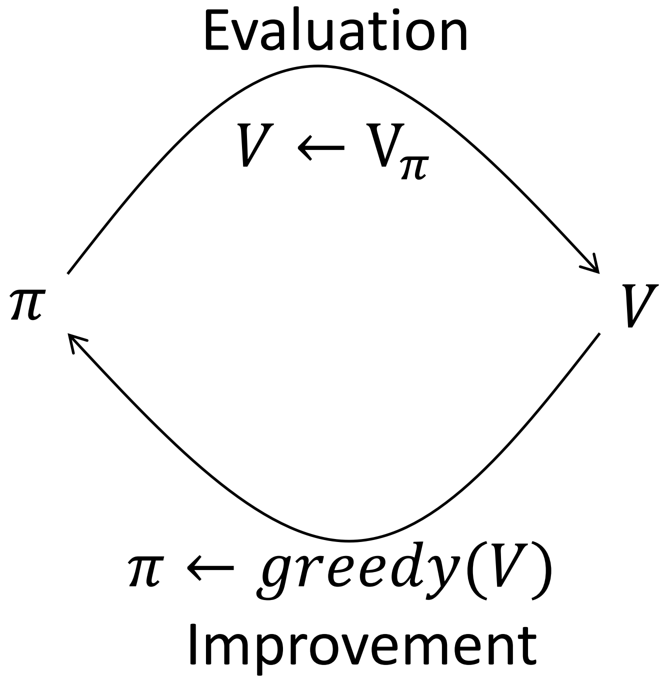
ECE 414/517 Reinforcement Learning

LECTURE 10: MONTE CARLO METHODS

SEP. 27 2022

Generalized Policy Iteration

1. Policy iteration –
   - Full policy evaluation (up until convergence)
   - Policy improvement
2. Value Iteration –
   - One iteration of policy evaluations
   - Policy Improvement
3. Asynchronous Dynamic Programming
   - Policy evaluation/policy improvement even further interleaved.
   - For example, updating only a single step before performing policy improvement
   - Can be done stochastically, as long as all states are visited.



Monte Carlo Policy Evaluation

1. We still need to estimate $V_\pi(s)$. Estimate from experience:
   - Average all returns observed after visiting a given state.
   - Each occurrence of state s in an episode is called a visit to s
2. Generate an episode following π:
$$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T,$$
3. First visit MC:
   - Method averages just the returns following first visits to s
   - Easier to show convergence. Why?
4. Every Visit MC:
   - Method averages the returns following all the visits to s

Monte Carlo Policy Evaluation

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

First Visit Vs. Every Visit

Assume we have three state *X,Y,T* for a non-discounted episodic MDP

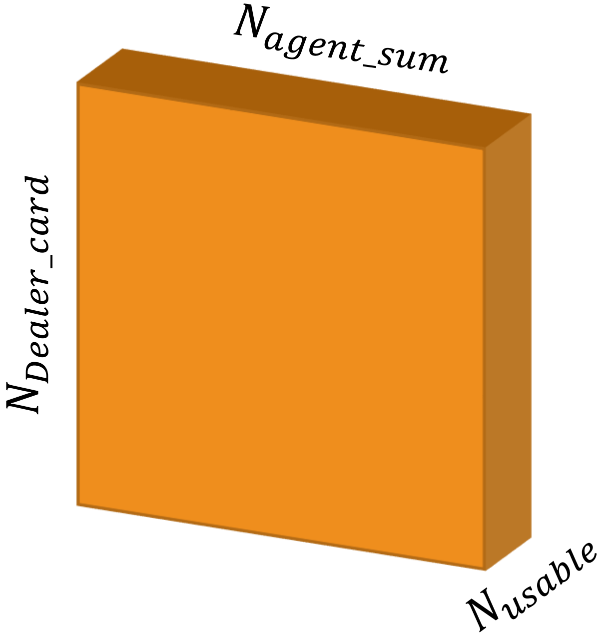
We produce the following episodes:

After all episodes have ran what will the return arrays look like for first visit/every visit? What will be V(s)?

s_0	a_0	r_1	s_1	a_1	r_2	s_2	a_2	r_3	s_3
X		*1*	*Y*		*3*	*Y*		*2*	*T*
X		*0*	*X*		*2*	*Y*		*0*	*T*
Y		*2*	*X*		*2*	*X*		*1*	*T*
Y		*1*	*Y*		*4*	*X*		*0*	*T*

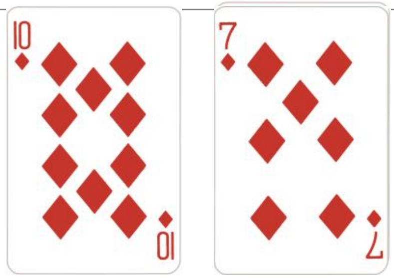
Blackjack - MDP

1. EpisodicStates?
   - $N_{usable} = \pm 1$
   - $N_{Dealer\_card} = 1 \ldots 10$
   - $N_{agent\_sum} = (12 \ldots 21)$
   - $N_{usable} \times N_{Dealer\_card} \times N_{agent\_sum} = 2 \times 10 \times 10 = 200\ (+1)$
2. Actions?
   - $\{hit, stand\}$
3. Reward?
   - $win: +1$
   - $lose: -1$
4. State value function: array shape?



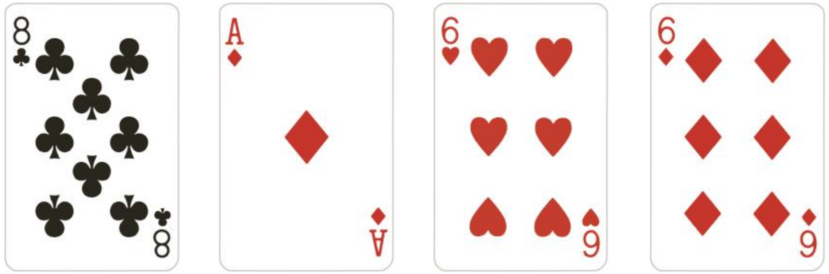
Policy Evaluation Example

Consider a policy that only stands if the player’s sum is 20 or 21.



Consider the following cases:

player cards:[11, 8] dealer shows:
10
player hits:6
player hits:6
player holds
dealer sum: 17
reward: 1



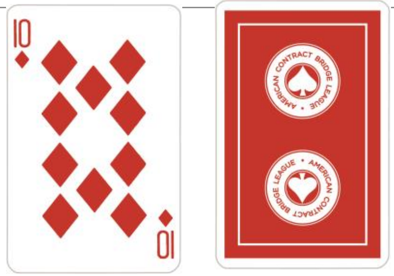
What values will change and to what?

Policy Evaluation Example

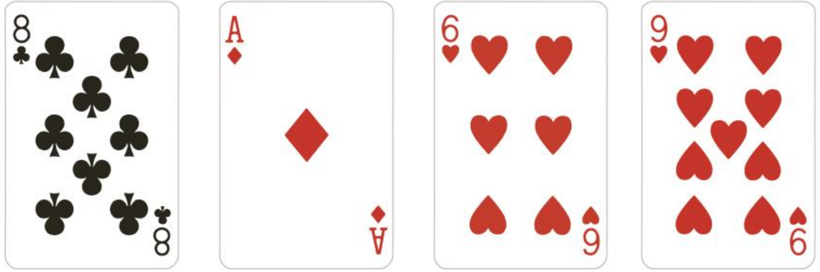
Consider a policy that only stands if the player’s sum is 20 or 21.

Use first visit or every visit?

Consider the following cases:



player cards:[11, 8] dealer shows:
10
player hits:6
player hits:9
bust!



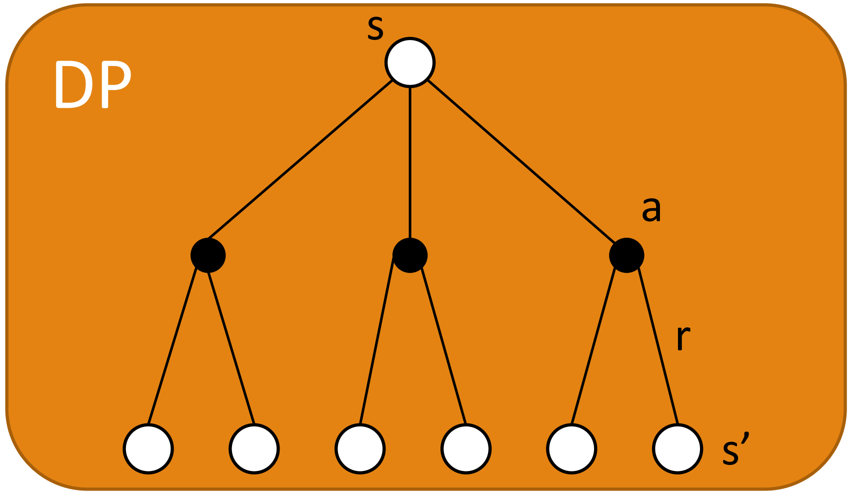
What values will change and to what?

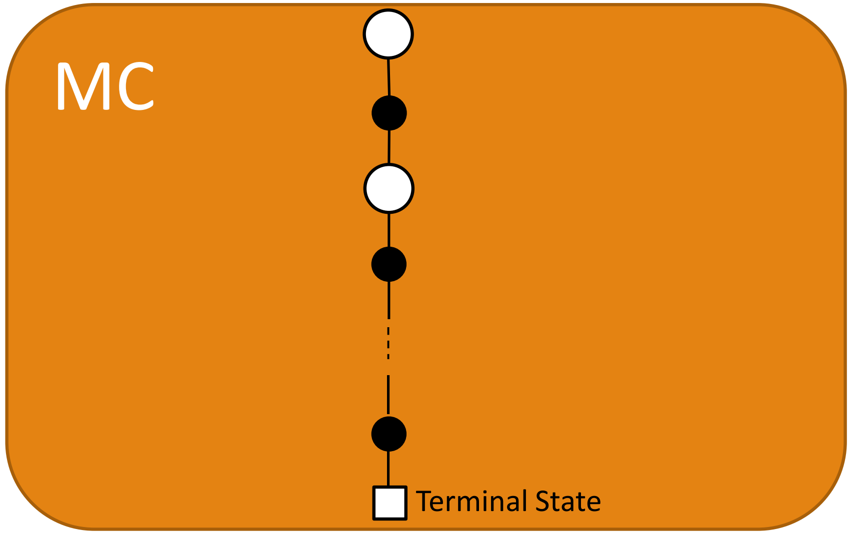
Blackjack – Use DP?

1. Although we have complete knowledge of the environment, it would not be easy to apply DP policy evaluation.
2. Hard to calculate $p(s', r|s, a)$
   - e.g. player's sum is 14 and dealer's displayed card is jack – what is the expected reward?
3. Since all of these calculations must be done prior to running DP – it is often an impractical approach
4. In contrast, generating sample games (for MC) is much easier to do
5. Surprising insight: even if the environment's dynamics are known, MC is often a more efficient method to apply
   - Estimating values are independent (no "bootstrapping") – why helpful?
   - Optimal policy trajectory corresponds to a small state subset

Monte Carlo Vs. Dynamic Programming

1. Can the backup diagram be applied to MC methods?
   - Recall that backup diagrams show top node to be updated and below all the transitions and leaf nodes that contribute to the update
2. For MC the root is a state node and below are all the nodes visited until terminal node is reached.
   - Shows only transitions sampled on one episode.
3. DP focuses on one-step transitions, whereas MC goes all the way to the end of the episode.
4. Updating Values for states is independent in MC.
   - computational complexity of updating one node is independent of $|s|$
   - An attractive feature since one can estimate only a subset of the node values (not have to do all)





Now what?

1. We described a way to find $V^{\pi}(s)$. Now what?
2. Since we do not have model, cannot easily do policy improvement: find the new greedy π
$$\pi'(s) = arg\max_a \sum_{s',r} p(s',r\,|s,a)[r + \alpha V^{\pi}(s')]$$
3. Instead we should estimate $Q^{\pi}(s,a)$.
$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$
4. Use the same MC approach. The agent records the rewards received after taking action a at state s
   - Problem?
5. Need to force exploration or some actions will never be chosen.
   - How?
6. One way: **exploring starts**. Each state-action pair at the beginning of an episode has a non zero probability.
   - Will consider the general stochastic approach later.

Monte Carlo Control

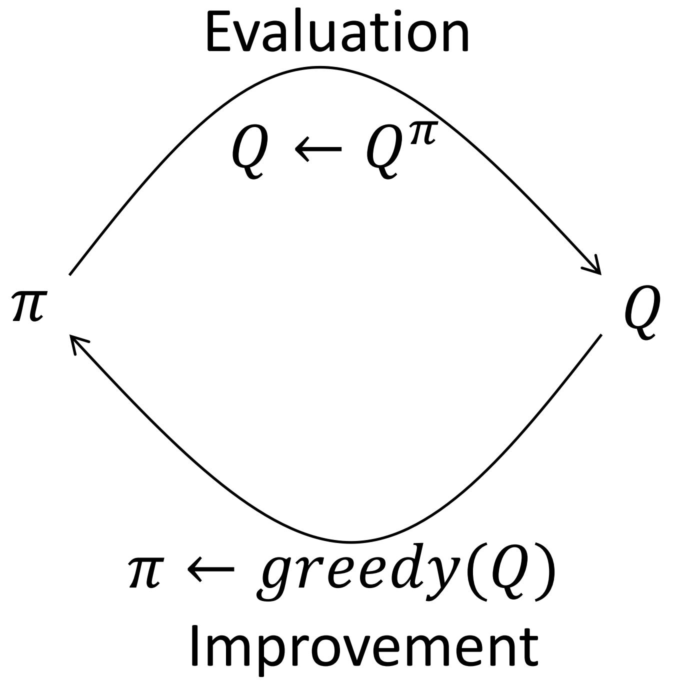
How do we use the policy evaluation step to find the optimal policy?

1. Policy Evaluation
   - Achieved by averaging returns over many outcomes

2. Policy improvement
   - Policy improvement is done by selecting greedy actions, i.e.:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

$$\pi_0 \xrightarrow{E} Q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} Q_{\pi_*}$$



Monte Carlo Control

1. The policy improvement theorem holds since for all s:

$$\begin{aligned}
Q^{\pi_k}\big(s, \pi_{k+1}(s)\big) &= Q^{\pi_k}(s, \underset{a}{arg\max}\, Q^{\pi_k}(s,a)) \\
&= \max_a Q^{\pi_k}(s,a) \\
&\geq Q^{\pi_k}\big(s, \pi_k(s)\big) \\
&= V^{\pi_k}(s)
\end{aligned}$$

2. If the two policies are equal, they are both optimal
   - This way MC can lead to optimal policies with no model
3. Assumes exploring starts and infinite number of episodes for MC policy evaluation
4. Latter not really needed (similar to DP):
   - Update only to a given level of performance
   - Don't solve for Q^{π_k} but move towards it
   - Alternate between evaluation and improvement per episode.

Monte Carlo (Exploring Starts)

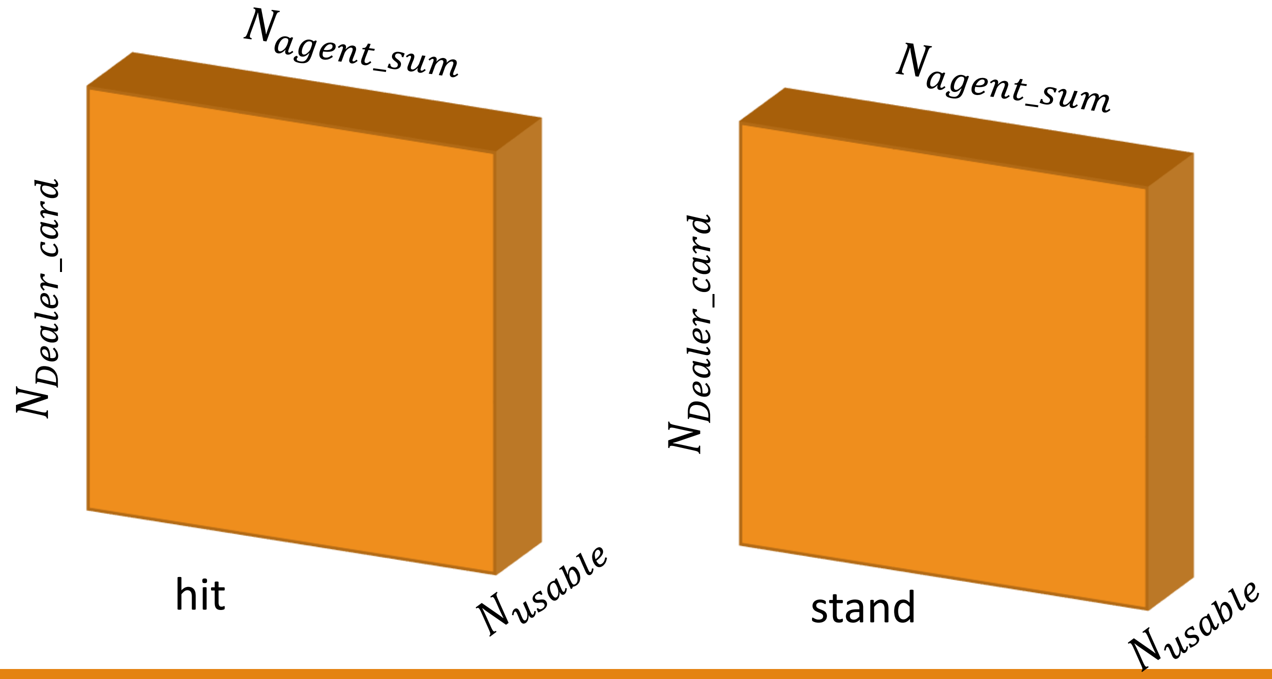
Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
- $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
- Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ such that all pairs have probability > 0
- Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append G to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

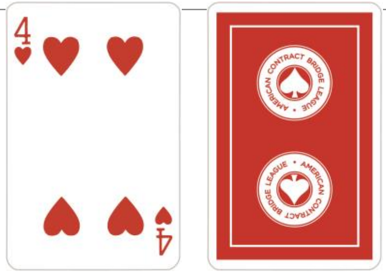
Blackjack - MDP

1. EpisodicStates?
   - $N_{usable} = \pm 1$
   - $N_{Dealer\_card} = 1 \dots 10$
   - $N_{agent\_sum} = (12 \dots 21)$
   - $N_{usable} \times N_{Dealer\_card} \times N_{agent\_sum} = 2 \times 10 \times 10 = 200\ (+1)$
2. Actions?
   - $\{hit, stand\}$
3. Reward?
   - $win: +1$
   - $lose: -1$
4. Acttion value function: array shape?

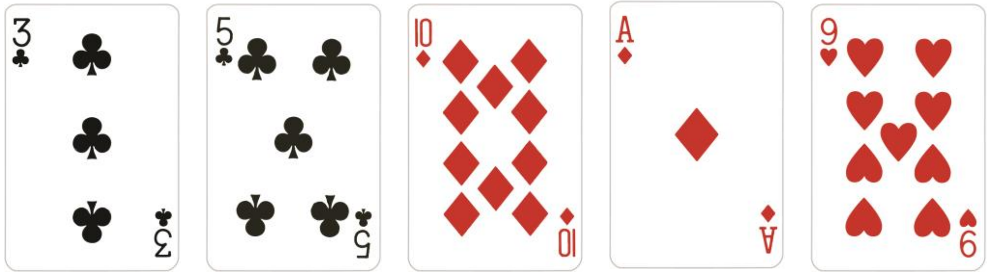


Exploring Starts Example

Consider the following cases:



player cards:[5, 3] dealer shows: 4
player hits:10
player hits:1
player hits:9
bust!



What values will change and to what?

No Exploring Starts

1. Exploring starts not always possible.
   - Why?
2. On the other hand can't simply follow greedy approach – no exploration.
3. We can employ an ϵ-greedy policy instead, where $\pi(s,a) > 0 \quad \forall \quad s,a$:
   - With probability $1-\epsilon$ choose the greedy action
   - With probability ϵ – explore uniformly
4. Probabilities of actions:
   - For non max actions: $\frac{\epsilon}{|\mathcal{A}(s)|}$
   - For max action: $1-\epsilon+\frac{\epsilon}{|\mathcal{A}(s)|}$

Policy Improvement under ϵ-greedy

Show that the policy improvement theory holds.

$$\begin{aligned}
Q^{\pi_k}\big(s, \pi_{k+1}(s)\big) &= \sum_a \pi_{k+1}(a|s)\, Q^{\pi_k}(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^{\pi_k}(s,a) + (1-\epsilon)\max_a Q^{\pi_k}(s,a) \\
&\geq \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^{\pi_k}(s,a) + (1-\epsilon)\sum_a \frac{\pi_k(a|s) - \frac{\epsilon}{|\mathcal{A}(s)|}}{1-\epsilon} Q^{\pi_k}(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^{\pi_k}(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^{\pi_k}(s,a) + \pi_k(a|s) Q^{\pi_k}(s,a) \\
&= V^{\pi_k}(s)
\end{aligned}$$

On policy Monte Carlo

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

On-policy Vs. Off-policy

1. So far all the methods we discussed are **on-policy methods,** where you evaluate and improve the policy that is used to create the episodes.
   - To ensure all actions are selected we need to employ an ϵ-soft policy
2. Instead we can use **off-Policy methods.** In this case the policy we use to generate the episodes is different than the behavior we are trying to evaluate and improve.
   - Behavior policy $b(a|s)$ – The policy used to generate episodes
   - Estimation policy $\pi(a|s)$ – The policy being evaluated and improved.
3. This has some advantages:
   - Include On-policy as special case ($b(a|s) = \pi(a|s)$)
   - Estimation policy can be deterministic (e.g. greedy) and therefore optimal vs near optimal ($\epsilon-$soft).
   - Can learn from observing the behavior of other non-learning agents (such as humans)
4. Disadvantages:
   - More complex and slower to converge.