An Introduction of Model-based RL

Xihan Li

University College London

Apr 23, 2021

Table of Contents

Basic Concepts: Model and Planning

Background Planning
- Simulated Environment: Dyna-Q
- Assist Learning Algorithm: Policy Backprop

Model-free vs Model-based RL [2]

Collecting data: $\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$

- Model-free: learn policy directly from data
  $\mathcal{D} \to \pi$, e.g. Q-learning, policy gradient
- Model-based: learn model, then use it to learn or improve a policy
  $\mathcal{D} \to f \to \pi$

Table of Contents

Basic Concepts: Model and Planning

Background Planning
- Simulated Environment: Dyna-Q
- Assist Learning Algorithm: Policy Backprop

What is a model (of environment)

(A model) is something that mimics the behavior of the environment, or more generally, that allows inferences to be made about how the environment will behave. For example, given a state and action, the model might predict the resultant next state and next reward. [3, page 7]

Recap the definition of MDP $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$

- Set of states $s_t \in \mathcal{S}$
- Set of actions $a_t \in \mathcal{A}$
- Transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow prob(\mathcal{S})$, $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)$ ($prob(\mathcal{S})$ is the distribution over $\mathcal{S}$)
- Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$ (or $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$)
- Discount factor γ

Type of model

By direction [1]:

- Forward model: $(s_t, a_t) \to s_{t+1}$.
  (Typical meaning of a model in model-based RL)
- Backward/reverse model: $s_{t+1} \to (s_t, a_t)$
  (used in priorioriized sweeping discussed later)
- Inverse model: $(s_t, s_{t+1}) \to a_t$

By output type [3, page 159]:

- Distribution model: Given a state and an action, produce a probability distribution of all next states.
- Sample model: Given a state and an action, produce one next state sampled from the probability distribution.

By ways to obtain

- Known model (e.g., AlphaGo Zero)
- Learned model (e.g., Dyna)

Why to learn a model

- Planning with real robots (too expensive, too risky)
- Simulating complex physical dynamics (too expensive)
- Interactions with humans (no access)
- ......

What is planning

Models are used for planning, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced. [3, page 7]

The opposite of explicitly trial-and-error learners.

Planning vs Learning [3, page 161]

- Common: the estimation of value functions (by backing-up update operations[1])
- Difference:
    - Planning: simulated experience generated by a model
    - Learning: real experience generated by the environment

[1] i.e., the current value of the earlier state is updated to be closer to the value of the later state [3, page 9]

Example: Q-planning vs Q-learning

(Random-sample) Q-planning [3, page 161]:

```
Loop forever:
    ▶ Select a state, $s \in \mathcal{S}$, and an action, $a \in \mathcal{A}$, at random
    ▶ Send $s, a$ to a sample model, and obtain
      a sample next reward, $r$, and a sample next state, $s'$
    ▶ Apply one-step tabular Q-learning to $(s, a, r, s')$:
      $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \max_{a'} Q(s', a') - Q(s, a)]$
```

Q-learning [3, page 131]:

```
Loop for each episode:
    ▶ Initialize $s$
    ▶ Loop for each step of episode:
        ▶ Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        ▶ Take action $a$, observe $r$, $s'$
        ▶ $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
        ▶ $s \leftarrow s'$
    ▶ until $s$ is terminal
```

Type of planning [3, page 180]

- Background planning (e.g., Dyna-Q)
  - Gradually improve a policy or value function on the basis of simulated experience obtained from a model. (e.g., Q-planning)
  - Selecting actions is then a matter of comparing the current state's action values.
  - Planning is not focused on the current state.
- Decision-time planning (e.g., Heuristic Search)
  - Given s_t, output is the selection of a single action a_t.
  - Can look much deeper than one-step-ahead.
  - Focuses on a particular state.
  - Can act without any learning.

Type of planning [2]

$$\begin{tikzcd}
 & & & \text{Time of Planning} \arrow[dll] \arrow[drr] & & & \\
 & \text{Decision-time Planning} \arrow[dl] \arrow[dr] & & & & \text{Background Planning} \arrow[dl] \arrow[dr] & \\
\text{Continuous Action} & & \text{Discrete Action} & & \text{Simulated Environment} & & \text{Assist Learning Algorithm}
\end{tikzcd}$$

iLQR	Heuristic Search	Dyna (1990)	Policy Backprop
DDP	MCTS	MVE (ICML 2018)	SVG (NeurIPS 2015)
Pi^2	AlphaGo Zero (Nature, 2017)	MBPO (NeurIPS 2019)	Dreamer (ICLR 2020)
Direct collocation			PILCO (ICML 2011)
STOMP			

Table of Contents

Basic Concepts: Model and Planning

Background Planning

Simulated Environment: Dyna-Q

Assist Learning Algorithm: Policy Backprop

Simulated Environment vs. Assist Learning Algorithm[2]

- Simulated Environment (Dyna-type, black-box model)
  - First, using the current policy, data is gathered from interaction with the environment and then used to learn the dynamics model.
  - Second, the policy is improved with imagined data generated by the learned model.
  - Learn policies using model-free algorithms with rich imaginary experience without interaction with the real environment.
- Assist Learning Algorithm (policy search with backpropagation through time, white-box model)
  - Model is smooth and differentiable.
    E.g., for $s_{t+1} = f_s(s_t, a_t)$, we can have $\frac{\partial s_{t+1}}{\partial s_t}$ and $\frac{\partial s_{t+1}}{\partial a_t}$.
  - Compute the analytic gradient of the RL objective with respect to the policy, and improve the policy accordingly.

[2]http://www.cs.toronto.edu/~tingwuwang/mbrl.html, https://rlchina.org/lectures/lecture4.pdf

Simulated Environment: Dyna-Q[3]

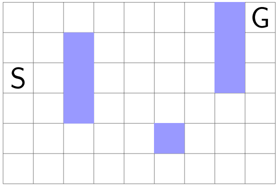
Dyna-Q: Q-learning + Model learning + Q-planning

$$\begin{tikzcd}
 & \text{Value / Policy} \arrow[dd, bend left, "\text{action}"] \\
\text{Model} \arrow[ur, "\text{planning}"] & \\
 & \text{Experience} \arrow[ul, "\text{model learning}"] \arrow[uu, bend left, "\text{direct RL}"]
\end{tikzcd}$$

```
Loop for each episode:
  ▶ Initialize $s$
  ▶ Loop for each step of episode:
      ▶ Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
      ▶ Take action $a$, observe $r$, $s'$        ← interact with env (learning)
      ▶ $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
      ▶ $Model(s,a) \leftarrow r, s'$        ← model learning
      ▶ Loop repeat $n$ times
          ▶ $s, a \leftarrow$ random previously experienced state-action pair
          ▶ $r, s' \leftarrow Model(s,a)$        ← interact with model (planning)
          ▶ $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
      ▶ $s \leftarrow s'$
  ▶ until $s$ is terminal
```

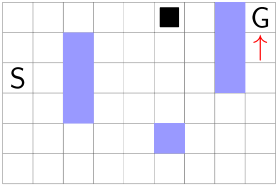
[3]Sutton (1990). Dyna, an integrated architecture for learning, planning, and reacting.

Why agent with planning can be better



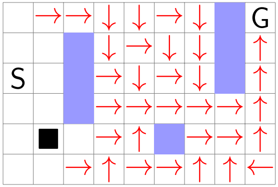
at the middle of the 2th episode

No planning ($n = 0$)



Only learn once at the end of 1th episode.

With planning ($n = 50$)



An extensive policy has been developed during the second episode via interacting with model.

Prioritized Sweeping

Recap the Q-planning procedure in Dyna-Q

Loop repeat n times

- $s, a \leftarrow$ random previously experienced state-action pair
- $r, s' \leftarrow Model(s, a)$ $\quad \leftarrow$ interact with model (planning)
- $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

Random sampling may not be the best.

- Consider the maze example before. At the beginning of the second episode, only the state–action pair leading directly into the goal $((9,5), \uparrow)$ has a positive value $(Q((9,5), \uparrow) > 0)$.
- If we randomly select (s, a) from previously experienced state-action pair, and obtain simulated r, s' from the model. $Q(s,a)$ and $Q(s',a')$ will be both zero most of the time.

Prioritized Sweeping

```
Loop for each episode:
  ▶ Initialize $s$
  ▶ Loop for each step of episode:
      ▶ Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
      ▶ Take action $a$, observe $r$, $s'$          $\leftarrow$ interact with env (learning)
      ▶ $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
      ▶ $P \leftarrow |r + \gamma \max_{a'} Q(s',a') - Q(s,a)|$          $\leftarrow$ priority
      ▶ if $P > \theta$, then insert $s, a$ into $PQueue$ with priority $P$
      ▶ $Model(s,a) \leftarrow r, s'$          $\leftarrow$ model learning
      ▶ Loop repeat $n$ times
          ▶ $s, a \leftarrow first(PQueue)$          $\leftarrow$ high priority pair first
          ▶ $r, s' \leftarrow Model(s,a)$          $\leftarrow$ interact with model (planning)
          ▶ $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
          ▶ Loop for all $(\bar{s}, \bar{a})$ predicted to lead to $s$:  $\leftarrow$ backward model
            1. $\bar{r} \leftarrow$ predicted reward of $\bar{s}, \bar{a}, s$
            2. $P \leftarrow |\bar{r} + \gamma \max_{a'} Q(s,a') - Q(\bar{s},\bar{a})|$
            3. if $P > \theta$, then insert $\bar{s}, \bar{a}$ into $PQueue$ with priority $P$
      ▶ $s \leftarrow s'$
  ▶ until $s$ is terminal
```

Simulated Environment: Other Influential Algorithms

- (MVE) Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning (ICML 2018) [4]
  - Only allow model imagination to fixed depth.
  - Use learned model for short-term horizon, use traditional Q-learning for long-term horizon.

    $$\mathcal{T} = r + \underbrace{\sum_{i=1}^{H} (\gamma^i \hat{r}(s_{i-1}, a_{i-1}, s_i))}_{\text{use learned model to rollout H steps}} + \gamma^{H+1} Q(s_H, a_H)$$

    ($H = 0$ will be the tranditional TD error)
  - Incorporating the model into Q-value target estimation.
- (MBPO) When to Trust Your Model: Model-Based Policy Optimization (NeurIPS 2019)

[4] https://zhuanlan.zhihu.com/p/102197348

Assist Learning Algorithm: Policy Backprop [2]

(smooth) models offer derivatives.
From $s_{t+1} = f_s(s_t, a_t)$ and $r_t = f_r(s_t, a_t)$ we can have

$$\frac{\partial s_{t+1}}{\partial s_t}, \frac{\partial s_{t+1}}{\partial a_t}, \frac{\partial r_t}{\partial s_t}, \frac{\partial r_t}{\partial a_t}$$

i.e. "how do small changes in action change next state?"

$$J(\theta) = \sum_{t=0}^{H} \gamma^t r_t, \quad a_t = \pi_\theta(s_t), \quad s_{t+1} = f_s(s_t, a_t), \quad r_t = f_r(s_t, a_t)$$

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{t=0}^{H} \gamma^t \frac{\partial r_t}{\partial \theta} = \sum_{t=0}^{H} \gamma^t \frac{\partial f_r(s_t, a_t)}{\partial \theta} \quad (r_t = f_r(s_t, a_t))$$

$$= \sum_{t=0}^{H} \gamma^t \left(\frac{\partial f_r(s_t, a_t)}{\partial s_t}\frac{\partial s_t}{\partial \theta} + \frac{\partial f_r(s_t, a_t)}{\partial a_t}\frac{\partial \pi_\theta(s_t)}{\partial \theta}\right) \quad (a_t = \pi_\theta(s_t))$$

Assist Learning Algorithm: Policy Backprop

$$J(\theta) = \sum_{t=0}^{H} \gamma^t \left(\frac{\partial f_r(s_t, a_t)}{\partial s_t} \frac{\partial s_t}{\partial \theta} + \frac{\partial f_r(s_t, a_t)}{\partial a_t} \frac{\partial \pi_\theta(s_t)}{\partial \theta} \right)$$

$$\begin{aligned} \frac{\partial s_t}{\partial \theta} &= \frac{\partial f_s(s_{t-1}, a_{t-1})}{\partial \theta} \\ &= \frac{\partial f_s(s_{t-1}, a_{t-1})}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial \theta} + \frac{\partial f_s(s_{t-1}, a_{t-1})}{\partial a_{t-1}} \frac{\partial \pi_\theta(s_{t-1})}{\partial \theta} \end{aligned}$$

Calculated recursively backwards in time (i.e. RNNs)

Assist Learning Algorithm: Other Influential Algorithms

- PILCO: A Model-Based and Data-Efficient Approach to Policy Search (ICML 2011) [5]
  - Use Gaussian Process to model dynamic system $x_t = f(x_{t-1}, u_{t-1})$, i.e., $p(x_t | x_{t-1}, u_{t-1}) = \mathcal{N}(\mu_t, \Sigma_t)$
  - Use learned model to approximate $V^\pi(x_0) = \sum_{t=0}^{T} \int c(x_t) p(x_t) dx_t$
  - Gradient back-propagate to update π, $\min_{\pi \in \Pi} V^{\pi_\theta}(x_0)$
  - Very high data efficiency (only 7-8 episodes for CartPole)
- (SVG) Learning Continuous Control Policies by Stochastic Value Gradients (NeurIPS 2015)
- (Dreamer) Dream to Control: Learning Behaviors by Latent Imagination (ICLR 2020 Spotlight)

[5] `https://zhuanlan.zhihu.com/p/138337983`, `https://zhuanlan.zhihu.com/p/27537744`

References I

Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker.
Model-based Reinforcement Learning: A Survey.
arXiv:2006.16712 [cs, stat], July 2020.
arXiv: 2006.16712.

Igor Mordatch and Jessica Hamrick.
Tutorial on Model-Based Methods in Reinforcement Learning.
https://sites.google.com/view/mbrl-tutorial.

Richard S. Sutton and Andrew G. Barto.
Reinforcement learning: an introduction.
Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018.