10.1 - Episodic Semi-Gradient Control!

Jan 26, 2022

The semi-gradient Sarsa algorithm is the next step after semi-gradient TD(0).

Instead of $S_t \mapsto U_t$, now it's $S_t, A_t \mapsto U_t$.

Gradient Descent!

$$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha\left[U_t - \hat{q}(S_t, A_t, \vec{w}_t)\right] \nabla \hat{q}(S_t, A_t, \vec{w}_t)$$

For One-Step Sarsa!

$$\vec{w}_{t+1} \equiv \vec{w}_t + \alpha\left[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \vec{w}_t) - \hat{q}(S_t, A_t, \vec{w}_t)\right] \nabla \hat{q}(S_t, A_t, \vec{w}_t)$$

↳ Episodic Semi-Gradient One-Step Sarsa

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)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla \hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

10.2 - Semi-Gradient n-step Sarsa; n-step Gradient Descent;

$$G_{t:t+n} \equiv R_{t+1} + \gamma R_{t+2} \ldots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \vec{w}_{t+n-1})$$

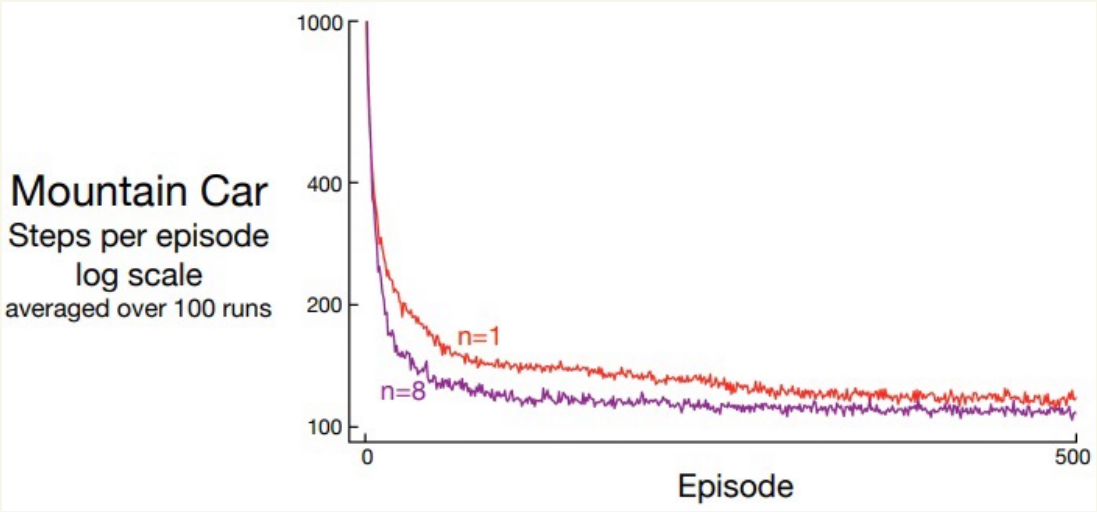
$$\vec{w}_{t+n} \equiv \vec{w}_{t+n-1} + \alpha \left[G_{t:t+n} - \hat{q}(S_t, A_t, \vec{w}_{t+n-1}) \right] \nabla \hat{q}(S_t, A_t, \vec{w}_{t+n-1})$$

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 $t = 0, 1, 2, \ldots$:
    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 $\tau = T - 1$

One-Step Sarsa vs 8-Step Sarsa!



Expected Sarsa to Q-Learning!

Expected Sarsa:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\Big(R_{t+1} + \gamma \sum_{a'} \pi(a' \mid S_{t+1}) \hat{q}(S_{t+1}, a', \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})\Big) \nabla \hat{q}(S_t, A_t, \mathbf{w})$$

Q-learning

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\Big(R_{t+1} + \gamma \max_{a'} \hat{q}(S_{t+1}, a', \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})\Big) \nabla \hat{q}(S_t, A_t, \mathbf{w})$$

10.3 - Average Reward:

Average reward is used for continuing problems without the need for discounting. The quality of a policy π is defined as the average reward while following that policy:

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

$$= \lim_{t\to\infty} E[R_t | S_0, A_{0:t-1} \sim \pi]$$

$$= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a) r$$

Differential Return:

In the average-reward setting, returns are defined in terms of differences between rewards and the average reward:

$$G_t \equiv R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) \ldots$$

This is the differential return

Differential TD Errors:

The differential form of the two TD errors.

$$\delta_t \equiv R_{t+1} - \bar{R}_t + \hat{v}(S_{t+1}, \vec{w}_t) - \hat{v}(S_t, \vec{w}_t)$$

$$\delta_t \equiv R_{t+1} - \bar{R}_t + \hat{q}(S_{t+1}, A_{t+1}, \vec{w}_t) - \hat{q}(S_t, A_t, \vec{w}_t)$$

$\bar{R}_t$: estimate of average reward $r(\pi)$ at t

Pseudo code:

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)
  $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \beta\delta$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
  $S \leftarrow S'$
  $A \leftarrow A'$