly.
- Need a single architecture to integrate learning and planning

Model-based RL



Model-based RL architecture

Pros and Cons –Model-based RL

- Advantages:
  - efficient way to learn model by supervised learning approaches
  - a way of reasoning about model uncertainty
- Disadvantages:
  - two processes: learn a model then construct a value function based on the model
  - consequence: two sources of approximation errors

What is a Model

- A model, denoted as M, is a representation of an MDP $< S, A, P, R, \theta >$, θ is the parameters
- State and action spaces S and A are known
- So a model M = $< P, R; \theta >$ represents
  - state transitions: $S_{t+1} \sim P(S_{t+1} | S_t, A_t; \theta)$
  - rewards: $R_{t+1} = R(R_{t+1} | S_t, A_t ; \theta)$
- Based on Markov property: $P[S_{t+1}, R_{t+1} | S_t, A_t] = P[S_{t+1} | S_t, A_t]P[R_{t+1} | S_t, A_t]$

Model learning

- The purpose: estimate model M from experience {S1, A1, …, ST}
- This is actually a supervised learning problem
-
$$
\begin{aligned}
S_1, A_1 &\to R_2, S_2 \\
S_2, A_2 &\to R_3, S_3 \\
&\vdots \\
S_{T-1}, A_{T-1} &\to S_T, R_T
\end{aligned}
$$

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 6 2021

Model-Free RL

- To integrate learning and planning, only model-based approaches are not enough
- Model-free RL methods provide learning methods.
- Model-free, like the name suggested, no model provided
- Learn value function and / or policy from real experience

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 6 2021

Planning with a Model

- Given a model M = <P, R; θ>
- Solve the MDP <S, A, P, R, θ>
- Using either of planning algorithms:
  - value iteration
  - policy iteration
  - tree search
  - sample-based planning

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 6 2021

Sample-based Planning

- This is a simple but powerful approach to planning
- Use the model only to generate samples
- Sample experience form model
  - $S_{t+1} \sim P(S_{t+1} \mid S_t, A_t; \theta)$
  - $R_{t+1} = R(R_{t+1} \mid S_t, A_t ; \theta)$
- Apply model-free RL to samples, e.g:
  - Monte-Carlo control
  - Sarsa
  - Q-learning

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 6 2021

Examples of Models

- For example,
  - Table lookup model
  - Linear expectation model
  - Linear Gaussian Model
  - Gaussian Process Model
  - Deep Belief Network
  - and more …

Table Lookup Model

- It is an explicit MDP $< S, A, P, R, \theta >$.
- Denote N(s, a) as the count of visit to each state-action pair:
  - $P(a, s, s') = \dfrac{\sum_{t=1}^{T} 1(S_t = s, A_t = a, S_{t+1} = s')}{N(s, a)}$
  - $r(s, a) = \dfrac{\sum_{t=1}^{T} 1(S_t = s, A_t = a)R_t}{N(s, a)}$

Example of Table Lookup Model

- For example, having a two states A, B
- A 8 episodes of experience without discounting:

  $A,0,B,0$
  $B,1$
  $B,1$
  $B,1$
  $B,1$
  $B,1$
  $B,1$
  $B,0$

- Can construct a table lookup model based on the experiences:

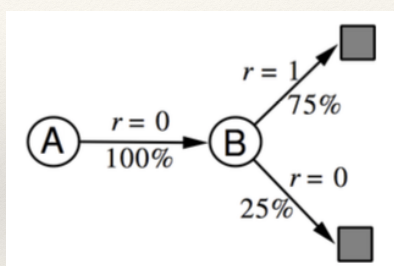


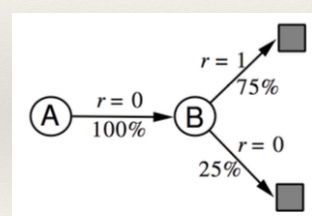
Table lookup model of two states

Example of Table Lookup Model

- Construct a table-lookup model from real experience
- Apply model-free RL to sampled experience, e.g., Monte-Carlo learning: V(A)=1, V(B)=0.75

Real experience

A,	0,	B,	0
B,	1		
B,	1		
B,	1		
B,	1		
B,	1		
B,	1		
B,	0		



Sampled experience

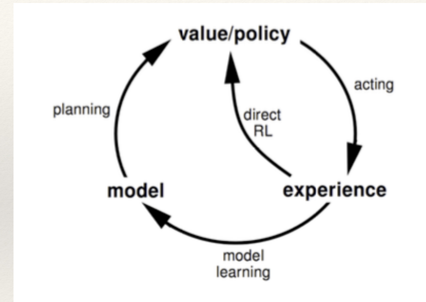
B,	1		
B,	0		
B,	1		
A,	0,	B,	1
B,	1		
A,	0,	B,	1
B,	1		
B,	0		

Issues of Planning

- If the model is not accurate enough, for instance,
- Given an inaccurate model <P, R; θ > $\neq$ <P, R>
- Model-based RL is limited to optimal policy for approximation MDP
- It is only as good as the estimated model
- the consequence of the planning based on this imperfect model is a suboptimal policy
- To solve this issue:
  - if model is wrong, use model-free RL
  - reasoning model uncertainty to increase accuracy

Dyna Architecture

- The single integrating architecture is called Dyna
- Learn a model from real experience
- Learn and plan value function and/or policy from real and simulated experience



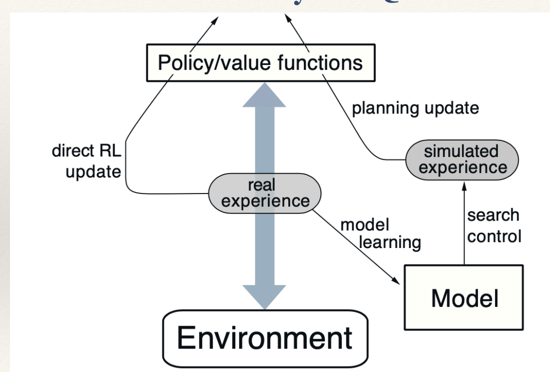
α

Dyna-Q Algorithm

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Loop repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

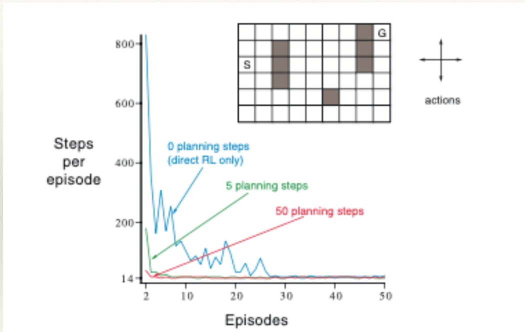
Dyna-Q Architecture



Dyna-Q Architecture

- Interaction between **agent** and **environment**
- **Direct RL** operating on real experience to improve the value function and the policy
- **Model** is learned from real experience and gives rise to simulated experience
- **Planning** is updated with the simulated experiences

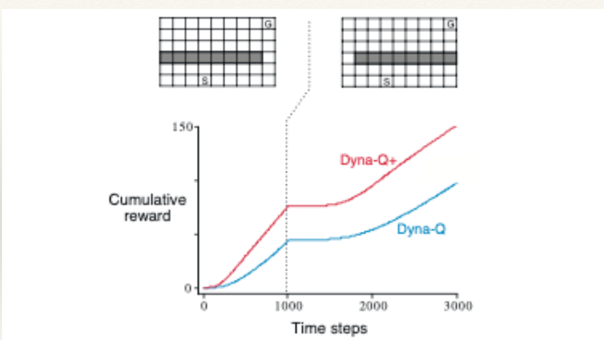
Example of Dyna-Q Algorithm



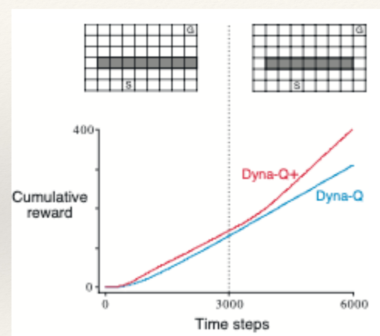
When the model is wrong

- In previous case, the model started out empty and then filled with correct information
- This ideal situations won't happen in realistic applications.
- When the model is incorrect,
  - environment is stochastic,
  - only a limited samples observed,
  - learned from an approximated function with noise and poor generalization, etc
- The planning process is likely to compute a suboptimal policy.
  - Sometimes, this suboptimal policy can lead to the discovery and correction of the model errors
  - However, there is conflict between exploration and exploitation caused by planning
- To solve this issue of Dyna-Q, a Dyna-Q+ agent uses a heuristic approach "bonus reward" is given to encourage behavior that tests long-untried actions.

Example of Dyna-Q Algorithm with an Inaccurate Model



Example of Dyna-Q Algorithm with an Inaccurate Model



Recap

- Introduce a unified view of RL methods that work for either model-based or model-free approaches
- Model-based methods rely on ***planning***,
- model-free methods rely on ***learning***
- A model of environment is defined as what an agent can use to predict the responses of environment.
- *distribution models* produce a description of all possibilities and their probabilities
- *sample models* produce just one of the possibilities, sampled according to the probabilities.
- Dyna architecture integrates planning, acting, and learning.
- Dyna-Q is a simple architecture integrating the major functions needed in an online planning agent.

Questions
The Critical Thinking Test