A Guided Tour of Chapter 10: Reinforcement Learning for Control

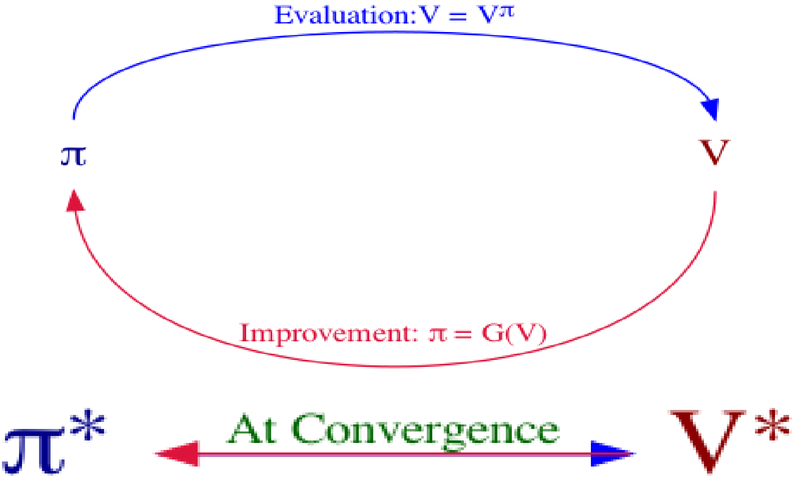
Ashwin Rao

ICME, Stanford University

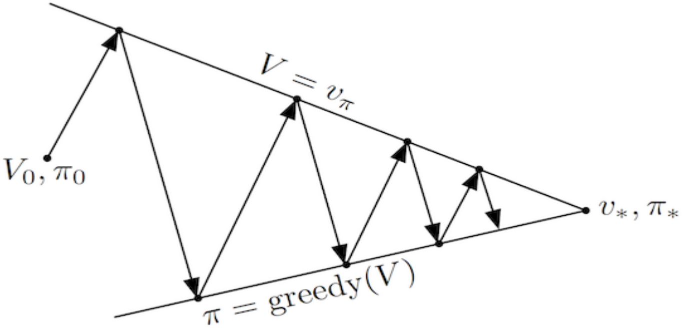
RL does not have access to a probability model

- DP/ADP assume access to probability model (knowledge of $\mathcal{P}_R$)
- Often in real-world, we do not have access to these probabilities
- Which means we'd need to *interact* with the *actual environment*
- Actual Environment serves up individual experiences, not probabilities
- Even if MDP model is available, model updates can be challenging
- Often real-world models end up being too large or too complex
- Sometimes estimating a *sampling model* is much more feasible
- So RL interacts with either *actual* or *simulated* environment
- Either way, we receive *individual experiences* of next state and reward
- We saw how RL Prediction learns from individual experiences
- Now we extend those ideas to RL Control: Learning Optimal VF
- We start with Tabular RL Control

Let us recall the Policy Iteration algorithm



The idea of Generalized Policy Iteration (GPI)



- *Any* Policy Evaluation method, *Any* Policy Improvement method
- Policy Evaluation estimates $V^{(\pi)}$, eg: Iterative Policy Evaluation
- Policy Improvement produces $\pi' \geq \pi$, eg: Greedy Policy Improvement
- Policy Evaluation and Policy Improvement alternate until convergence

Natural Idea: GPI with Tabular Monte-Carlo Evaluation

- Let us explore GPI with Tabular Monte-Carlo evaluation
- So we will do Policy Evaluation with Tabular MC evaluation
- And we will do the usual Greedy Policy Improvement
- But Greedy Policy Improvement requires a model of MDP

$$\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}} \{\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s,a,s') \cdot V^{\pi}(s')\}$$

- However, it works if we were working with Action-Value Function

$$\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a)$$

- This means: Policy Evaluation for Action-Value Function $Q^{\pi}(s,a)$
- Following a policy π, update Q-value for each (S_t, A_t) each episode:

$$Count(S_t, A_t) \leftarrow Count(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{Count(S_t, A_t)} \cdot (G_t - Q(S_t, A_t))$$

ϵ-Greedy Policy Improvement

- A full Policy Evaluation with MC takes too long
- So we typically improve policy after each episode
- This can lead to some actions not being tried enough
- Which can lead to premature (greedy) domination of an action
- Which can lead to other actions getting "locked-out"
- Same as *Explore v/s Exploit* dilemma of Multi-Armed Bandit problem
- Simple solution: Perform an ϵ-Greedy Policy Improvement
- All $|\mathcal{A}|$ actions are tried with non-zero probability (for each state)
- Pick the greedy action with probability $1 - \epsilon$
- With probability ϵ, randomly choose one of the $|\mathcal{A}|$ actions

$$\text{Stochastic Policy } \pi(s, a) = \begin{cases} \frac{\epsilon}{|\mathcal{A}|} + 1 - \epsilon & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(s, b) \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

ϵ-Greedy improves the policy

Theorem

For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to Q^π is an improvement, i.e., $\boldsymbol{V}^{\pi'}(s) \geq \boldsymbol{V}^{\pi}(s)$ for all $s \in \mathcal{N}$.

- Applying $\boldsymbol{B}^{\pi'}$ repeatedly (starting with $\boldsymbol{V}^{\pi}$) converges to $\boldsymbol{V}^{\pi'}$:

$$\lim_{i \to \infty} (\boldsymbol{B}^{\pi'})^i(\boldsymbol{V}^{\pi}) = \boldsymbol{V}^{\pi'}$$

- So the proof is complete if we prove that:

$$(\boldsymbol{B}^{\pi'})^{i+1}(\boldsymbol{V}^{\pi}) \geq (\boldsymbol{B}^{\pi'})^i(\boldsymbol{V}^{\pi}) \text{ for all } i = 0, 1, 2, \ldots$$

- Increasing tower of Value Functions $[(\boldsymbol{B}^{\pi'})^i(\boldsymbol{V}^{\pi}) | i = 0, 1, 2, \ldots]$ with repeated applications of $\boldsymbol{B}^{\pi'}$

Proof of ϵ-Greedy improving the policy

To prove base case (proof by induction), note: $\boldsymbol{B}^{\pi'}(\boldsymbol{V}^{\pi})(s) = Q^{\pi}(s, \pi'(s))$

$$\begin{aligned}
Q^{\pi}(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(s, a) \cdot Q^{\pi}(s, a) \\
&= \frac{\epsilon}{|\mathcal{A}|} \cdot \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) + (1 - \epsilon) \cdot \max_{a \in \mathcal{A}} Q^{\pi}(s, a) \\
&\geq \frac{\epsilon}{|\mathcal{A}|} \cdot \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) + (1 - \epsilon) \cdot \sum_{a \in \mathcal{A}} \frac{\pi(s, a) - \frac{\epsilon}{|\mathcal{A}|}}{1 - \epsilon} \cdot Q^{\pi}(s, a) \\
&= \sum_{a \in \mathcal{A}} \pi(s, a) \cdot Q^{\pi}(s, a) = \boldsymbol{V}^{\pi}(s)
\end{aligned}$$

Induction step is proved by monotonicity of $\boldsymbol{B}^{\pi}$ operator (for any π):

$$\text{Monotonicity Property of } \boldsymbol{B}^{\pi} : \boldsymbol{X} \geq \boldsymbol{Y} \Rightarrow \boldsymbol{B}^{\pi}(\boldsymbol{X}) \geq \boldsymbol{B}^{\pi}(\boldsymbol{Y})$$

$$\text{So } (\boldsymbol{B}^{\pi'})^{i+1}(\boldsymbol{V}^{\pi}) \geq (\boldsymbol{B}^{\pi'})^{i}(\boldsymbol{V}^{\pi}) \Rightarrow (\boldsymbol{B}^{\pi'})^{i+2}(\boldsymbol{V}^{\pi}) \geq (\boldsymbol{B}^{\pi'})^{i+1}(\boldsymbol{V}^{\pi})$$

$\square$

GLIE

Definition

Greedy in the Limit with Infinite Exploration (GLIE):

- All state-action pairs are explored infinitely many times

$$\lim_{k \to \infty} N_k(s, a) = \infty$$

- The policy converges to a greedy policy

$$\lim_{k \to \infty} \pi_k(s, a) = \mathbb{I}_{a = \arg\max_{b \in \mathcal{A}} Q(s,b)}$$

ϵ-greedy can be made GLIE if ϵ is reduced as: $\epsilon_k = \frac{1}{k}$

GLIE Tabular Monte-Carlo Control

- Sample k-th episode using π: $\{S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T\} \sim \pi$
- For each state S_t and action A_t in the episode:

$$Count(S_t, A_t) \leftarrow Count(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{Count(S_t, A_t)} \cdot (G_t - Q(S_t, A_t))$$

- Improve policy at end of episode based on updated Q-Value function:

$$\epsilon \leftarrow \frac{1}{k}$$

$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

Theorem

GLIE Tabular Monte-Carlo Control converges to the Optimal Action-Value function: $Q(s, a) \rightarrow Q^*(s, a)$.

MC versus TD Control

- TD learning has several advantages over MC learning:
  - Lower variance
  - Online
  - Can work with incomplete traces or continuing traces
  - Generic interface of `Iterable` of atomic experiences allows for serving up atomic experiences in any order (eg: atomic experience replays)
- So use TD instead of MC in our Control loop
  - Apply TD to $Q(S, A)$ (instead of $V(S)$)
  - Use ϵ-greedy Policy Improvement
  - Update $Q(S, A)$ after each *atomic experience*
  - ϵ-greedy policy automatically updated after each atomic experience

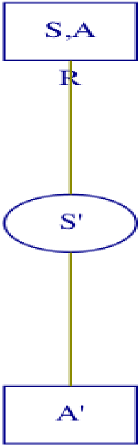
Tabular SARSA Algorithm

- Tabular SARSA is our first TD Control algorithm
- Like Tabular MC Control, Policy Improvement is ϵ-greedy
- But here Policy Evaluation is with a TD target, as below:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \cdot (R_{t+1} + \gamma \cdot Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

- Note that $Q(S, A)$ is updated after each atomic experience
- ϵ-greedy policy automatically updated after each atomic experience
- Action A_t is chosen from State S_t based on ϵ-greedy policy
- Action A_{t+1} is chosen from State S_{t+1} based on ϵ-greedy policy
- Note: Instead of ϵ-greedy, we could employ a more sophisticated exploratory policy derived from Q-value function (ϵ-greedy is just our default simple exploratory policy derived from Q-value function)

SARSA Visualization



Convergence of Tabular SARSA

Theorem

Tabular SARSA converges to the Optimal Action-Value function, $Q(s,a) \rightarrow Q^*(s,a)$, under the following conditions:

- *GLIE sequence of policies $\pi_t(s,a)$*
- *Robbins-Monro sequence of step-sizes α_t*

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Tabular n-step SARSA

- Tabular SARSA bootstraps the Q-Value Function with update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

- So it's natural to extend this to bootstrapping with 2 steps ahead:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2}) - Q(S_t, A_t))$$

- Generalize to bootstrapping with $n \geq 1$ time steps ahead:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_{t,n} - Q(S_t, A_t))$$

- $G_{t,n}$ (call it n-step bootstrapped return) is defined as:

$$\begin{aligned} G_{t,n} &= \sum_{i=t+1}^{t+n} \gamma^{i-t-1} \cdot R_i + \gamma^n \cdot Q(S_{t+n}, A_{t+n}) \\ &= R_{t+1} + \gamma \cdot R_{t+2} + \ldots + \gamma^{n-1} \cdot R_{t+n} + \gamma^n \cdot Q(S_{t+n}, A_{t+n}) \end{aligned}$$

Tabular λ-Return SARSA

- Instead of $G_{t,n}$, a valid target is a weighted-average target:

$$\sum_{n=1}^{N} u_n \cdot G_{t,n} + u \cdot G_t \text{ where } u + \sum_{n=1}^{N} u_n = 1$$

- Any of the u_n or u can be 0, as long as they all sum up to 1
- The λ-Return target is a special case of weights u_n and u

$$u_n = (1-\lambda) \cdot \lambda^{n-1} \text{ for all } n = 1, \ldots, T-t-1$$

$$u_n = 0 \text{ for all } n \geq T-t \text{ and } u = \lambda^{T-t-1}$$

- We denote the λ-Return target as $G_t^{(\lambda)}$, defined as:

$$G_t^{(\lambda)} = (1-\lambda) \cdot \sum_{n=1}^{T-t-1} \lambda^{n-1} \cdot G_{t,n} + \lambda^{T-t-1} \cdot G_t$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \cdot (G_t^{(\lambda)} - Q(S_t, A_t))$$

Tabular SARSA(λ)

- λ can be tuned from SARSA ($\lambda = 0$) to MC Control ($\lambda = 1$)
- Note that for $\lambda > 0$, λ-Return SARSA is an Offline Algorithm
- SARSA(λ) is online "version" of λ-Return SARSA
- Similar to TD(λ) for Prediction, SARSA(λ) uses Eligibility Traces
- Eligibility Trace for a given trace experience at time t is a function

$$E_t : \mathcal{N} \times \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$$

$$E_0(s, a) = 0, \text{ for all } s \in \mathcal{N}, a \in \mathcal{A}$$

$$E_t(s, a) = \gamma \cdot \lambda \cdot E_{t-1}(s, a) + \mathbb{I}_{S_t = s, A_t = a}, \text{ for all } s \in \mathcal{N}, a \in \mathcal{A}, \text{ for all } t$$

- Tabular SARSA(λ) performs following update at each time step t in each trace experience (for each $s \in \mathcal{N}, a \in \mathcal{A}$):

$$Q(s, a) \leftarrow Q(s, a) + \alpha \cdot (R_{t+1} + \gamma \cdot Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)) \cdot E_t(s, a)$$

Key Takeaways from this Chapter

- Bias-Variance tradeoff of TD versus MC
- MC learns the mean of the observed returns while TD learns something "deeper" - it implicitly estimates an MRP from given data and produces the Value Function of the implicitly-estimated MRP
- Understanding TD versus MC versus DP from the perspectives of:
  - "Bootstrapping"
  - "Experiencing"
- "Equivalence" of λ-Return Prediction and TD(λ) Prediction
- TD is equivalent to TD(0) and MC is "equivalent" to TD(1)