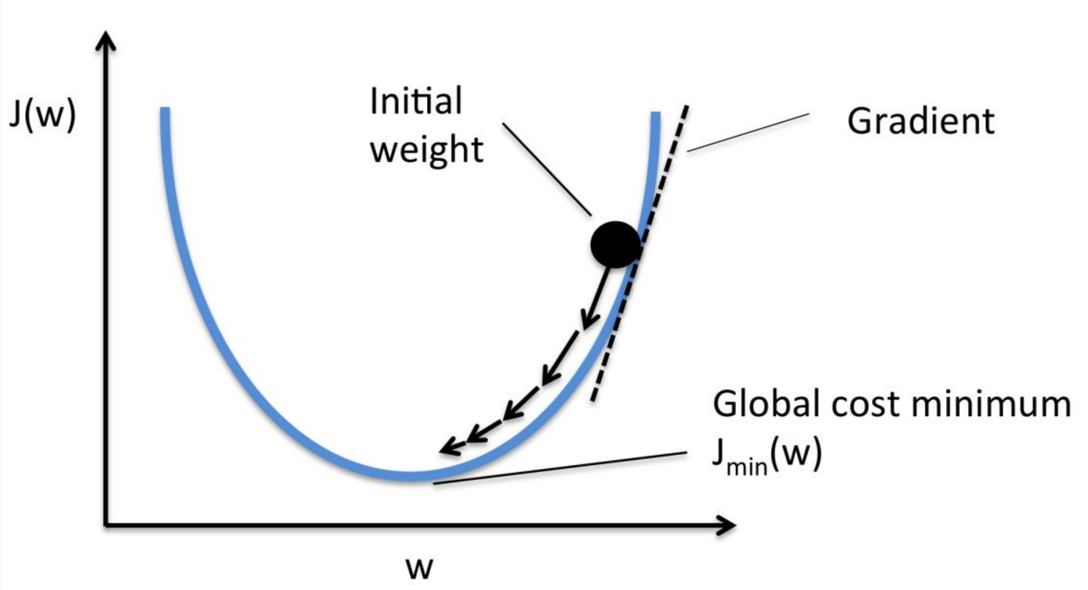
Chapter 06. 스스로 전략을 짜는 강화학습 (Reinforcement Learning)

Value Function Approximation



Markov Decision Process

MDP: MDP는 Markov reward process에 action이라는 요소가 추가된 모델로써, <S,A,P,R,γ>라는 tuple로 정의

Policy 정책 (π) : 정책은 각 상태 (s ∈ S)에 대해 Actions (a∈ A)에 대한 확률 분포 를 정의하는 함수

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

State Transition (P) : MDP가 주어진 π를 따를 때, s에서 s′으로 이동할 확률

$$p_{\pi}(s'|s) = \sum_{a \in A} \pi(a|s) p(s'|s,a) \qquad (9)$$

Reward(P) : s에서 얻을 수 있는 reward

$$r_{\pi}(s) = \sum_{a \in A} \pi(a|s) r(s,a) \qquad (10)$$

Tabular Method

Game Board:

Current state (s): 0 0 0
0 1 0

Q Table: $\gamma = 0.95$

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Tabular Method

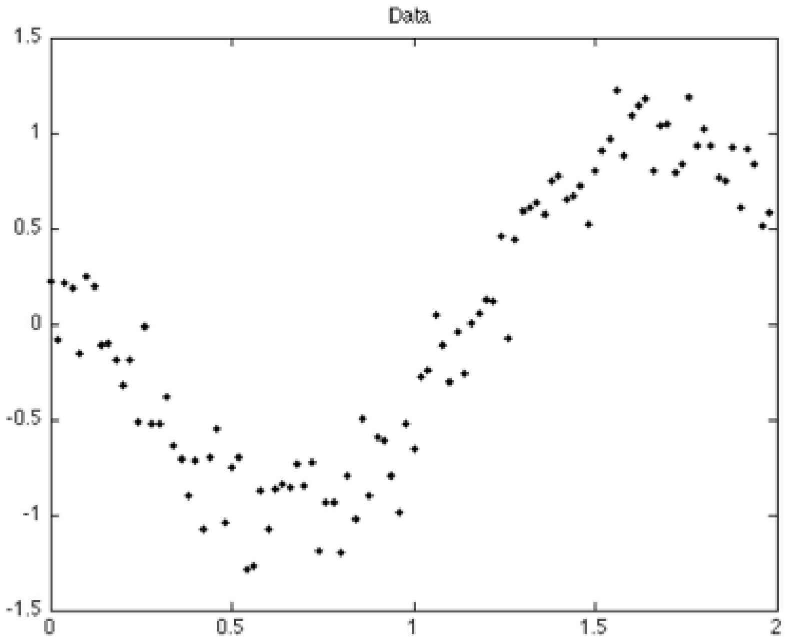
We have so far assumed that our estimates of value functions are represented as a table with one entry for each state or for each state-action pair. This is a particularly clear and instructive case, but of course **it is limited to tasks with small numbers of states and actions.** The problem is not just the memory needed for large tables, but the time and data needed to fill them accurately. In other words, **the key issue is that of generalization**

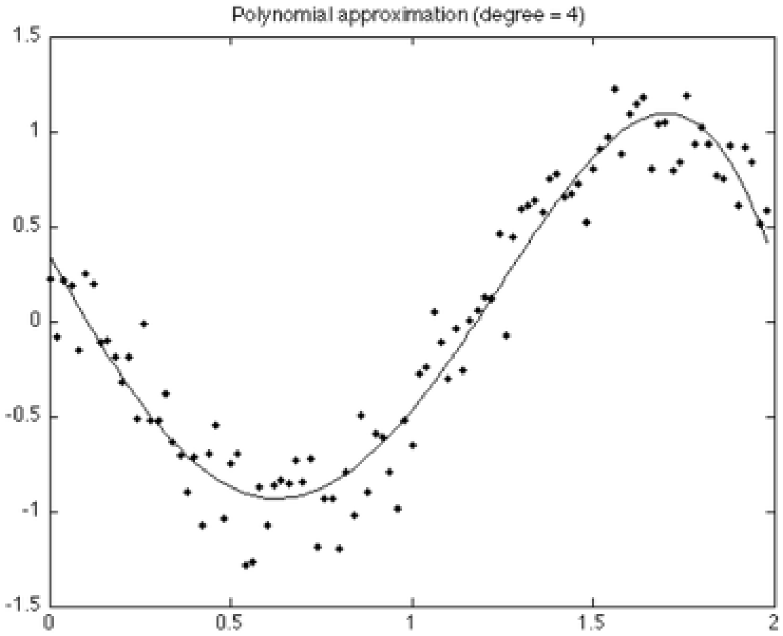
fast campus

Function Approximation

https://sumniya.tistory.com/17?category=781573

1) 실제로 가지고 있지 않은 data도 func.을 통해서 구할 수 있다
2) 실제 data의 noise를 배제하고 training할 수 있다
3) 고차원의 data도 효율적으로 저장이 가능하다





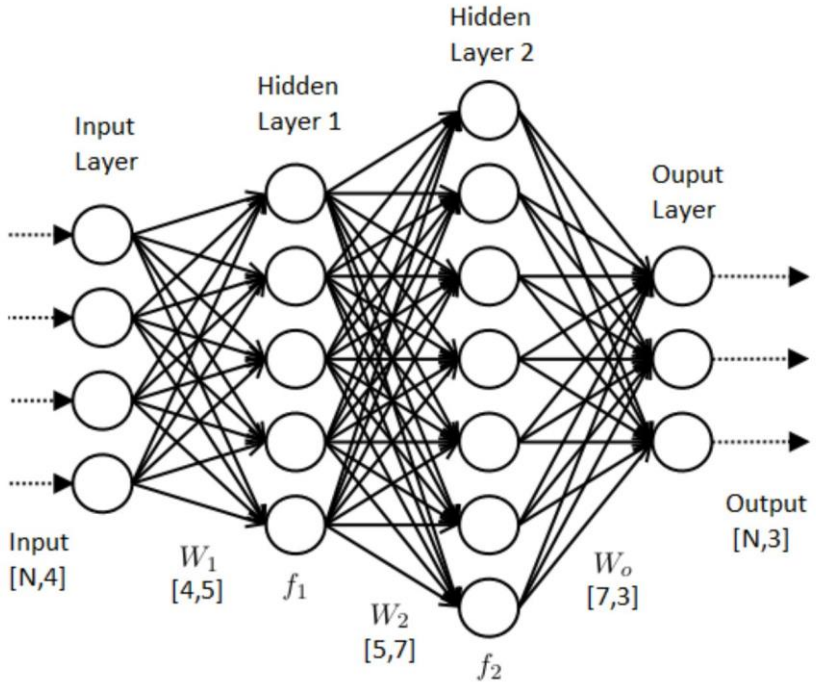
Function Approximation

https://sumniya.tistory.com/17?category=781573

$$ax^3+bx^2+cx+d \qquad parameter(a,b,c,d)$$

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

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

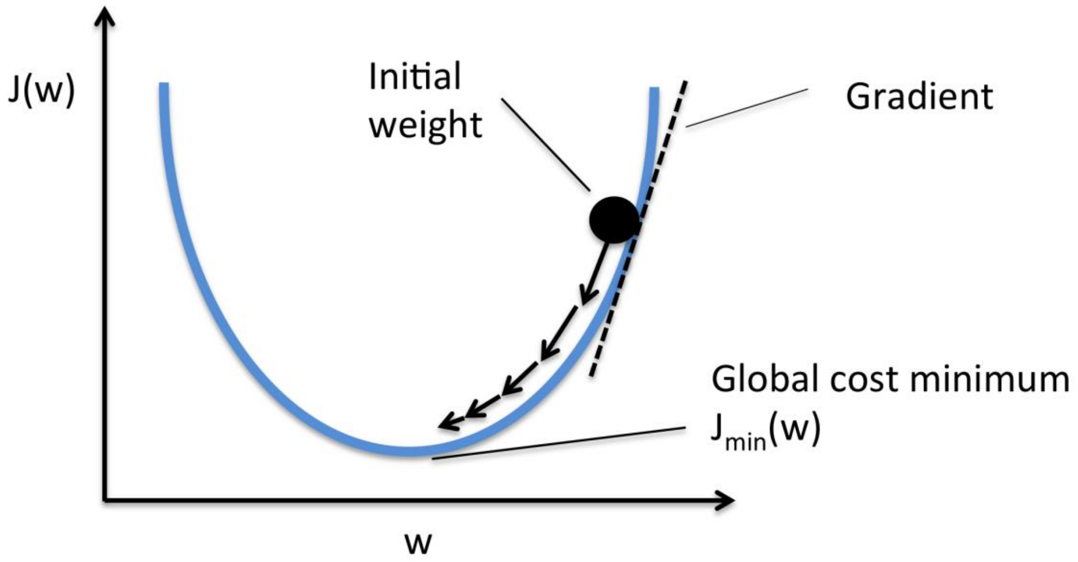


Function Approximation

Gradient Descent

$$\nabla_w J(w) = \begin{pmatrix} \frac{\partial J(w)}{\partial w_1} \\ \vdots \\ \frac{\partial J(w)}{\partial w_n} \end{pmatrix}$$

$$\Delta w = -\frac{1}{2}\alpha \nabla_w J(w)$$



Function Approximation

state-value function

$$J(w) = \mathrm{E}_\pi[\{v_\pi(s) - \hat{v}(s, \mathbf{w})\}^2]$$

$$\Delta w = -\frac{1}{2}\alpha\nabla_w J(w)$$

$$= \alpha\mathrm{E}_\pi[(v_\pi(s) - \hat{v}(s, \mathbf{w}))\nabla_w \hat{v}(s, \mathbf{w})]$$

$$For\ \ MC, \qquad \Delta w = \alpha\mathrm{E}_\pi[(G_t - \hat{v}(s, \mathbf{w}))\nabla_w \hat{v}(s, \mathbf{w})]$$

$$For\ \ TD(0), \qquad \Delta w = \alpha\mathrm{E}_\pi[(R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(s, \mathbf{w}))\nabla_w \hat{v}(s, \mathbf{w})]$$

Function Approximation

action-value function

$$loss, \quad J(w) = \mathrm{E}_{\pi}[\{q_{\pi}(S, A) - \hat{q}(S, A, \mathbf{w})\}^2]$$

$$gradient\ descent, \quad \Delta w = -\frac{1}{2}\alpha\nabla_{w}J(w) = \alpha\mathrm{E}_{\pi}[(q_{\pi}(S, A) - \hat{q}(S, A, \mathbf{w}))\nabla_{w}\hat{q}(S, A, \mathbf{w})]$$

$$For\ MC, \quad \Delta w = \alpha\mathrm{E}_{\pi}[(G_t - \hat{q}(S_t, A_t, \mathbf{w}))\nabla_{w}\hat{q}(S_t, A_t, \mathbf{w})]$$

$$For\ TD(0), \quad \Delta w = \alpha\mathrm{E}_{\pi}[(R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, w) - \hat{q}(S_t, A_t, \mathbf{w}))\nabla_{w}\hat{q}(S_t, A_t, \mathbf{w})]$$

Function Approximation

action-value function

$$\text{For the forward-view } TD(\lambda), \quad \Delta w = \alpha E_\pi[(q_t^\lambda - \hat{q}(S_t, A_t, \mathbf{w}))\nabla_w \hat{q}(S_t, A_t, \mathbf{w})]$$

$$\text{For the backward-view } TD(\lambda), \quad \Delta w = \alpha E_\pi[\delta_t E_t]$$

$$s.t. \quad \delta_t = R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, w) - \hat{q}(S_t, A_t, \mathbf{w})$$

$$E_t = \gamma\lambda E_{t-1} + \nabla_w \hat{q}(S_t, A_t, \mathbf{w})$$

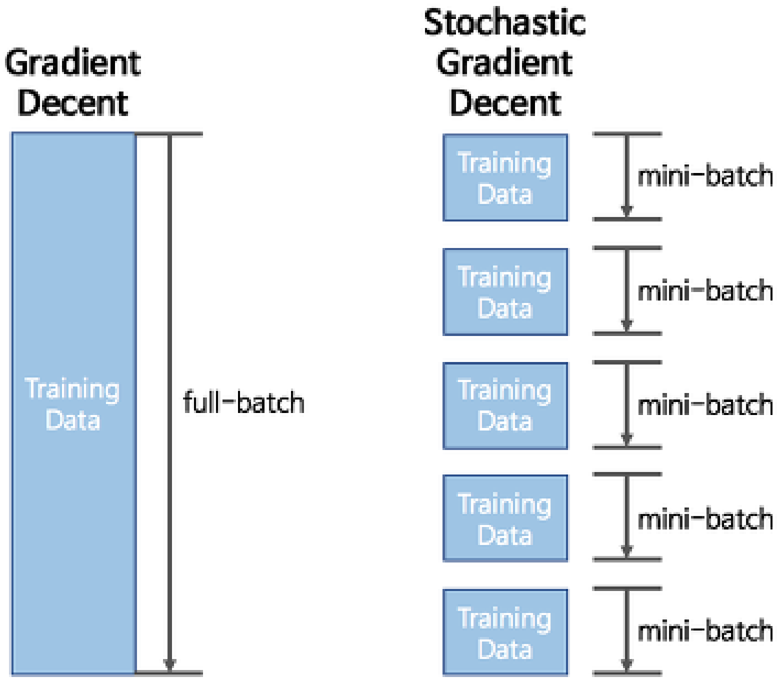
fast campus

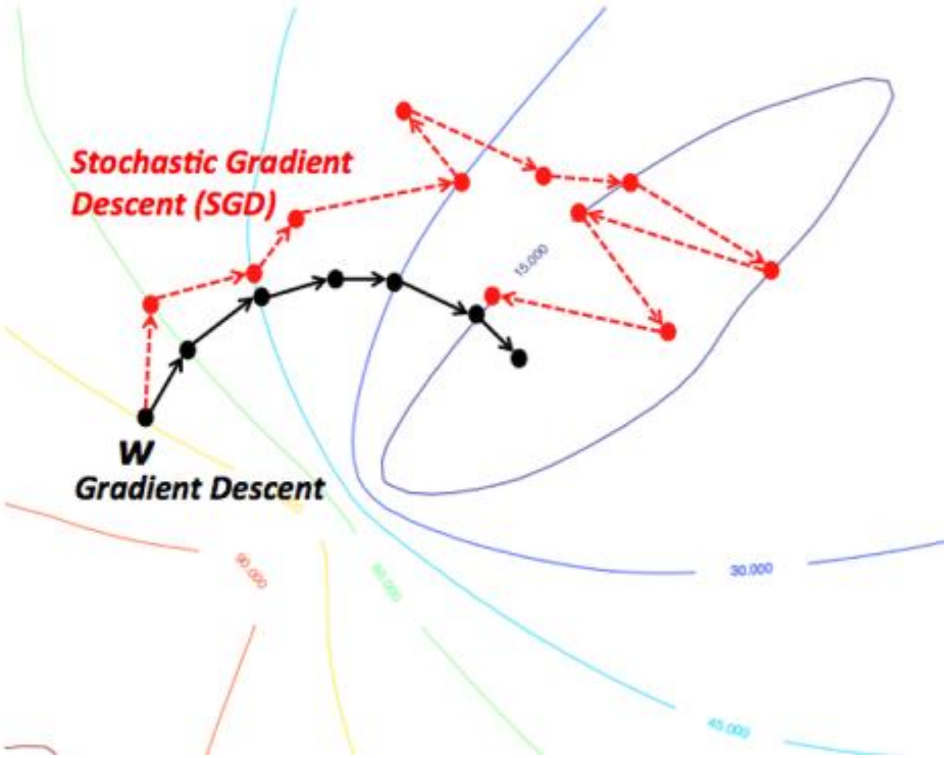
Function Approximation

https://seamless.tistory.com/38

Stochastic gradient descent(SGD)

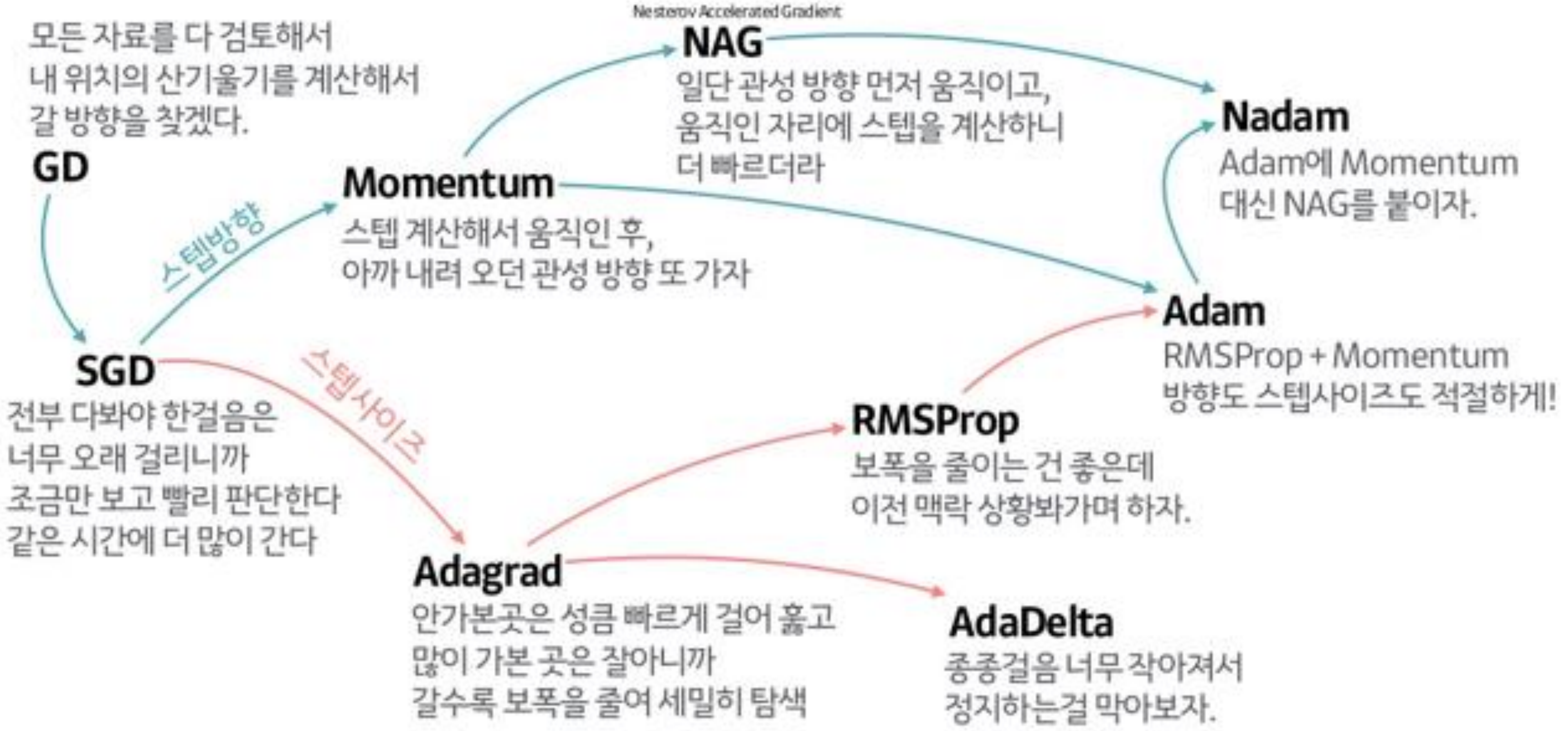
$$J(w) = \{v_\pi(s) - \hat{v}(s, w)\}^2, \quad for\ s \in S$$





Function Approximation

https://seamless.tistory.com/38



Function Approximation

Monte Carlo Value Function Approximation

- Return G_t is an unbiased but noisy sample of the true expected return $V^\pi(s_t)$
- Therefore can reduce MC VFA to doing supervised learning on a set of (state,return) pairs: $\langle s_1, G_1\rangle, \langle s_2, G_2\rangle, \ldots, \langle s_T, G_T\rangle$ ← estimate of $V^\pi(s_T)$
  - Substitute G_t for the true $V^\pi(s_t)$ when fit function approximator
- Concretely when using linear VFA for policy evaluation

episodic

$$
\begin{aligned}
\Delta \mathbf{w} &= \alpha(G_t - \hat{V}(s_t;\mathbf{w}))\nabla_{\mathbf{w}}\hat{V}(s_t;\mathbf{w}) \\
&= \alpha(G_t - \hat{V}(s_t;\mathbf{w}))\mathbf{x}(s_t) \\
&= \alpha(G_t - \underbrace{\mathbf{x}(s_t)^T\mathbf{w}})\mathbf{x}(s_t)
\end{aligned}
$$

- Note: G_t may be a very noisy estimate of true return

Function Approximation

Temporal Difference (TD(0)) Learning with Value Function Approximation

- In value function approximation, target is $r + \gamma \hat{V}^\pi(s'; \mathbf{w})$, a biased and approximated estimate of the true value $V^\pi(s)$
- Can reduce doing TD(0) learning with value function approximation to supervised learning on a set of data pairs:
  - $\langle s_1, r_1 + \gamma \hat{V}^\pi(s_2; \mathbf{w}) \rangle, \langle s_2, r_2 + \gamma \hat{V}(s_3; \mathbf{w}) \rangle, \ldots$
- Find weights to minimize mean squared error

$$J(\mathbf{w}) = \mathbb{E}_\pi[(r_j + \gamma \hat{V}^\pi(s_{j+1}, \mathbf{w}) - \hat{V}(s_j; \mathbf{w}))^2]$$

Function Approximation

Incremental Model-Free Control Approaches

- Similar to policy evaluation, true state-action value function for a state is unknown and so substitute a target value
- In Monte Carlo methods, use a return G_t as a substitute target

$$\Delta \mathbf{w} = \alpha(G_t - \hat{Q}(s_t, a_t; \mathbf{w}))\nabla_{\mathbf{w}}\hat{Q}(s_t, a_t; \mathbf{w})$$

- For SARSA instead use a TD target $r + \gamma\hat{Q}(s', a'; \mathbf{w})$ which leverages the current function approximation value

$$\Delta \mathbf{w} = \alpha(r + \gamma\hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w}))\nabla_{\mathbf{w}}\hat{Q}(s, a; \mathbf{w})$$

- For Q-learning instead use a TD target $r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w})$ which leverages the max of the current function approximation value

$$\Delta \mathbf{w} = \alpha(r + \gamma \max_{a'} \underbrace{\hat{Q}(s', a'; \mathbf{w})}_{X(s',a')w} - \hat{Q}(s, a; \mathbf{w}))\nabla_{\mathbf{w}}\hat{Q}(s, a; \mathbf{w})$$

$X(s,a)w$

- *Thank you*