Reinforcement Learning 2

Review: MDP $(S, A, \{P_{sa}\}, \gamma, R)$

policy $\pi : S \to A$

$V^\pi(s) \triangleq$ Expected total future payoff of policy π starting from s

$V^*(s) \triangleq \max_\pi V^\pi(s)$

π^* optimal policy

π^* is greedy w.r.t. $V^*(s)$

$$\pi^*(s) = \arg\max_a \left(R(s) + \gamma \sum_{s'} P_{sa}(s') V^*(s') \right)$$

$$V^*(s) = V^{\pi^*}(s)$$

Bellman Equations

Linear system of equations

$$V^\pi(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^\pi(s')$$

$$V^*(s) = R(s) + \gamma \max_a \sum_{s'} P_{sa}(s') V^*(s')$$

Value iteration (VI), Policy iteration (PI)

VI, PI: known dynamics $\{P_{sa}\}$

– not true in many real settings eg. robotics

Assumption: Given s, a we can sample $s' \sim P_{sa}$

model-based RL	vs.	model-free RL
explicitly learn transition dynamics from trajectories sampled		doesn't learn the transitions explicitly policy gradient (REINFORCE) Q-learning

Tabular case	Continuous case
discrete state space	$S = \mathbb{R}^d$
$\lvert S\rvert < \infty$, $\lvert A\rvert < \infty$	$A = \mathbb{R}^K$

Model-based RL, tabular case

learn the dynamics

Suppose we have trajectories:

$s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \cdots$

$s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}} s_2^{(2)} \cdots$

$$P_{sa}(s') = \frac{\#\text{times we arrived at } s' \text{ after taking action } a \text{ in state } s}{\#\text{times we took action } a \text{ in state } s}$$

no Laplace smoothing

replace $\frac{0}{0}$ by $\frac{1}{|S|}$

Model-based RL algo:

Initialize π randomly, dataset of trajectories $D \leftarrow \phi$

Loop {

(a) Execute π in the real environment to collect trajectories

{ $s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \longrightarrow \cdots$

$s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \longrightarrow \cdots$ }

add them to D

(b) Estimate P_{sa} using data in D

(Estimate the reward R)

(c) invoke PI or VI to get V^* for estimated $\{P_{sa}\}$

$\pi^* = \arg\max_a \sum_{s'} P_{sa}(s') V^*(s')$

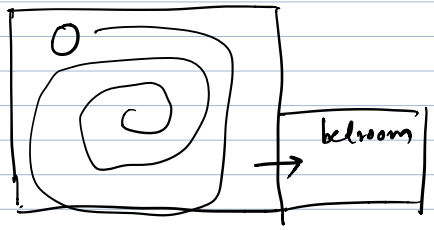
$\pi = \pi^*$

}

Why loop?

— P_{sa} in the beginning is not accurate

— initial policy may not see all the important states

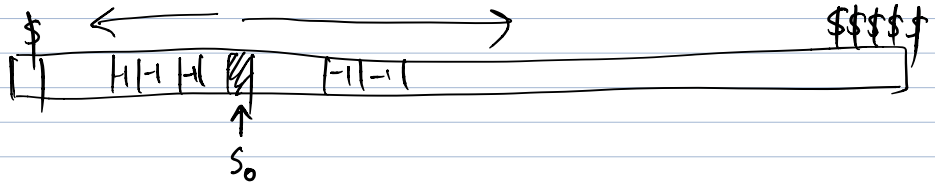


Exploitation vs. Exploration trade off

find the best policy given the current information

collect more diverse data to get more information

exploration strategy: adding noise to actions generated by policy

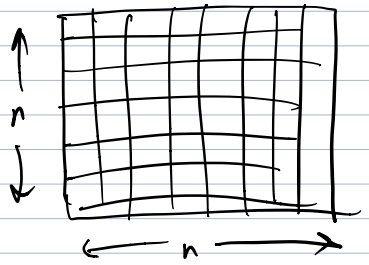


Continuous state space $S = \mathbb{R}^d$

action space (discrete, continuous)

$d = 2$ (or $d = 3$)

discretization



|new state space| = n^2

$d \geq 4$ too many states!

Q1: how do we learn / represent the transition dynamics $\{P_{sa}\}$

Q2: Value iteration (or PI)

A1 for Q1: physical simulator

A2: learning a model from data

represent the dynamics $\{P_{sa}\}$

model $s' \sim P_{sa}$ by

$$s' = f(s, a) + \xi$$

$f(\,)$ deterministic

ξ noise (eg. Gaussian)

eg. $a \in \mathbb{R}^k$, $s \in \mathbb{R}^d$

linear dynamics

$$s' = As + Ba + \xi$$

$A \in \mathbb{R}^{d \times d}$, $B \in \mathbb{R}^{d \times k}$

Linear Quadratic Regulator

$$\begin{cases} s_0^{(1)} \xrightarrow{a_0^{(1)}} \\ s_0^{(2)} \xrightarrow{a_0^{(2)}} \end{cases}$$

$\Rightarrow$ supervised learning dataset

$$\{ \underbrace{((s_0^{(1)}, a_0^{(1)})}_{x}, \underbrace{s_1^{(1)}}_{y}), \quad \ldots \quad ((s_{T-1}^{(1)}, a_{T-1}^{(1)}), s_T^{(1)})$$

$$(\underbrace{(s, a)}_{x}, \underbrace{s'}_{y})$$

linear case: regression

$$A, B = \operatorname{argmin} \sum_{i=1}^{n} \sum_{t=0}^{T-1} \| s_{t+1}^{(i)} - A s_t^{(i)} - B a_t^{(i)} \|_2^2$$

non-linear

f_θ parametrized by θ

$$\theta = \arg\min_\theta \sum_{i=1}^{n} \sum_{t=0}^{T-1} \| s_{t+1}^{(i)} - f_\theta(s_t^{(i)}, a_t^{(i)}) \|_2^2$$

$$s' = f_\theta(s, a) + \xi$$

$$f_\theta(s, a) = A \phi_1(s) + B \phi_2(a) + \xi$$

(ϕ_1: feature map, ϕ_2: feature map)

$$f_\theta(s, a) = \text{neural network}(s, a)$$

— Value iteration with continuous state space

Idea: Represent the value $V(\ ) : \mathbb{R}^d \to \mathbb{R}$

$$V(s) = \theta^T \Phi(s) \qquad \Phi(s): \text{feature map}$$
$$= \text{Neural net}(s; \theta)$$

Review: VI for discrete case

$$V(s) = R(s) + \gamma \max_a \sum_{s'} P_{sa}(s') V(s')$$
$$= R(s) + \gamma \max_a \mathbb{E}_{s' \sim P_{sa}} [V(s')]$$
$$= \max_a \left(R(s) + \gamma \mathbb{E}_{s' \sim P_{sa}} [V(s')] \right)$$

target value

Assume: A is discrete

Enforce BE on some randomly chosen S

Loop {

(a) // estimate RHS of BE for randomly chosen $s^{(1)} \dots s^{(n)}$

fix i, $s^{(i)}$ take $s = s^{(i)}$

$\forall a \in A$ sample $s_1' \dots s_K' \sim P_{sa}$ ← estimated dynamics

$$q(a) = R(s) + \frac{\gamma}{K} \sum_{j=1}^{K} V(s_j')$$

estimator for $\mathbb{E}_{s' \sim P_{sa}}[V(s')]$

$$y^{(i)} = \max_a q(a)$$

target value

(b) // enforce $V(s^{(i)}) \approx y^{(i)}$

$$\theta = \arg\min_\theta \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T \phi(s^{(i)}))^2$$

}

fitted Value Iteration

$V^* \longrightarrow \pi^*$?

$$\pi^*(s) = \arg\max_a \sum_{s'} P_{sa}(s') V^*(s')$$

$$= \arg\max_a \mathbb{E}_{s' \sim P_{sa}}[V^*(s')]$$

Choice 1: estimate the expectation by sampling

$$\approx \frac{1}{K} \sum_{i=1}^{K} V(s_i') \qquad s_1' \dots s_K' \sim P_{sa}$$

Choice 2: $\mathbb{E}_{s' \sim P_{sa}}[V^*(s')] = \mathbb{E}[V^*(f(s,a) + \xi)]$

$$\approx V^*(f(s,a))$$

Continuous **A**

$$y^{(i)} = \max_a q(a)$$

$$y^{(i)} = \max_a \left(R(s^{(i)}) + \gamma \, \mathbb{E} \, V(f(s^{(i)}, a)) \right)$$

by gradient ascent over actions