APS 1080 – LECTURE 3
MODEL BASED RL – DYNAMIC PROGRAMMING

$$M \underset{S,\ R}{\overset{A}{\rightleftharpoons}} E$$

~Goal

$$S_0 \ A_0 \ R_1 \ S_1 \ A_1 \ \ldots \ S_i \ | \ A_i \quad R_{i+1} \ S_{i+1} \ A_{i+1} \ \ldots \ S_T$$

What Action to take if you're in State i.

Return: Sum of subsequent Rewards as a result of our actions

$$G_i = R_{i+1} + \gamma R_{i+2} + \gamma^2 R_{i+3} + \ldots$$

Goal: select A_i such that the expected value of the return is maximized

$\downarrow$
Action

$$\max \mathbb{E}[G_i]$$

VALUE FUNCTION

$$V_\pi(s) = \mathbb{E}[G_t \mid S_t = s]$$

$$q_\pi(s) = \ldots$$

Given V_π

p : model of the environment

We can solve Action Selection Problem.

To solve Action Selection Problem, you need either:

① V and $p \Rightarrow \pi$

② $Q \Rightarrow \pi$

The Policy, which is the mechanism for action selection.

The states are drawn from $\mathbb{S} = \{ S^{(1)}, S^{(2)}, \ldots, S^{(N)} \}$

S	$V_\pi(s)$
$S^{(1)}$	$\mathbb{E}[G_t \mid S_t = S^{(1)}]$
$S^{(2)}$	
$\vdots$	
$S^{(N)}$	

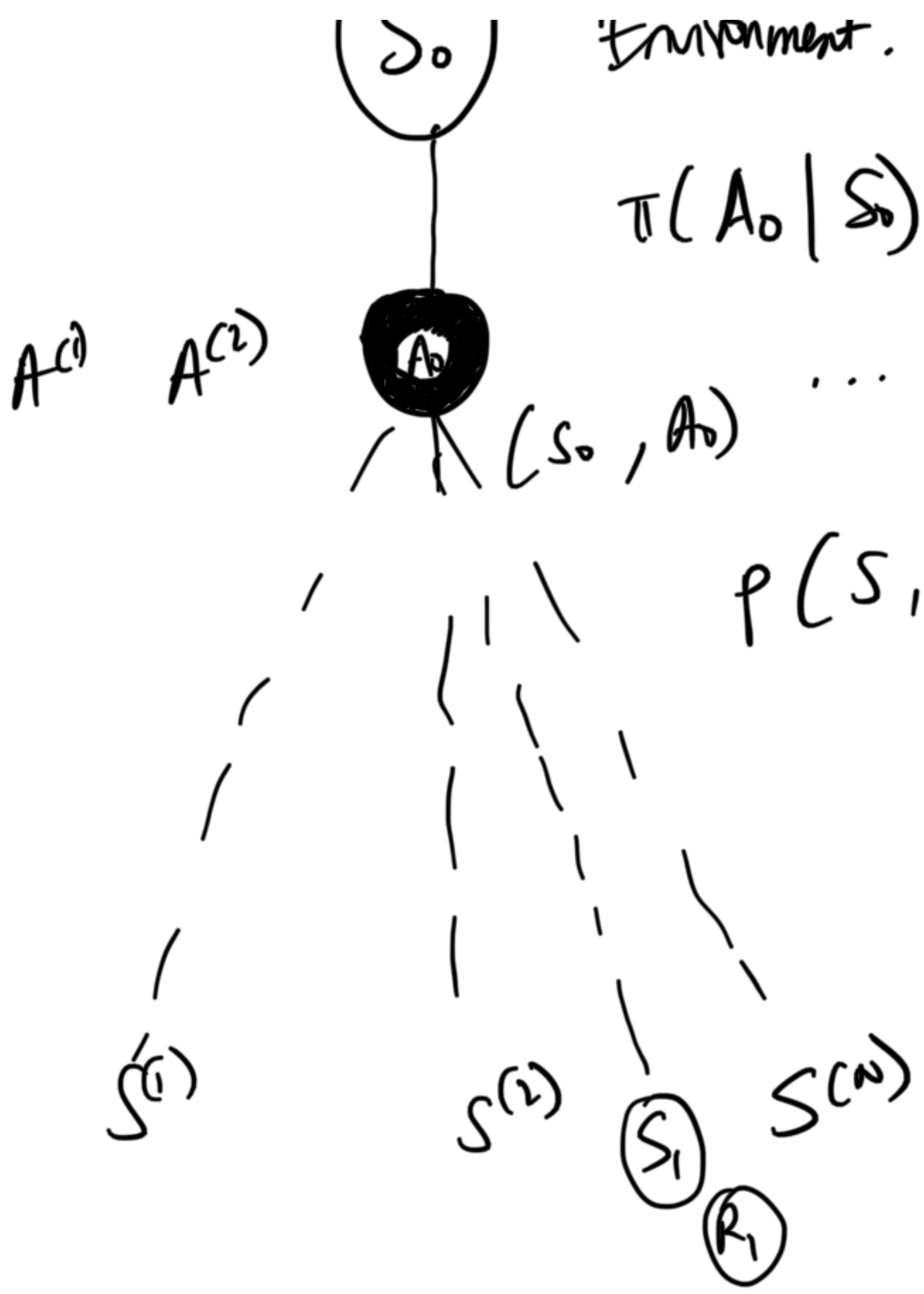
Target state: which state to go to

then consult "p" to see what action to take to get to that probability.

$$V_\pi(s) = \mathbb{E}[G_t \mid S_t = s]$$

Subscript: Time sequence

Superscript: Selection from the set.



environment.

to states the time t_1 ... which is drawn from the state of possible actions.

$p(S_1, R_1 | S_0, A_0) \rightarrow$ Model.

$A_0 \in \{ A^{(1)}, A^{(2)}, \dots A^{(K)} \}$

This is how π and p interplay in continuous state/action rewards.
$S \rightarrow A$ transition

Action $\rightarrow$ State (weighted by distribution p)

The model we may or may not have it.

The

with p we can calculate V_π, π and π^*
$\downarrow$
optimal policy.

Dynamic programming does NOT use experience

How we determine V_π and π when we don't have P?
$\rightarrow$ You need experience + learning to be surrogates of

Probability of selecting an action given you're in state s.

$$V_\pi(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} \sum_{r \in R} \left[p(s', r | s, a) \left[r + \gamma V_\pi(s') \right] \right]$$

Indices of a for loop

for loop

consider all possible states

over all possible rewards

Expected value → Weighted Average over the entire tree.

You're building the Agent.

No knowledge about the problem → π would be an equiprobable distribution.

→ Eventually, you'd get to a more granular π with a less naive distribution.

We have an equation that is recursively defined.

$$V_\pi(s) = \gamma V_\pi(\cdot)$$

gamma

We can calculate $V_\pi(\cdot)$ via linear system methods but it's not very scalable because you end up with too many variables and it's very computationally expensive. Also, it would only work if you have "p".

Instead of doing that, we calculate $V_\pi(\cdot)$ via a successive approximation.

$$f(\cdot) = \underset{\text{Contraction}}{\Gamma}(f(\cdot))$$

$f(\cdot) \leftarrow$ initiate $-$ function $\forall x$

loop

  loop $\forall x$

    prev $\leftarrow f(x)$

    next $\leftarrow \Gamma(\text{prev})$

    $f(x) \leftarrow$ next

    $\Delta(x) \leftarrow |\text{next} - \text{prev}|$

  if $\|\Delta\| < \theta$

    $\rightarrow$ break.

Initialize $V_\pi(x)$ for all x arbitrarily.
We set the terminal states to zero and all the other values at an arbitrary number.

V_π

States	
$S^{(1)}$	
$S^{(2)}$	
$\vdots$	
$S^{(n)}$	

$$V_\pi(S_T) = 0$$

π	$a^{(1)}$	$a^{(2)}$	$\dots$	$a^{(k)}$	
$S^{(1)}$					$\Sigma = 1$
$S^{(2)}$					$\Sigma = 1$
$\vdots$					
$S^{(n)}$					

Input π

Init $V_\pi(s)$ arbitrarily $\forall s$

loop $\Delta \longleftarrow 0$

loop $\forall s$

prev $\leftarrow V_\pi(s)$

Next $\leftarrow \Gamma(V_\pi(\cdot))$

$\Sigma \pi \ldots \Sigma\Sigma \left[r_p + \gamma V_\pi(s') \right]$

$\Delta \leftarrow \max\left[\Delta, (\text{prev} - \text{next}) \right]$

if $\Delta < \theta$:

Break

Iterate over the states, successively estimating the values of the value functions

when the deltas are low enough then you stop.

Given an arbitrary model and policy, we can determine a value function

$p, \pi \longrightarrow V_\pi(\cdot)$ $\longrightarrow$ Evaluation of the policy.

Evaluation of π in $\mathcal{E}$ ~ "Prediction"

$\pi \xrightarrow{\text{eval}} V_\pi(\cdot)$

$\pi \longrightarrow \pi'$

eval

$$\left\{\begin{array}{l} \pi' \longrightarrow V_{\pi'}(\cdot) \\ \pi' \xrightarrow[\text{Improvement}]{V_{\pi'}} \pi'' \end{array}\right.$$

$$\left\{\begin{array}{l} \pi'' \longrightarrow V_{\pi''}(\cdot) \\ \pi'' \longrightarrow \pi''' \end{array}\right.$$

$$\left.\begin{array}{l} \text{until } V_\pi(i) \approx V_\pi(i+1) \\ \pi(i) \simeq \pi(i+1) \end{array}\right] \begin{array}{l} \text{Fixed Point} \\ \pi_* \end{array}$$

You modify the policy to help you improve the likelihood of going to the most desirable states.

Summarize Dynamic Programming

1. $P, \pi_0 \longrightarrow V_{\pi_0}$ (Prediction)

2. $\pi_0, V_0 \longrightarrow \pi_1$ $(\pi_1 \geq \pi_0)$ (Improvement)

3. Generalized Policy Iteration (G.P.I)

   Prediction
   Improvement

   Until fixed Point is reached $\longrightarrow \pi_*$

4. GPI $\rightarrow$ Evaluation 1
   Improvement 2

   Evaluation 1
   Improvement 2

GPI:	Eval	Improv.	Eval	Improv.			Eval	Improv.	Eval
π_0	1	2	1	2		. . .	V_{π_n}	V_{π_n}	V_{π_n}

$$\underbrace{V_{\pi_n} \quad V_{\pi_n} \quad V_{\pi_n}}_{\text{Fixed Point}}$$

V_{π^*}, π^*

This is an asymptotic Process

Other approach

Run it a few times and truncate the evaluation.

Select the best value amongst bag of truncated evaluations.

Step 4 helps us move away from the need of having a model.

What are other ways to approximate a value function that would not require the use of a "p"?

Next week:

How to approximate V_π without P by leveraging experience?

→ How can we approximate q_π and a policy from experience?

↳ π