Theory and Methods for Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 10: Model based RL

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-618 (Spring 2020)

lions@epfl
HASLERSTIFTUNG SDSC FNSNF Google AI ZEISS
Fonds national suisse
Schweizerischer Nationalfonds
Fondo nazionale svizzero
Swiss National Science Foundation
Office of Naval Research GLOBAL Science & Technology
erc EPFL

License Information for Reinforcement Learning Slides

- This class:
  1. Model-Based RL
- Next class:
  1. Deep Model-Based RL

Recommended reading

- Chapter 8 in S. Sutton, and G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

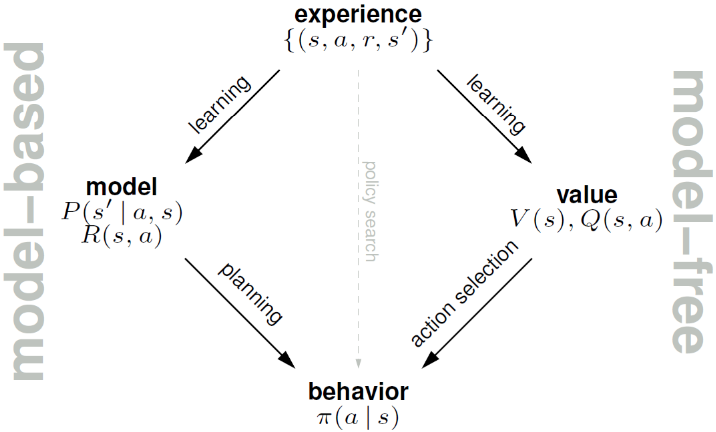
Model-Based Reinforcement Learning

- Policy based methods: learn policy directly from experience
- Value based methods: learn value function directly from experience
- Model-Based RL: learn model directly from experience

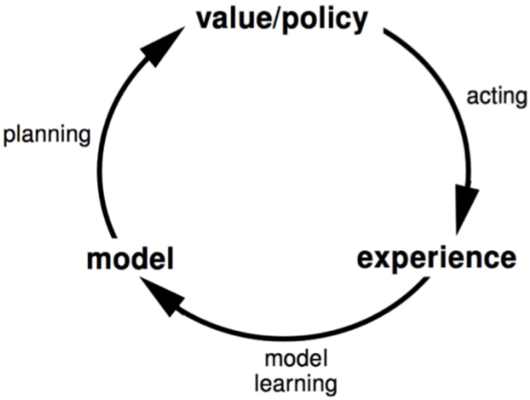
Model-Based and Model-Free RL

- Model-Free RL
  - no model
  - learn value function (and/or policy) from experience
  - e.g., Monte-Carlo and TD
- Model-Based RL
  - learn a model from experience
  - plan value function (and/or policy) from model
  - e.g., Dynamic Programming

Model-Based and Model-Free RL



Model-Based RL[1]



Advantages of Model-Based RL

- Advantages:
  - can efficiently learn model by supervised learning methods
  - can reason about model uncertainty
  - when dynamics are easy, can be more sample efficient
  - once we have the model, we can do planning at decision time
- Disadvantages:
  - first learn a model, then construct a value function
  - two sources of approximation error
  - cumulative error for long horizons

What is a Model?

- A model $\mathcal{M}$ is a representation of an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, parametrized by η
- We will assume state space $\mathcal{S}$ and action space $\mathcal{A}$ are known
- So a model $\mathcal{M}_\eta = (\mathcal{P}_\eta, \mathcal{R}_\eta)$ represents state transitions $\mathcal{P}_\eta \approx \mathcal{P}$ and rewards $\mathcal{R}_\eta \approx \mathcal{R}$:

$$\begin{aligned} S_{t+1} &\sim \mathcal{P}_\eta(S_{t+1} \mid S_t, A_t) \\ R_{t+1} &= \mathcal{R}_\eta(R_{t+1} \mid S_t, A_t) \end{aligned}$$

- Typically assume conditional independence between state transitions and rewards

$$\mathbb{P}\left[S_{t+1}, R_{t+1} \mid S_t, A_t\right] = \mathbb{P}\left[S_{t+1} \mid S_t, A_t\right] \mathbb{P}\left[R_{t+1} \mid S_t, A_t\right]$$

Model Learning

- Goal: estimate model $\mathcal{M}_\eta$ from experience $\{S_1, A_1, R_2, \ldots, S_T\}$
- This is a supervised learning problem

$$\begin{aligned} S_1, A_1 &\rightarrow R_2, S_2 \\ S_2, A_2 &\rightarrow R_3, S_3 \\ &\cdots \\ S_{T-1}, A_{T-1} &\rightarrow R_T, S_T \end{aligned}$$

- Learning $s, a \rightarrow r$ is a regression problem
- Learning $s, a \rightarrow s'$ is a density estimation problem
- Pick loss function, e.g., mean-squared error, KL divergence, etc.
- Find parameters η that minimize empirical loss

Examples of Models

- Table Lookup Model
- Linear Expectation Model
- Linear Gaussian Model
- Gaussian Process Model

Table Lookup Model

- Model is an explicit MDP, $\hat{\mathcal{P}}, \hat{\mathcal{R}}$
- Count visits $N(s,a)$ to each state action pair

$$\hat{\mathcal{P}}^a_{s,s'} = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}\left\{S_t, A_t, S_{t+1} = s, a, s'\right\}$$

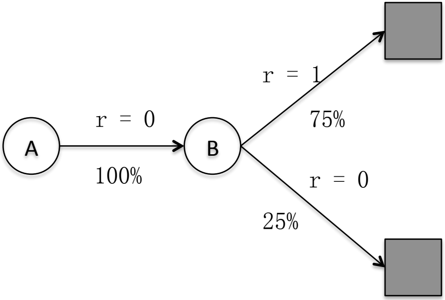
$$\hat{\mathcal{R}}^a_s = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}\left\{S_t, A_t = s, a\right\} R_t$$

- Alternatively
  - At each time-step t, record experience tuple $(S_t, A_t, R_{t+1}, S_{t+1})$
  - To sample model, randomly pick tuple matching $(s, a, \cdot, \cdot)$

AB Example

- Two states A, B; no discounting; 8 episodes of experience

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



- We have constructed a table lookup model from the experience

Planning with a Model

- Given a model $\mathcal{M}_\eta = (\mathcal{P}_\eta, \mathcal{R}_\eta)$
- Solve the MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}_\eta, \mathcal{R}_\eta)$
- Using any planning algorithm
  - Value iteration
  - Policy iteration
  - Generalized policy iteration

Sample-Based Planning

- A simple but powerful approach to planning
- Use the model only to generate samples
- Sample experience from model

$$
\begin{aligned}
S_{t+1} &\sim \mathcal{P}_\eta(S_{t+1} \mid S_t, A_t) \\
R_{t+1} &= \mathcal{R}_\eta(R_{t+1} \mid S_t, A_t)
\end{aligned}
$$

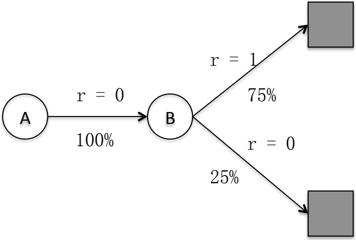
- Apply model-free RL to samples, e.g.:
  - Monte-Carlo control
  - Sarsa
  - Q-Learning
- Sample-based planning methods are often more efficient

AB Example

- Construct a table-lookup model from real experience
- Apply model-free RL to sampled experience

Real experience

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



Sampled experience

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

- e.g. Monte-Carlo learning: $V(A) = 1, V(B) = 0.75$

Planning with an Inaccurate Model

- Given an imperfect model $(\mathcal{P}_\eta, \mathcal{R}_\eta) \neq (\mathcal{P}, \mathcal{R})$

- Performance of model-based RL is limited to optimal policy for approximate MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}_\eta, \mathcal{R}_\eta)$.

- i.e. Model-based RL is only as good as the estimated model

- When the model is inaccurate, planning process will compute a suboptimal policy

- Solution 1: when model is wrong, use model-free RL

- Solution 2: reason explicitly about model uncertainty

Real and Simulated Experience

- We consider two sources of experience
- Real experience: sampled from environment (true MDP)

$$\begin{aligned} S' &\sim \mathcal{P}^{a}_{s,s'} \\ R &= \mathcal{R}^{a}_{s} \end{aligned}$$

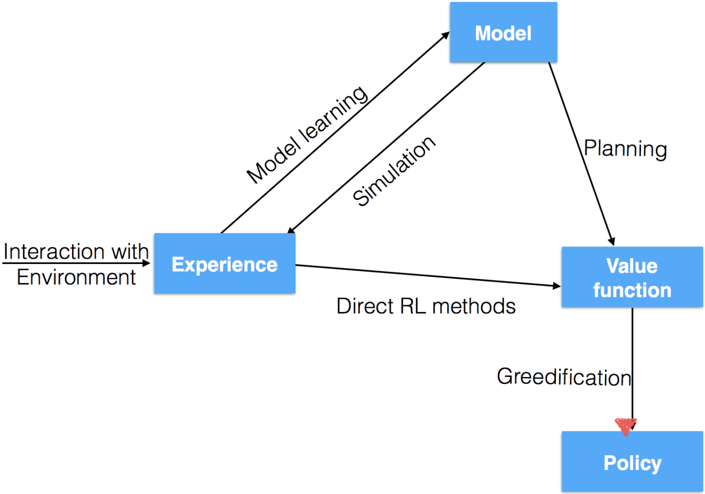
- Simulated experience: sampled from model (approximate MDP)

$$\begin{aligned} S' &\sim \mathcal{P}_{\eta}(S' \mid S, A) \\ R &= \mathcal{R}_{\eta}(R \mid S, A) \end{aligned}$$

Integrating Learning and Planning

- Model-Free RL
  - no model
  - learn value function (and/or policy) from real experience
- Model-Based RL (using Sample-Based Planning)
  - learn a model from real experience
  - plan value function (and/or policy) from simulated experience
- Dyna
  - learn a model from real experience
  - learn and plan value function (and/or policy) from real and simulated experience

Integrating Learning and Planning



Integrating Learning and Planning

• A learning algorithm can be substituted for the key update step of a planning method.

• Learning methods require only experience as input, and they can be applied to simulated experience just as well as to real experience.

Random-sample one-step tabular Q-planning

Repeat (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':

   $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$

• This method converges to the optimal policy for the model under the same conditions that one-step tabular Q-learning converges to the optimal policy for the real environment.

Dyna Architecture

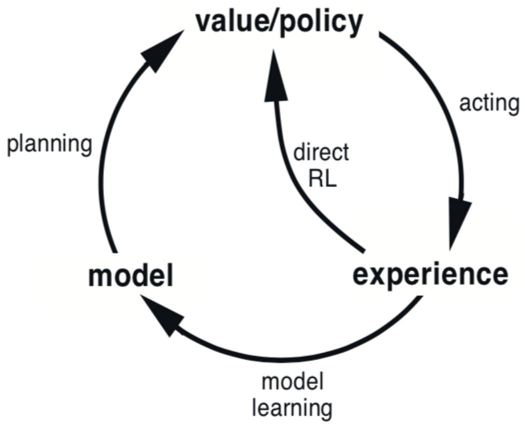


Figure: Relationship between learning, planning and acting

Dyna-Q Algorithm

Tabular Dyna-Q

Initialize $Q(s,a)$ and Model(s,a), for all $s \in \mathcal{S}, a \in \mathcal{A}$

Repeat (forever):

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \epsilon$-greedy(S, Q)

(c) Execute action A; observe resultant reward, R, and state, S'

(d) $Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

(e) $Model(S,A) \leftarrow R, S'$ (assume the environment is deterministic)

(f) **Repeat** n times:

$S \leftarrow$ random previously observed state

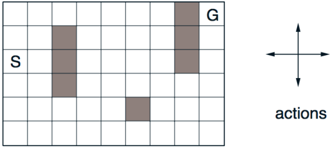
$A \leftarrow$ random action previously taken in S

$R, S' \leftarrow Model(S,A)$

$Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

Example: Dyna Maze

- Consider the maze problem with obstacles as shown below:
  - four possible (deterministic) moves: up, down, left, right.
  - reward is zero for all transitions except for transitioning to goal, which is $+1$.
  - task is episodic and discounted with $\gamma = 0.95$, step size $\alpha = 0.1$, $\epsilon = 0.1$.



Example: Dyna Maze

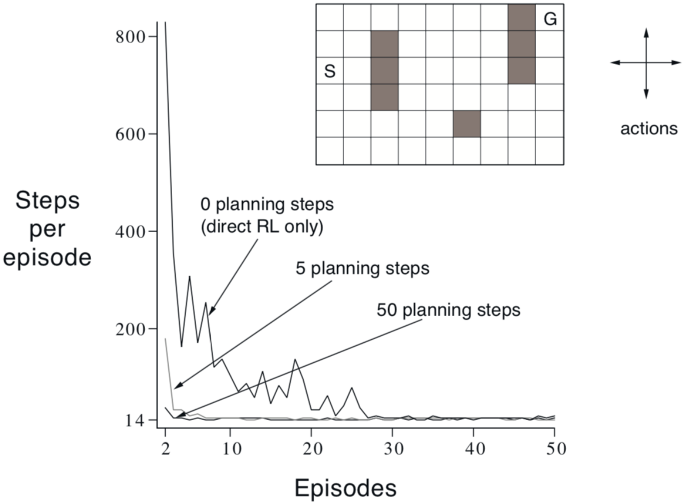


Figure: Learning curve for Dyna maze example with varying planning steps.

Example: Dyna Maze

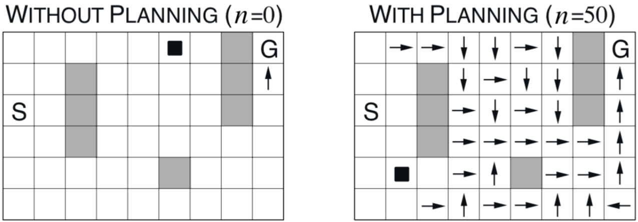


Figure: Policies for 0 planning steps and 50 planning steps

Dyna-Q with an Inaccurate Model

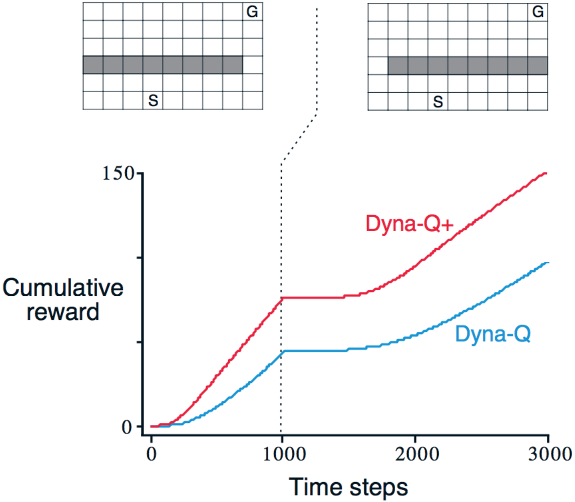


Figure: The changed environment is harder

Dyna-Q with an Inaccurate Model

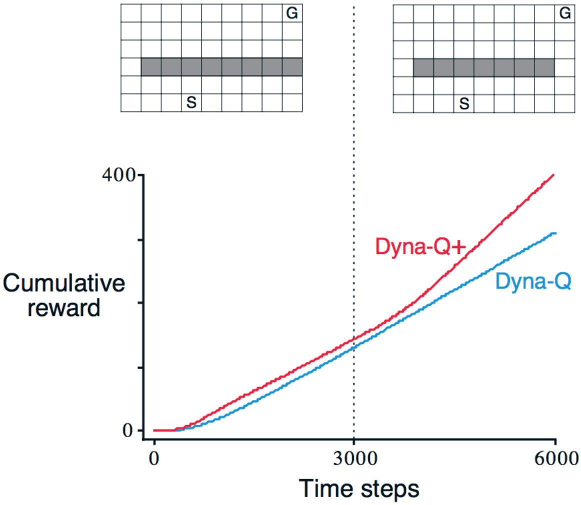


Figure: The changed environment is easier

Dyna-Q+

- Dyna-Q+ uses an additional heuristic based on exploration/exploitation:
  - for each (s, a) pair, algorithm keeps track of the time passed since their last trial.
  - bonus reward for long-untested (s, a) pairs on simulated experiences.
  - r: simulated reward for a given pair (s, a)
  - τ: time until last trial of the pair (s, a)
  - bonus reward: $r + \kappa\sqrt{\tau}$, where κ is *small*.

References

[1] Richard S Sutton and Andrew G Barto.
Reinforcement learning: An introduction, volume 2.
MIT press Cambridge, 2018.