On-policy Control with Approximation

Outline

Introduction

- 이전 장의 On-policy Prediction 문제를 Control로 가져와 적용.
- 파라미터 기반의 행동 가치 함수 근사를 가중치 벡터 w 포함해 다룸.

$$\hat{q}(s, a, \mathbf{w}) \approx q_*(s, a)$$

- Semi-gradient TD(0)와 이와 연관된 On-policy TD의 Sarsa에 대해 다룸.
- 에피소딕 문제는 Semi-gradient TD(0) 확장이 가능.
- 연속적인 문제는 Average-reward를 적용해서 함수 근사를 적용.

10.1 Episodic Semi-gradient Control

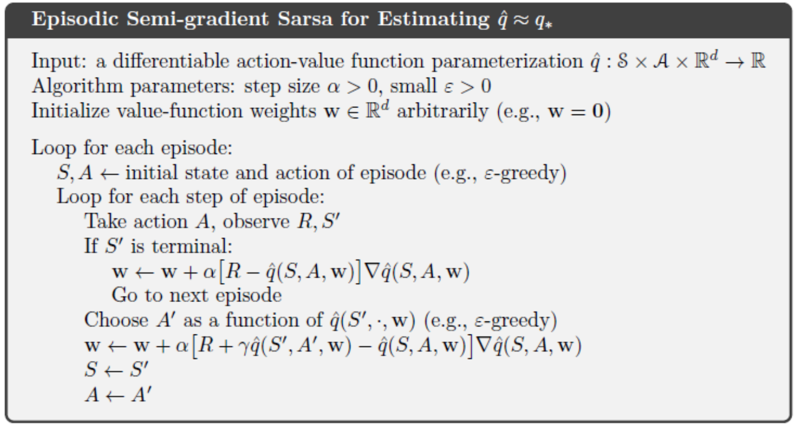
- 우선 Episodic 상황에서 Action-value 예측을 위한 General gradient-descent update는 다음과 같음.

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\Big[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)\Big]\nabla\hat{q}(S_t, A_t, \mathbf{w}_t).$$

- 이 Rule을 one-step Sarsa에 적용하면 다음과 같음(Episodic semi-gradient one-step Sarsa).

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\Big[R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)\Big]\nabla\hat{q}(S_t, A_t, \mathbf{w}_t).$$

10.1 Episodic Semi-gradient Control

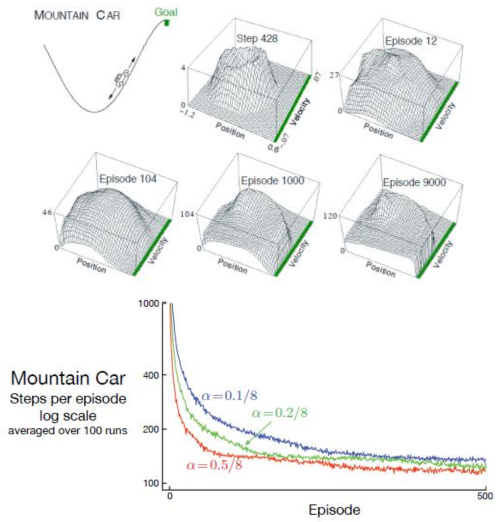


```
Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

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., $\varepsilon$-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., $\varepsilon$-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] Episodic Semi-gradient Sarsa for Estimating $\hat{q}(s, a, \mathbf{w}) \approx q_*(s, a)$

- 연속적인 행동이나 큰 규모의 이산 집합에 속하는 행동에 적절한 기법은 현재 연구 중.
- Action 집합이 이산적이고 크지 않다면 $\hat{q}(S_{t+1}, a, \mathbf{w}_t)$ 그렇다면 Greedy action $A^*_{t+1} = \arg\max_a \hat{q}(S_{t+1}, a, \mathbf{w}_t)$.
- 여기서 알고리즘 파라미터 α를 정해 제어.

10.1 Episodic Semi-gradient Control

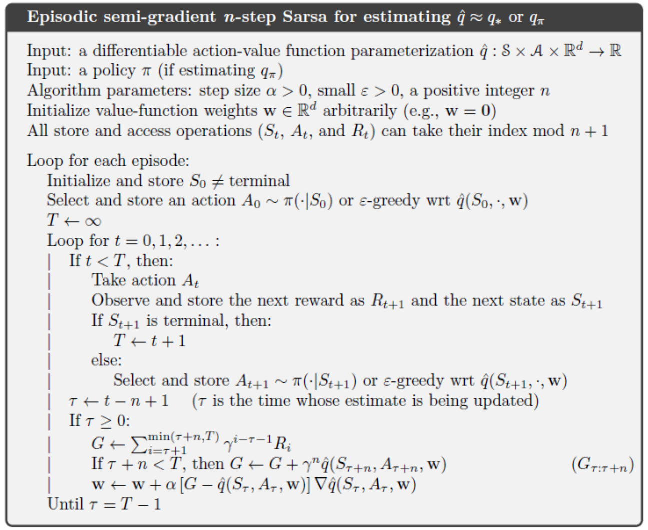


[2] Mountain Car Task

- 중력이 자동차 엔진보다 힘이 세서 경사면을 오르기 위한 가속에 어려움.
- 반대편 경사로를 따라 올라간 다음 전속력으로 전진하면 충분한 관성(시스템에 때로는 –의 행동을 해줘야 함).
- 이때 파라미터 벡터는 가중치 벡터와 함께 다음과 같이 선형적으로 결합되어 최적 action-value function을 찾게 됨.

$$\hat{q}(s,a,\mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s,a) = \sum_{i=1}^{d} w_i \cdot x_i(s,a),$$

10.2 Semi-gradient n-step Sarsa



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 $\pi$ (if estimating $q_\pi$)
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 $\varepsilon$-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 $\varepsilon$-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
  |  $\tau \leftarrow t - n + 1$    ($\tau$ 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 $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$        $(G_{\tau:\tau+n})$
  |    $\mathbf{w} \leftarrow \mathbf{w} + \alpha [G - \hat{q}(S_\tau, A_\tau, \mathbf{w})] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
  Until $\tau = T - 1$
```

- 10.1의 수식을 확장하여 n-step 형태의 Sarsa 알고리즘으로 확장.

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad t+n < T,$$

with $G_{t:t+n} \doteq G_t$ if $t + n \geq T$

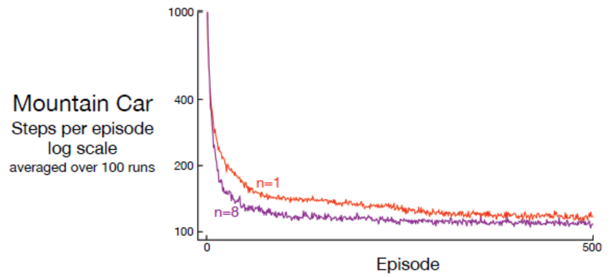
- n-step return이 알고리즘에 함께 적용.

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha \left[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})\right] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T$$

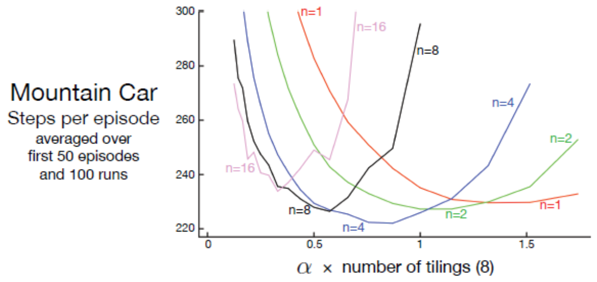
- 이를 통해 최종 가중치 벡터를 활용한 Sarsa는 위와 같음.

[3] Episodic Semi-gradient n-step Sarsa for Estimating $\hat{q}(s, a, \mathbf{w}) \approx q_*(s, a)$

10.2 Semi-gradient n-step Sarsa



[4.1] Mountain Car Task with n-step



[4.2] Mountain Car Task with n-step, α

- [4.1]을 통해 실제 n=8 step인 Sarsa가 중간 수준의 부트스트랩이 적용되면 학습 성능이 개선되는 것을 관찰할 수 있음.
- [4.2]를 통해 n 뿐 아니라, 하이퍼파라미터(알고리즘파라미터) α에 따라 복합적으로 학습 속도에 영향을 준다는 것을 알 수 있음.

10.3 Average Reward: A New Problem Setting for Continuing Tasks

- Approximation 상황에서의 Continuing tasks를 위한 Average reward를 설정.(Discount 대체)
- 정책 π의 성능이 정책을 따르는 동안 발생하는 보상 평균 비율이 Average reward.

$$\begin{aligned} r(\pi) &\doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi] \\ &= \lim_{t\to\infty} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi], \\ &= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) r, \end{aligned}$$

- MDP의 Ergodicity(시간에 대한 평균 = 상태 공간에 대한 평균)을 만족 → 극한 값 존재 증명.
- Steady-state distribution에서는 아래를 만족하는 분포.

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

10.3 Average Reward: A New Problem Setting for Continuing Tasks

- Average reward에서는 아래처럼 이득이 보상과 평균 보상 사이의 차이로 정의.(Differential return)

$$G_t \doteq R_{t+1} - r(\pi) \; + \; R_{t+2} - r(\pi) \; + \; R_{t+3} - r(\pi) \; + \; \cdots .$$

- 예전의 정의했던 Bellman expectation equation, Bellman optimality equation을 아래와 같이 Differential return 형태로 재정의할 수 있음.

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

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

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

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

10.3 Average Reward: A New Problem Setting for Continuing Tasks

- TD error도 다음과 같은 두가지 형태로 재정의할 수 있음.

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

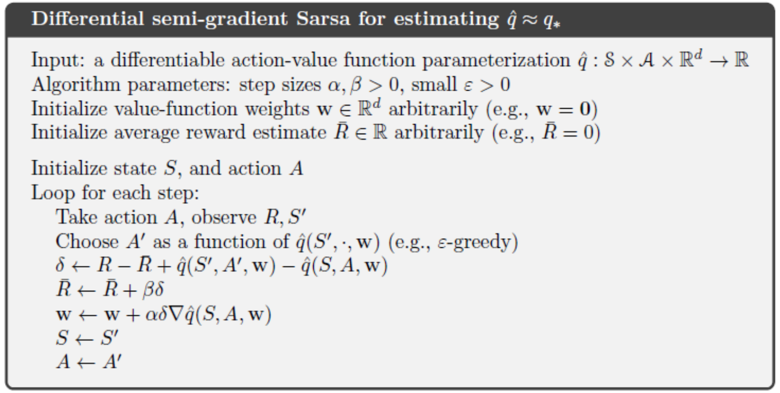
and

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

- 앞장에서 정의했던 Episodic semi-gradient one-step Sarsa를 TD error를 활용하여 다음과 같이 정의할 수 있다.(TD error는 위에서 얻음.)

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \nabla \hat{q}(S_t, A_t, \mathbf{w}_t),$$

10.3 Average Reward: A New Problem Setting for Continuing Tasks



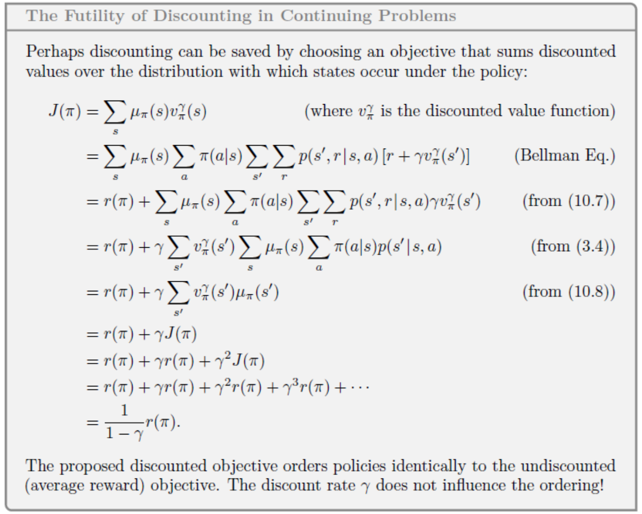
Differential semi-gradient Sarsa for estimating $\hat{q} \approx q_*$

```
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 $\alpha, \beta > 0$, small $\varepsilon > 0$
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., $\varepsilon$-greedy)
    $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$
```

[5] Differential Semi-gradient n-step Sarsa for Estimating $\hat{q}(s, a, \mathbf{w}) \approx q_*(s, a)$

- 결과적으로 Average reward를 활용한 Continuing tasks의 Differential semi-gradient one-step Sarsa는 왼쪽과 같다.

10.4 Deprecating the Discounted Setting



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!

[6] The Futility of Discounting in Continuing Problems

- 할인(Discounted)이 실제로 Approximation에서 Deprecating하다는 것을 증명을 통해 보여줌.
- 할인율은 Policy ordering에 영향을 주지 않음.

10.5 Differential Semi-gradient n-step Sarsa

- Differential semi-gradient n-step Sarsa 문제를 n단계 부트스트랩 문제로 일반화시키기 위해 다음과 같이 TD error의 확장이 필요. 이를 위해 Approximation 된 n단계 Differential 형태로 Return을 정의.

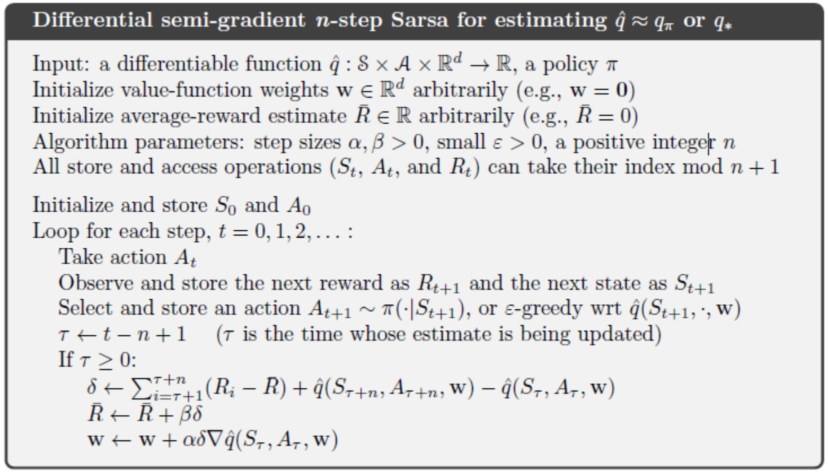
$$G_{t:t+n} \doteq R_{t+1} - \bar{R}_{t+n-1} + \cdots + R_{t+n} - \bar{R}_{t+n-1} + \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}),$$

where $\bar{R}$ is an estimate of $r(\pi)$, $n \geq 1$, and $t+n < T$. If $t+n \geq T$, then we define $G_{t:t+n} \doteq G_t$ as usual. The n-step TD error is then

- 결과적으로 n단계 TD error는 다음과 같이 정의.

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

10.5 Differential Semi-gradient n-step Sarsa



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 $\pi$
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 sizes $\alpha, \beta > 0$, small $\varepsilon > 0$, 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 $\varepsilon$-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
    $\tau \leftarrow t - n + 1$    ($\tau$ is the time whose estimate is being updated)
    If $\tau \geq 0$:
        $\delta \leftarrow \sum_{i=\tau+1}^{\tau+n}(R_i - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(S_\tau, A_\tau, \mathbf{w})$
        $\bar{R} \leftarrow \bar{R} + \beta\delta$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S_\tau, A_\tau, \mathbf{w})$
```

- 이러한 TD error를 가지고 Differential semi-gradient n-step Sarsa를 정의하면 왼쪽과 같음.

Summary

- Prediction(Value 학습)을 Control 문제(Policy 찾기)로 확장
- Episodic tasks의 경우 확장이 자연스럽게 가능.
- Continuing tasks의 경우 Average reward를 최대화하는 것을 통해 문제를 새롭게 구성해야 함.
- 또한 기존의 Discount가 Approximation에서는 무관하다는 것을 파악.
- 결과적으로 Continuing tasks의 경우 Differential return을 정의하고 이를 통해 Bellman equation에 적용하며, TD error와 함께 Differential semi-gradient Sarsa로 활용.

Thanks!

Any ***questions*** ?