10.0: Intro

In RL, we talk a lot about prediction and control. Prediction refers to the prediction of how well a fixed policy performs, and control refers to the search of a better policy.

- In ch 9, we talked about on-policy prediction. Now we talk about the control setting:
  Function approximation of the action-value function $\hat{q}(s, a, w) \approx q_*(s, a)$.

- we will see semi-gradient SARSA which is an extension of the semi-gradient TD(0) from last chapter, but with action values.
- Episodic extension is straightforward but continous is trickier.

10.1 : Episodic Semi-Gradient Control

Instead of learning state values $V(s)$, we're now learning action-values $Q(s,a)$ using function approximation.

The approximate action-value function $\hat{q} \approx q_\pi$ is the parameterized function w/ weight vector w. The general gradient-descent update for action-value prediction is:

$$w_{t+1} = w_t + \alpha \left[U_t - \hat{q}(S_t, A_t, w_t) \right] \nabla \hat{q}(S_t, A_t, w_t)$$

- It looks a lot like the state value version in Ch 9.
- U_t is the target, for example, $R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, w_t)$ for one-step SARSA, but it could be the full Monte-Carlo return or any of the n-step SARSA returns.
- With the one-step SARSA target, the method is called Episodic Semi-gradient one-step SARSA.

This forms the action-value prediction part, and for control we need to couple that to policy improvement and action selection techniques:

- action selection: $A^*_t = \underset{a}{\operatorname{argmax}}\, \hat{q}(S_t, a, w_t)$
- policy improvement: ϵ-greedy.

Algo : 1-step SARSA using function approximation

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    If S' is terminal:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
      Go to next episode
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

1-step

// Semi-gradient bc we're treating the target as if it were fixed, even though it actually depends on the weight we're updating.

10.2: Semi-gradient n-step SARSA

use n-step return as the target (U_t) in the usual update.

$$w_{t+n} \doteq w_{t+n-1} + \alpha \left[G_{t:t+n} - \hat{q}(S_t, A_t, w_{t+n-1}) \right] \nabla \hat{q}(S_t, A_t, w_{t+n-1}),$$

where $G_{t:t+n} \doteq$

$$\begin{cases} R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, w_{t+n-1}), & \text{if } t+n < T \\ G_t, & \text{if } t+n \geq T. \end{cases}$$

Algo: n-step SARSA using function approximation

Episodic semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_*$ or q_π

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$, a positive integer n
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$ or ε-greedy wrt $\hat{q}(S_0, \cdot, \mathbf{w})$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \dots$:
    If $t < T$, then:
      Take action A_t
      Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
      If S_{t+1} is terminal, then:
        $T \leftarrow t+1$
      else:
        Select and store $A_{t+1} \sim \pi(\cdot|S_{t+1})$ or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
    $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    If $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ ($G_{\tau:\tau+n}$)
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[G - \hat{q}(S_\tau, A_\tau, \mathbf{w}) \right] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
  Until $\tau = T - 1$

10.3: Average Rewards: a new problem setting for Continuing tasks

We introduce a third classical setting for formulating the goal in MDPs.

- Episodic setting.
- Discounted setting.
- **Average reward setting.**

It applies to continuing problems (no start or end). There is no discounting. Discounting setting is problematic with function approximation; this setting replaces it.

In the average reward setting, the quality of a policy π is defined as the average rate of reward (short average reward) $r(\pi)$ while following that policy:

$$r(\pi) = \lim_{h \to \infty} \frac{1}{h} \sum_{t=1}^{h} E\left[R_t \mid S_0, A_{0:t-1} \sim \pi \right]$$

// Expected reward per time step when following policy π, conditioned on initial state S_0

- $h \to \infty$ implies this is continuous task.
- add up all the expected rewards from $t=1$ to $t=h$.
- Expected reward at time step t when following actions from time 0 to $t-1$ were chosen according to policy π.

$$= \lim_{t \to \infty} E\left[R_t \mid A_{0:t-1} \sim \pi \right]$$

// assumes $\mu_\pi(s) \doteq \lim_{t \to \infty} Pr\left(S_t = s \mid A_{0:t-1} \sim \pi \right)$

$$= \sum_s \mu_\pi(s) \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a)\, r, \quad \text{where}$$

$\mu_\pi(s)$ = the long-run probability of being in state s (steady-state distribution).

This works under the ergodicity assumption: μ_π is the steady-state distribution, and it exists for any π independently of S_0 (starting state and early decisions have no effect in the limit).

- All policies that attain the maximal value of $r(\pi)$ are considered optimal.
- The steady state distribution is the distribution under which when you select an action you end up in the same distribution. Formally,

$$\sum_s \mu_\pi(s) \sum_a \pi(a|s)\, p(s'|s,a) = \mu_\pi(s')$$

Ergodicity assumptions:

1) Irreducibility: There is a non-zero probability to go from any state to any other state,

2) Aperiodicity: You can return to any state at irregular intervals.

Differential Returns

Definition: In the average-reward setting, differential returns = differences btwn rewards and average rewards:

$$G_t = R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + \dots$$

The corresponding value functions are different value functions (remove all γ and replace rewards by the difference):

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r - r(\pi) + v_\pi(s')\right]$$

$$q_\pi(s) = \sum_{s',r} p(s',r|s,a)\left[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s',a')\right]$$

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r - \max_\pi r(\pi) + v_*(s')\right]$$

$$q_*(s) = \sum_{s',r} p(s',r|s,a)\left[r - \max_\pi r(\pi) + \max_{a'} q_*(s',a')\right]$$

Differential forms of TD error:

For state-values:

$$\delta_t = R_{t+1} - \bar{R}_{t+1} + \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$$

For action-values:

$$\delta_t = R_{t+1} - \bar{R}_{t+1} + \hat{q}(S_{t+1}, A_{t+1}, w_t) - \hat{q}(S_t, A_t, w_t)$$

where $\bar{R}_t$ is an estimate at time t of the average reward $r(\pi)$.

With these alternative definitions, most of the algorithms already seen carry through the average-reward setting without change. For example, the average reward version of semi-gradient SARSA is defined with the new version of the TD error:

$$w_{t+1} = w_t + \alpha \delta_t \nabla \hat{q}(S_t, A_t, w_t)$$

Algo: Differential 1-step SARSA.

Differential semi-gradient Sarsa for estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step sizes $\alpha, \beta > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)

Initialize state S, and action A
Loop for each step:
  Take action A, observe R, S'
  Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
  $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \beta\delta$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
  $S \leftarrow S'$
  $A \leftarrow A'$

Algo has 2 learning processes:

1) Average reward estimation: $\bar{R} \leftarrow \bar{R} + \beta\delta$

2) Action-value learning: $w \leftarrow w + \alpha\delta\nabla\hat{q}(S, A, w)$

Others:

The general form for a semi-gradient temporal difference update is:

$$w \longleftarrow w + \alpha\left[\text{Target} - \text{Estimate}\right]\nabla\left(\text{Estimate}\right).$$

For semi-gradient n-step off-policy TD, the update formula is:

$$w_{t+n} \doteq w_{t+n-1} + \alpha\rho_t\rho_{t+1}\cdots\rho_{t+n-1}\left[G_{t:t+n} - \hat{v}\left(S_t, w_{t+n-1}\right)\right]\nabla\hat{v}\left(S_t, w_{t+n-1}\right)$$

10.4: Depreciating the Discounted Setting

Section explains why discounting becomes problematic in the continuous task setting.

- The discounted setting is useful for the tabular case where the return for each state can be separated and averaged.
- In continuing tasks, we could measure the discounted return at each time step, but it's useless:
  - Average of discounted rewards is proportional to average reward.
  - Policy ranking remains the same regardless of discount factor.

⇒ Discounting does not change which policy is "best".

Proof: Idea: Symmetry argument. Each time step is the same as every other, with discounting each reward appears exactly once in return, the th reward will be undiscounted in the t^{st} return, discounted once in the $t-1^{th}$ return, and so on. The weight of the th reward is thus $1 + \gamma + \gamma^2 + \ldots = 1/(1-\gamma)$. Because all states are the same, they are all weighted by this, and thus the average of the returns will be this weight times the average reward.

Proof:

The Futility of Discounting in Continuing Problems

Perhaps discounting can be saved by choosing an objective that sums discounted values over the distribution with which states occur under the policy:

$$
\begin{aligned}
J(\pi) &= \sum_s \mu_\pi(s) v_\pi^\gamma(s) && \text{(where } v_\pi^\gamma \text{ is the discounted value function)} \\
&= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \,|\, s, a) \left[r + \gamma v_\pi^\gamma(s') \right] && \text{(Bellman Eq.)} \\
&= r(\pi) + \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \,|\, s, a) \gamma v_\pi^\gamma(s') && \text{(from (10.7))} \\
&= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s' \,|\, s, a) && \text{(from (3.4))} \\
&= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \mu_\pi(s') && \text{(from (10.8))} \\
&= r(\pi) + \gamma J(\pi) \\
&= r(\pi) + \gamma r(\pi) + \gamma^2 J(\pi) \\
&= r(\pi) + \gamma r(\pi) + \gamma^2 r(\pi) + \gamma^3 r(\pi) + \cdots \\
&= \frac{1}{1-\gamma} r(\pi).
\end{aligned}
$$

The proposed discounted objective orders policies identically to the undiscounted (average reward) objective. The discount rate γ does not influence the ordering!

Root cause of the difficulties w/ discounting is that we lost the policy improvement theorem with function approximation. It is no longer true that if we change the policy to improve the discounted value of one state then we are guaranteed to have improved the overall policy. This lack of theoretical guarantee is an ongoing area of research.

10.5 : Differential Semi-gradient n-step SARSA

To generalize to n-step bootstrapping, we need an n-step version of the TD error.

N-step return (differential form)

each reward is differential.

$$G_{t:t+n} = \overbrace{R_{t+1} - \bar{R}_{t+1}}^{} + \overbrace{R_{t+2} - \bar{R}_{t+2}}^{} + \dots + \overbrace{R_{t+n} - \bar{R}_{t+n}}^{} + \underbrace{\hat{q}(S_{t+n}, A_{t+n}, w_{t+n-1})}$$

bootstrap at the end : add estimated Q-value after n-steps.

- $\bar{R}_t$ is an estimate of $r(\pi)$ at time t, with $n \geq 1$ and $t+n < T$.
- If $t+n \geq T$, then we define $G_{t:t+n} = G_t$ as usual.

n-step error :

$$\delta_t = G_{t:t+n} - \hat{q}(S_t, A_t, w)$$

And then we can apply the usual semi-gradient SARSA update.

Algo :

Differential semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_\pi$ or q_*

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$, a policy π
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average-reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)
Algorithm parameters: step size $\alpha, \beta > 0$, a positive integer n
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Initialize and store S_0 and A_0
Loop for each step, $t = 0, 1, 2, \ldots$:
  Take action A_t
  Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$, or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
  $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  If $\tau \geq 0$:
    $\delta \leftarrow \sum_{i=\tau+1}^{\tau+n} (R_i - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(S_\tau, A_\tau, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S_\tau, A_\tau, \mathbf{w})$

10.6: Summary

- Extension of parametrized action-value functions and semi-gradient descent to control,
- Average-reward setting replacing discounting for continuous case.
- This new setting involves a differential version of everything.