Module IV: Monte Carlo Method

By: Dimple Bohra

Contents

- Model Based Vs Model Free Learning
- Monte Carlo Method
- Monte Carlo Prediction
- Advantages of Monte Carlo over Dynamic Programing,
- Monte Carlo Control
- Blackjack Problem
- on-policy, off policy
- Incremental Monte Carlo
- Issues/Assumptions in Monte Carlo Methods
- Solution of Blackjack using Monte Carlo Method

Model Based Vs Model Free Learning

- We know that dynamic programming is used to solve problems where the underlying model of the environment is known beforehand (or more precisely, model-based learning).
- Reinforcement Learning is all about learning from experience in playing games. And yet, in none of the dynamic programming algorithms, we learned through experience.
- We had a full model of the environment, which included all the state transition probabilities.
- However, in most real-life situations, the transition probabilities from one state to another (or the so-called model of the environment) are not known beforehand. It is not even necessary that the task follows a Markov property.

Model Based Vs Model Free Learning

- Let's say we want to train a bot to learn how to play chess. Consider converting the chess environment into an MDP.
- Now, depending on the positioning of pieces, this environment will have many states as well as a large number of possible actions. The model of this environment is almost impossible to design!
- One potential solution could be to repeatedly play a complete game of chess and receive a positive reward for winning, and a negative reward for losing, at the end of each game. **This is called learning from experience.**

Monte Carlo Methods

- Monte Carlo methods require only experience. Meaning, they sample states, actions, and rewards, while interacting with the environment.
- They are a way to solve RL problems based on averaging sample returns.
- MC is typically used for episodic tasks (if we would have a continuous task, it will be difficult to compute the average).
- Once an episode ends, the value estimates and policies change.

Monte Carlo Methods

- In model free learning, we don’t have the transition function p(s’, r | s, a).
- Instead, we update the states by averaging the returns we experience while traveling through those states.
- Basically, until now we knew all the transitions (s → s’) so we could just update a state value by calculating them.
- Now however, we have to actually explore, starting from state “s”, see what the next state and action look like from experience (i.e. sample a state and action), and update that state “s” value by averaging the results as we’re exploring.

Monte Carlo Methods

- Monte Carlo methods sample and average returns for each state–action pair much like the bandit methods.
- The main difference is that now there are multiple states, each acting like a different bandit problem (like an associative-search or contextual bandit) and that the different bandit problems are interrelated.
- That is, the return after taking an action in one state depends on the actions taken in later states in the same episode.
- Because all the action selections are undergoing learning, the problem becomes nonstationary from the point of view of the earlier state.

Monte Carlo Methods

- To handle the Non stationarity, we adapt the idea of general policy iteration (GPI). Whereas there we computed value functions from knowledge of the MDP, here we learn value functions from sample returns with the MDP.
- The value functions and corresponding policies still interact to attain optimality in essentially the same way (GPI).

Monte Carlo Prediction

- We've seen that the value of a state is the expected return starting from that state and then following a particular policy.
- An easy way to estimate it based on experience, would be to average the returns we observe after visiting a state.
- As we interact with the environment more and observe more returns, the average should converge to the expected value.
- That's the idea behind all Monte Carlo methods.

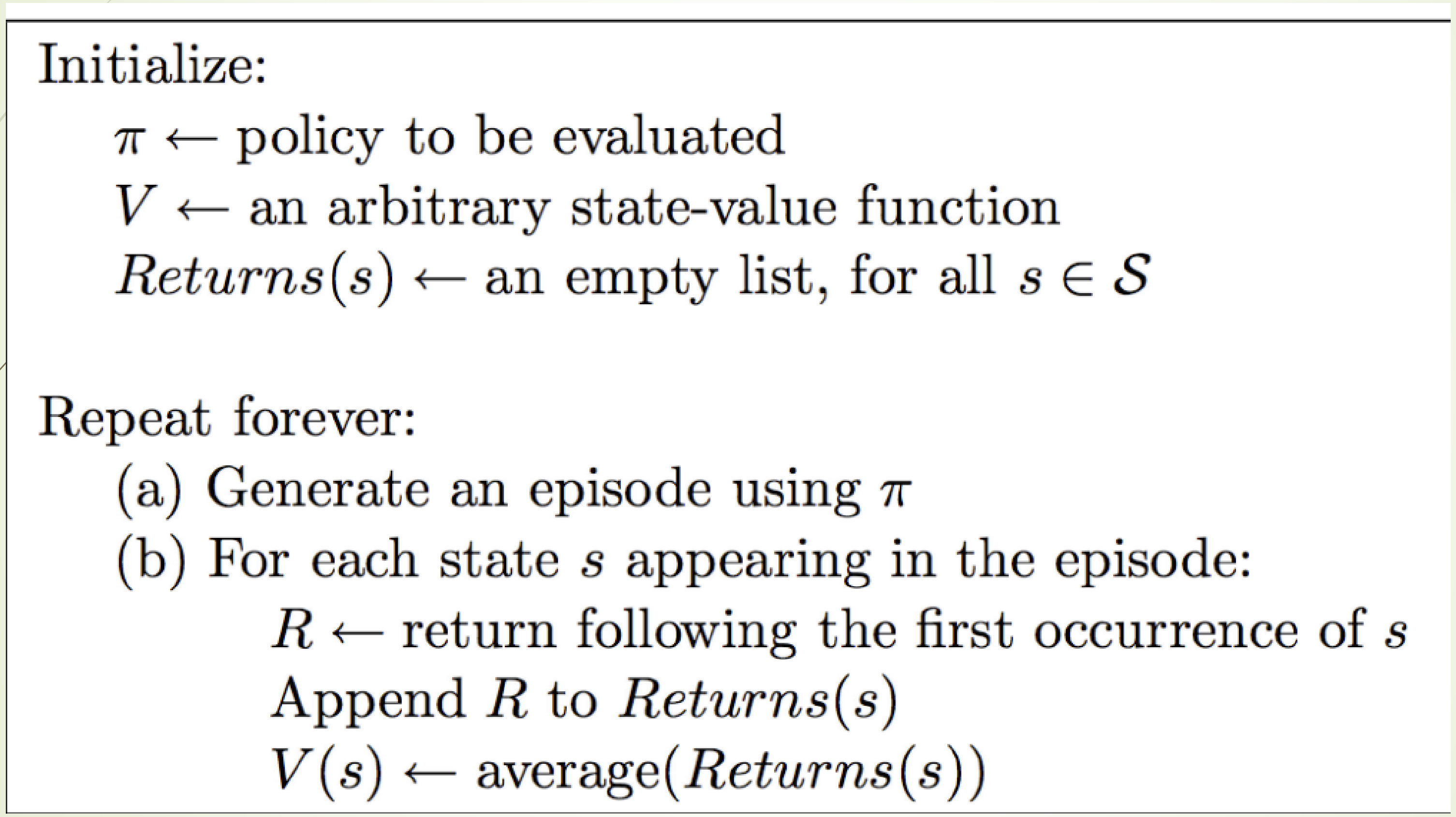
Monte Carlo Prediction

- Suppose we want to estimate the value of a state $V\pi(s)$.
- Each time we visit state "s" in an episode is called a "visit to s". Clearly, we may visit "s" many times in a single episode. Let's call the first time we visit "s" in an episode the "first visit to s".
- In **first-visit MC**, we estimate the value of state "s" by averaging the returns we observe after our first visit to "s".
- In **every-visit MC**, we estimate the value of "s" by averaging the returns after all visits to "s".

Monte Carlo Prediction

- These two Monte Carlo (MC) methods are very similar but have slightly different theoretical properties.
- First-visit MC has been most widely studied .
- Every-visit MC extends more naturally to function approximation

First Visit MC



```
Initialize:
    $\pi \leftarrow$ policy to be evaluated
    $V \leftarrow$ an arbitrary state-value function
    $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:
    (a) Generate an episode using $\pi$
    (b) For each state $s$ appearing in the episode:
            $R \leftarrow$ return following the first occurrence of $s$
            Append $R$ to $Returns(s)$
            $V(s) \leftarrow$ average($Returns(s)$)
```

$$\bar{V}_{\pi}(s) = \frac{1}{N} \sum_{i=1}^{N} G_{i,s}$$

- i – Episode index
- s – Index of state

Policy Evaluation using First Visit MC

- Suppose there's an environment where we have 2 states – A and B.
- Let's say we observed 2 sample episodes:

$$A+3 \rightarrow A+2 \rightarrow B-4 \rightarrow A+4 \rightarrow B-3 \rightarrow \text{terminate}$$

$$B-2 \rightarrow A+3 \rightarrow B-3 \rightarrow \text{terminate}$$

First Visit MC: summing all the rewards coming after A (after first visit to A) for all episodes. So cannot have more than one summation term per episode.
e.g.

For 1st Episode: = 3 + 2 +(-4) +4+(–3) = 2
For 2nd Episode: = 3+(-3)=0

$$\boxed{\bar{V}_{\pi}(s) = \frac{1}{N}\sum_{i=1}^{N} G_{i,s}}$$

V(A)= (2+0)/2 = 1

Policy Evaluation using Every Visit MC

- Suppose there's an environment where we have 2 states – A and B.
- Let's say we observed 2 sample episodes:

$$A+3 \rightarrow A+2 \rightarrow B-4 \rightarrow A+4 \rightarrow B-3 \rightarrow \text{terminate}$$

$$B-2 \rightarrow A+3 \rightarrow B-3 \rightarrow \text{terminate}$$

Every Visit MC: summing all the rewards coming after every occurrence of A (after every visit to A) for all episodes. So can have more than one summation terms per episode.
e.g.

For 1st Episode: = [3 + 2 +(-4) +4+(–3)]**+**[2+(-4)+4+(-3)]**+**[4+(-3)] = 2+(-1)+1
For 2nd Episode: = 3+(-3)=0

$$\boxed{\bar{V}_{\pi}(s) = \frac{1}{N}\sum_{i=1}^{N} G_{i,s}}$$

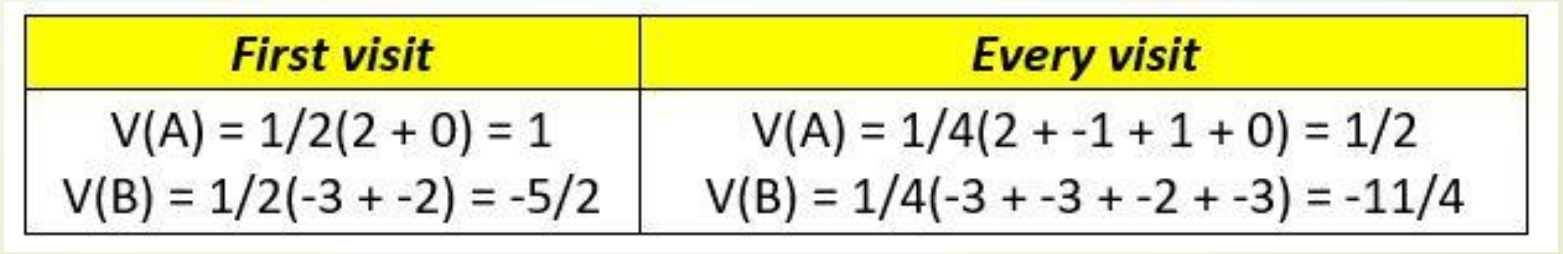
V(A)= (2+(-1)+1+0)/4 = ½ = 0.5

Policy Evaluation using First Visit MC and Every Visit MC

- Let's consider a simple example
- Suppose there's an environment where we have 2 states – A and B.
- Let's say we observed 2 sample episodes:

$$A + 3 \rightarrow A + 2 \rightarrow B - 4 \rightarrow A + 4 \rightarrow B - 3 \rightarrow \text{terminate}$$

$$B - 2 \rightarrow A + 3 \rightarrow B - 3 \rightarrow \text{terminate}$$



First visit	***Every visit***
V(A) = 1/2(2 + 0) = 1 V(B) = 1/2(-3 + -2) = -5/2	V(A) = 1/4(2 + -1 + 1 + 0) = 1/2 V(B) = 1/4(-3 + -3 + -2 + -3) = -11/4

Blackjack Problem

- Goal of Blackjack problem is to beat the dealer.
- The objective of the popular casino card game of blackjack is to obtain cards the sum of whose numerical values is as great as possible without exceeding 21.
- All face cards count as 10, and an ace can count either as 1 or as 11. We consider the version in which each player competes independently against the dealer.

Blackjack Problem

The game begins with two cards dealt to both dealer and player.

One of the dealer's cards is face up and the other is face down.

If the player has 21 immediately (an ace and a 10-card), it is called a natural. He then wins unless the dealer also has a natural, in which case the game is a draw.

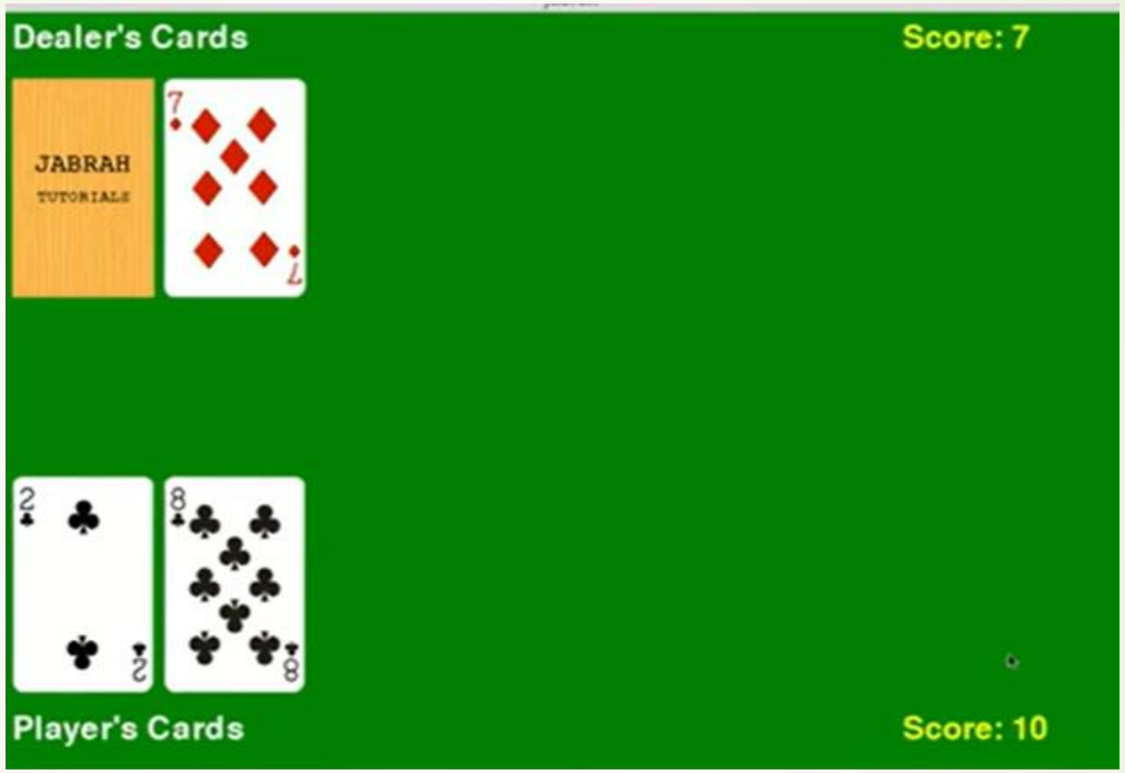
If the player does not have a natural, then he can request additional cards, one by one (hits), until he either stops (sticks) or exceeds 21 (goes bust).

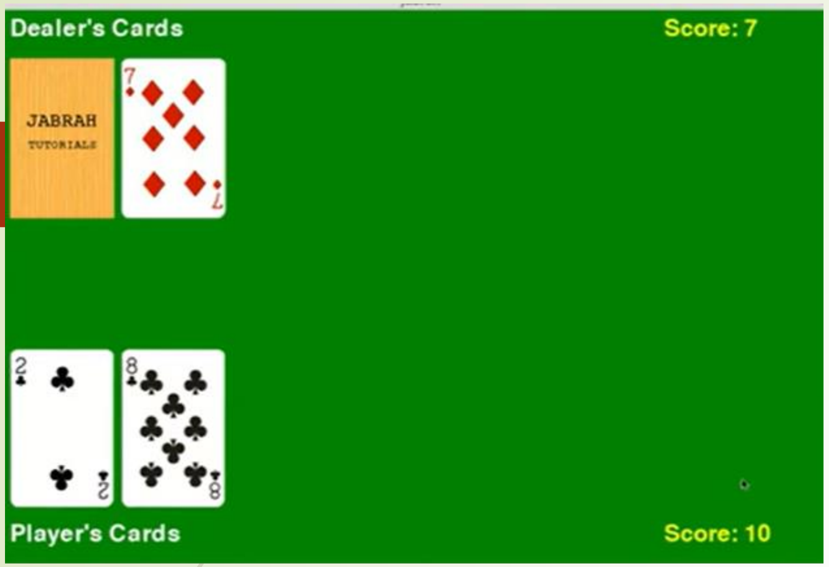
If he goes bust, he loses; if he sticks, then it becomes the dealer's turn.

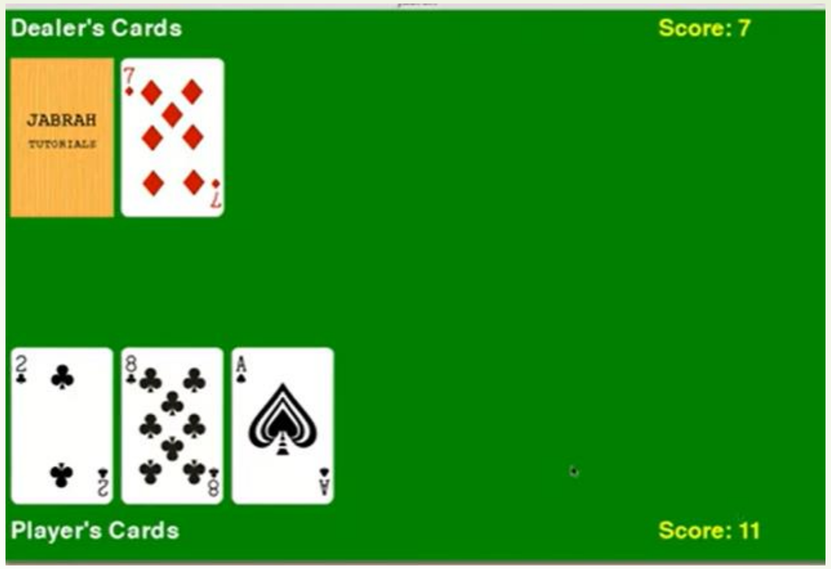
The dealer hits or sticks according to a fixed strategy without choice: he sticks on any sum of 17 or greater, and hits otherwise.

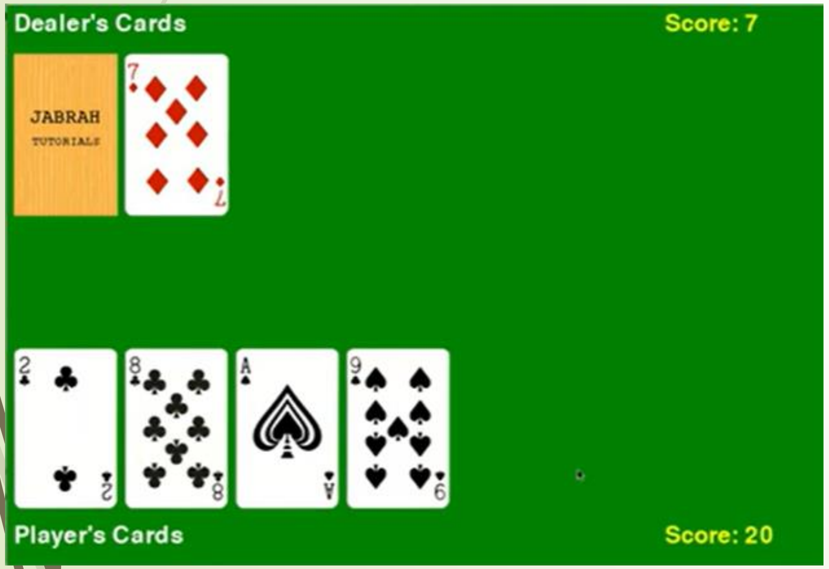
If the dealer goes bust, then the player wins; otherwise, the outcome—win, lose, or draw—is determined by whose final sum is closer to 21.

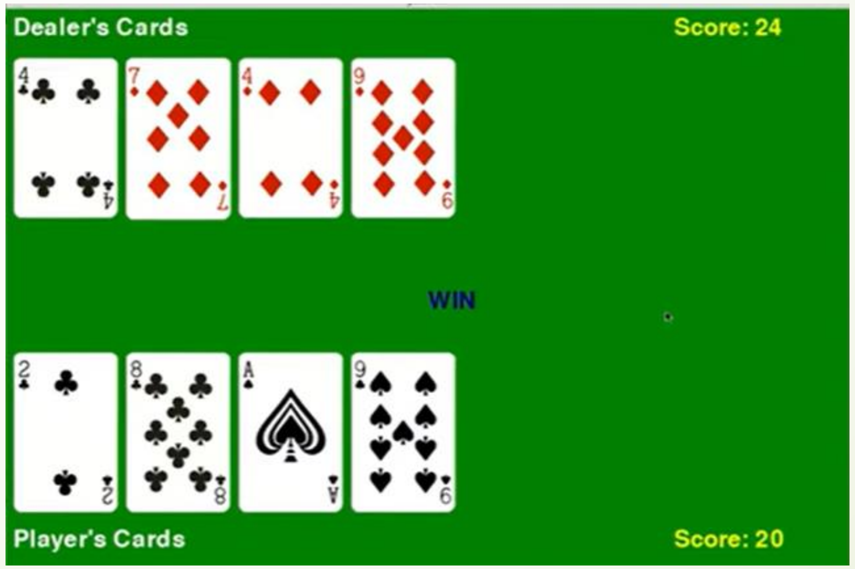
Game 1


Dealer's Cards
Score: 7
JABRAH
TUTORIALS
Player's Cards
Score: 10

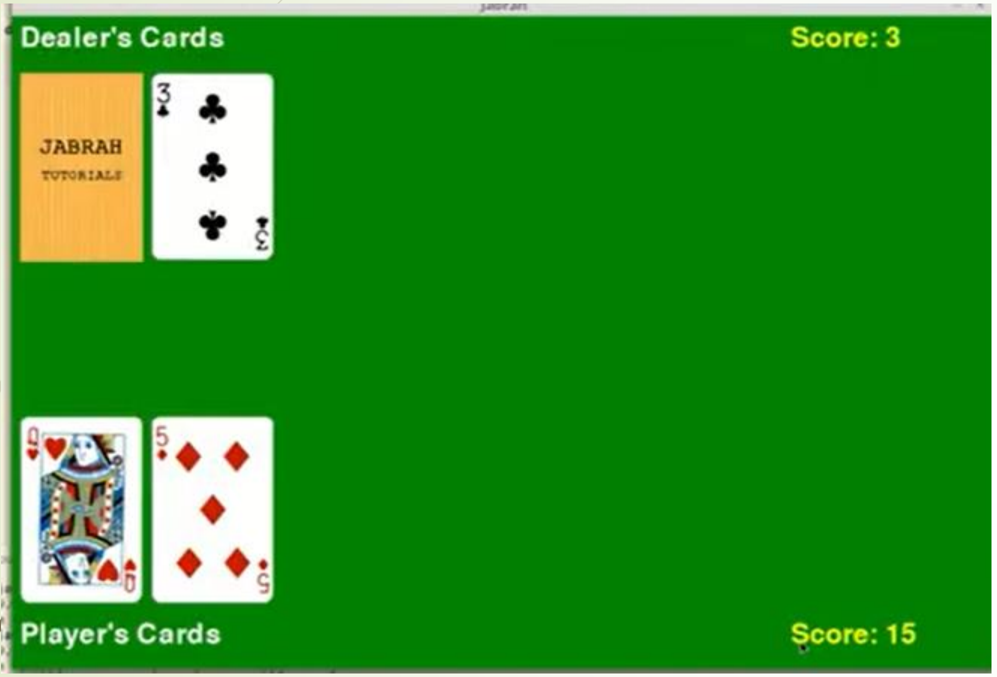

Dealer's Cards
Score: 7
JABRAH
TUTORIALS
Player's Cards
Score: 10


Dealer's Cards
Score: 7
JABRAH
TUTORIALS
Player's Cards
Score: 11

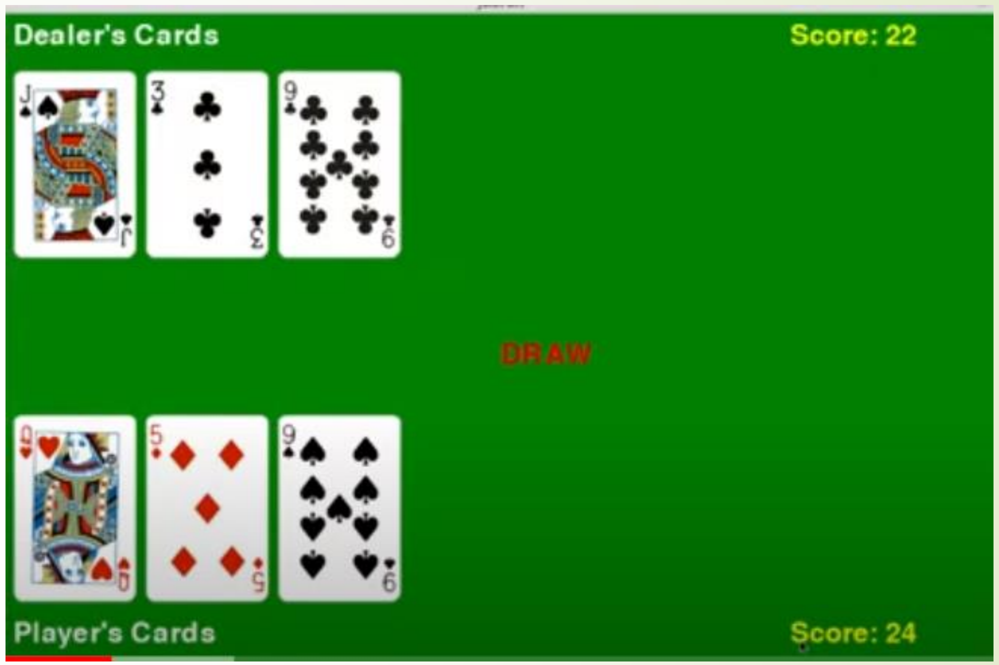

Dealer's Cards
Score: 7
JABRAH
TUTORIALS
Player's Cards
Score: 20


Dealer's Cards
Score: 24
WIN
Player's Cards
Score: 20

Game 2


Dealer's Cards Score: 3

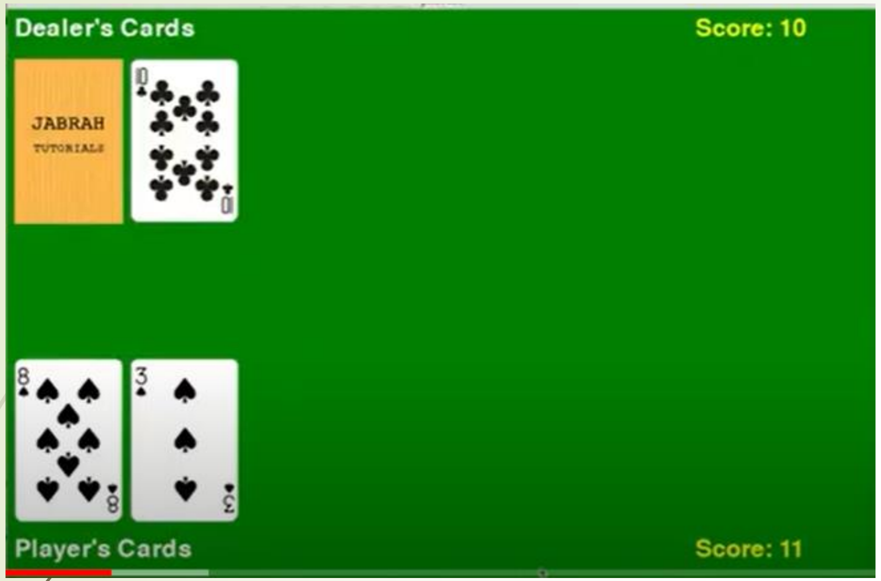
Player's Cards Score: 15

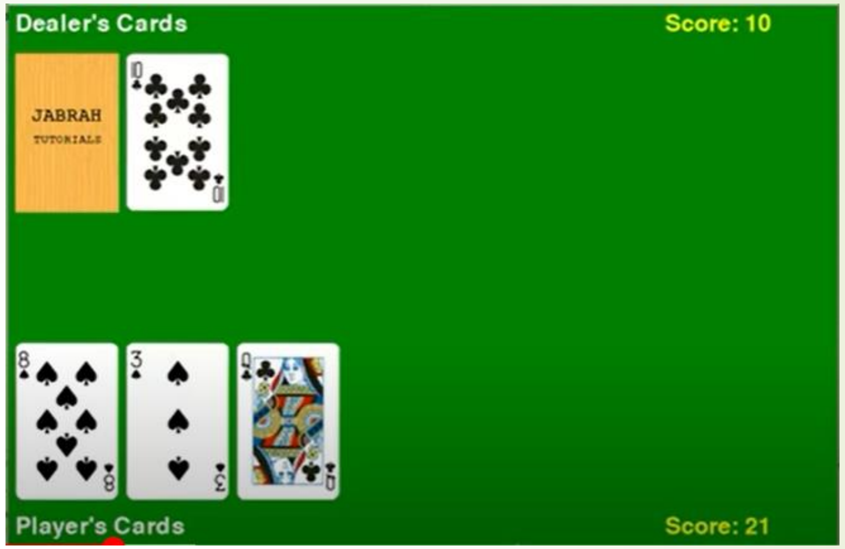

Dealer's Cards Score: 22

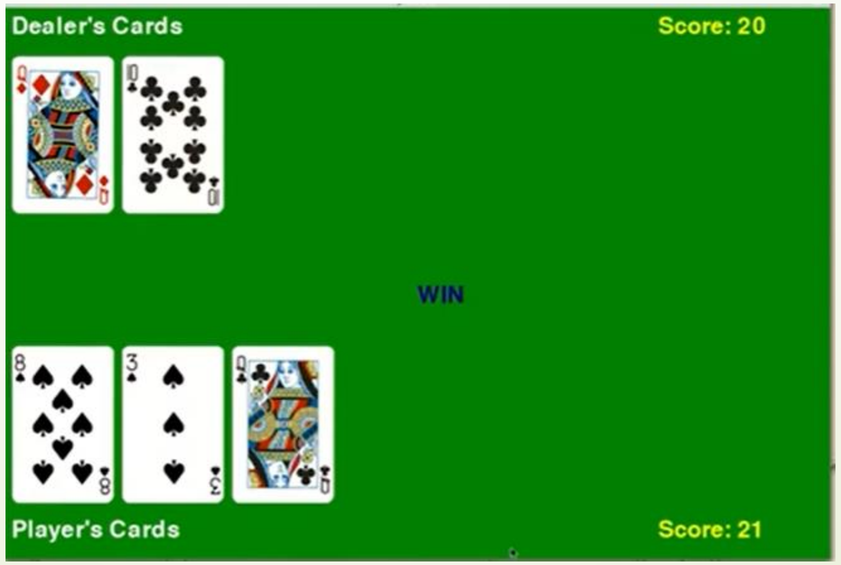
DRAW

Player's Cards Score: 24

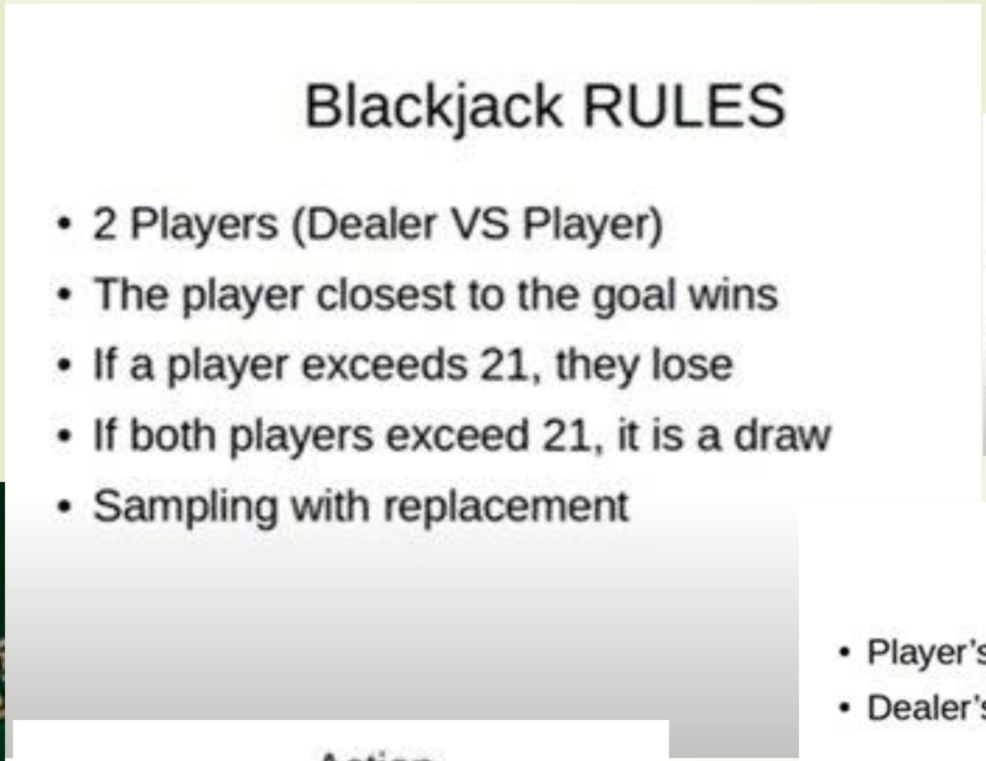
Game 3


Dealer's Cards
Score: 10
JABRAH
TUTORIALS
Player's Cards
Score: 11


Dealer's Cards
Score: 10
JABRAH
TUTORIALS
Player's Cards
Score: 21

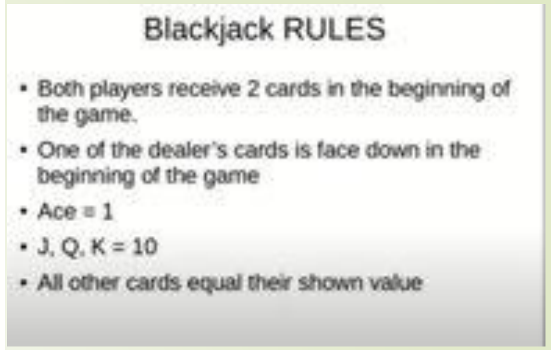

Dealer's Cards
Score: 20
WIN
Player's Cards
Score: 21

Blackjack Problem



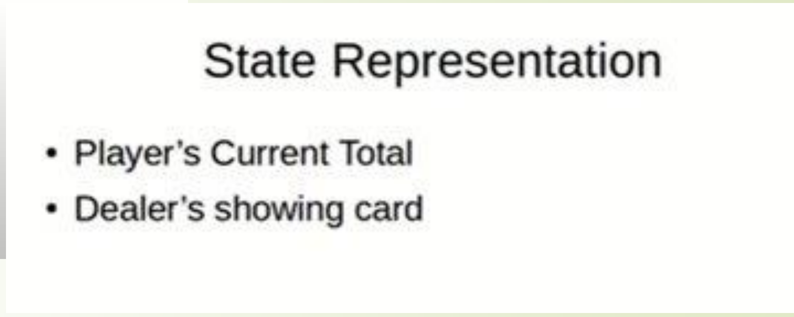
Blackjack RULES

- 2 Players (Dealer VS Player)
- The player closest to the goal wins
- If a player exceeds 21, they lose
- If both players exceed 21, it is a draw
- Sampling with replacement



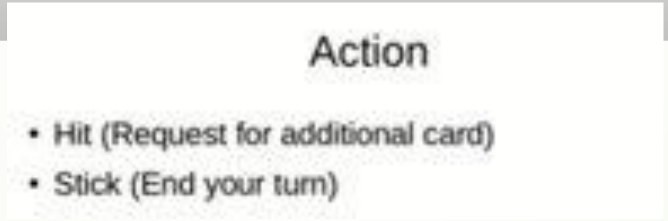
Blackjack RULES

- Both players receive 2 cards in the beginning of the game.
- One of the dealer's cards is face down in the beginning of the game
- Ace = 1
- J, Q, K = 10
- All other cards equal their shown value



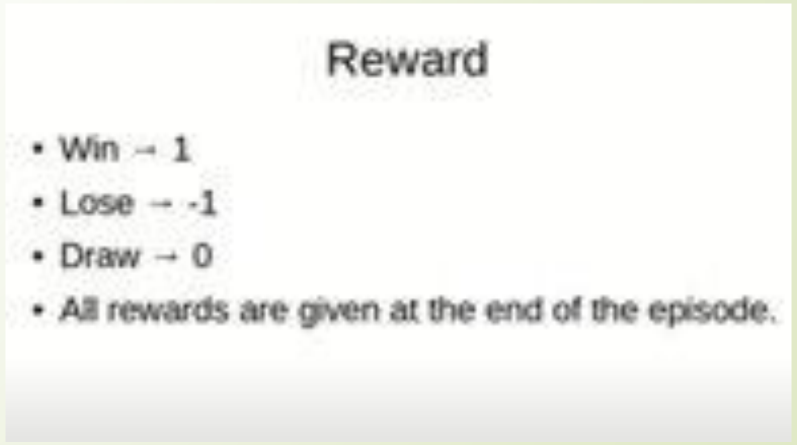
State Representation

- Player's Current Total
- Dealer's showing card



Action

- Hit (Request for additional card)
- Stick (End your turn)



Reward

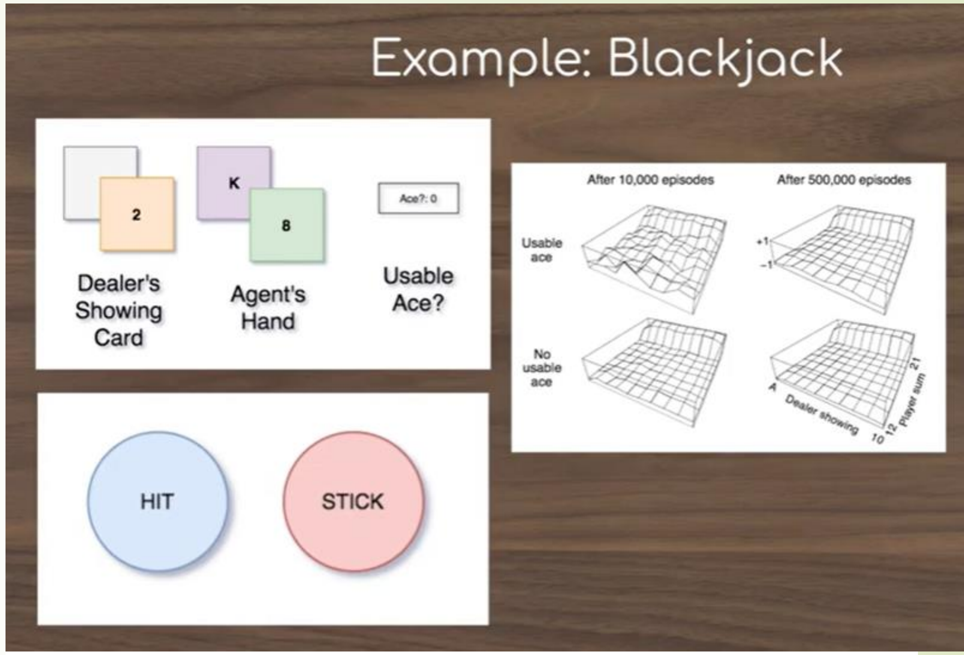
- Win → 1
- Lose → -1
- Draw → 0
- All rewards are given at the end of the episode.

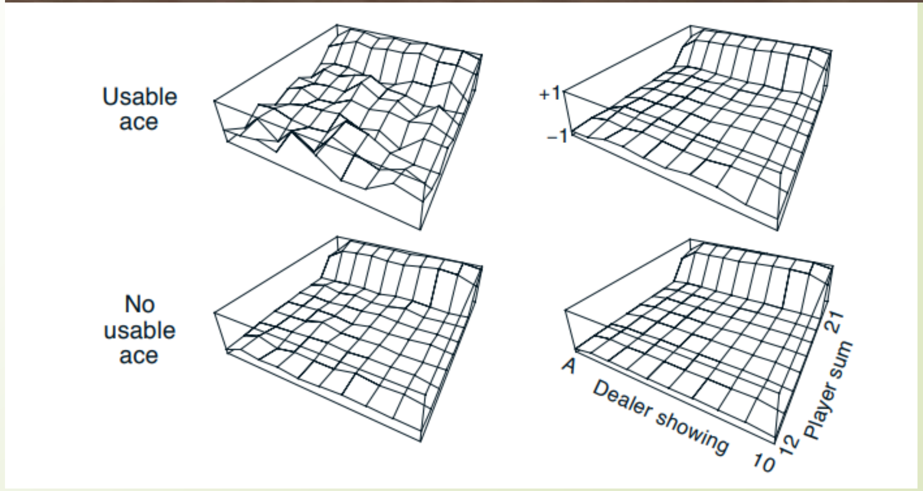
Blackjack Problem

- We assume that cards are dealt from an infinite deck so that there is no advantage to keeping track of the cards already dealt.
- If the player holds an ace that he could count as 11 without going bust, then the ace is said to be usable.
- In this case it is always counted as 11 because counting it as 1 would make the sum 11 or less, in which case there is no decision to be made because, obviously, the player should always hit.
- Thus, the player makes decisions on the basis of three variables: his current sum (12–21), the dealer's one showing card (ace–10), and whether or not he holds a usable ace.

State value function computation for Blackjack Problem

- Consider the policy that sticks if the player's sum is 20 or 21, and otherwise hits.
- To find the state-value function for this policy by a Monte Carlo approach, one simulates many blackjack games using the policy and averages the returns following each state.
- Note that in this task the same state never recurs within one episode, so there is no difference between first-visit and every-visit MC methods.
- In this way, we obtained the estimates of the state value function as shown in figure.



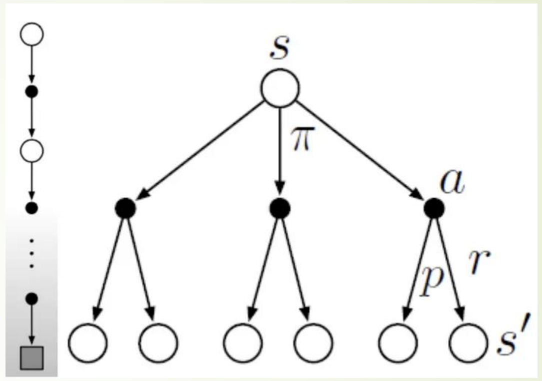


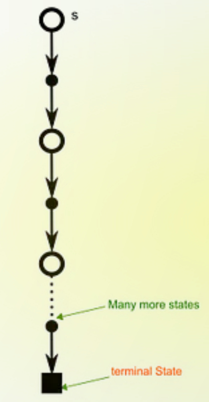
DP Vs Monte Carlo Methods

- Although we have complete knowledge of the environment in this task, it would not be easy to apply DP methods to compute the value function.
- DP methods require the distribution of next events—in particular, they require the quantities $p(s', r|s, a)$—and it is not easy to determine these for blackjack.
- For example, suppose the player's sum is 14 and he chooses to stick. What is his expected reward as a function of the dealer's showing card?
- All of these expected rewards and transition probabilities must be computed before DP can be applied, and such computations are often complex and error-prone.
- In contrast, generating the sample games required by Monte Carlo methods is easy.
- This is the case surprisingly often; the ability of Monte Carlo methods to work with sample episodes alone can be a significant advantage even when one has complete knowledge of the environment's dynamics.

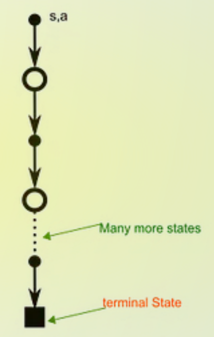
Backup Diagram for MC

- The general idea of a backup diagram is to show at the top the root node to be updated and to show below all the transitions and leaf nodes whose rewards and estimated values contribute to the update.
- For Monte Carlo estimation of v_π, the root is a state node, and below it is the entire trajectory of transitions along a particular single episode, ending at the terminal state, as in Figure.
- DP diagram shows all possible transitions, and the Monte Carlo diagram shows only those sampled on the one episode.
- DP diagram includes only one-step transitions, the Monte Carlo diagram goes all the way to the end of the episode.
- These differences in the diagrams accurately reflect the fundamental differences between the algorithms.





Monte Carlo State Value $v_\pi(s)$



Monte Carlo Action Value $q_\pi(s,a)$

DP Vs Monte Carlo

- Dynamic programming requires a complete knowledge of the environment or all possible transitions, whereas Monte Carlo methods work on a sampled state-action trajectory on one episode.
- DP includes only one-step transition, whereas MC goes all the way to the end of the episode to the terminal node.
- One important fact about the MC method is that the estimates for each state are independent, which means the estimate for one state does not build upon the estimate of any other state, as in the case of DP.

Advantages of MC

- The computational expense of estimating the value of a single state is independent of the number of states. This can make Monte Carlo methods particularly attractive when one requires the value of only one or a subset of states.
- One can generate many sample episodes starting from the states of interest, averaging returns from only these states ignoring all others.

Monte Carlo Estimation of Action Values

- The fundamental problem in action value estimation is many state action pairs never be visited.
- We made two unlikely assumptions in order to easily obtain this guarantee of convergence for the Monte Carlo method.

1. One was that the episodes have **exploring starts**
2. The other was that policy evaluation could be done with an **infinite number of episodes.**

Monte Carlo Estimation of Action Values

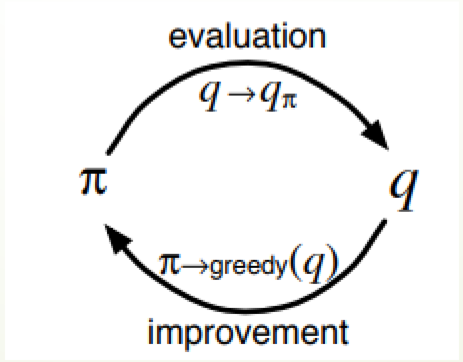
- If π is a deterministic policy, then in following π one will observe returns only for one of the actions from each state.
- With no returns to average, the Monte Carlo estimates of the other actions will not improve with experience.
- This is a serious problem because the purpose of learning action values is to help in choosing among the actions available in each state.
- To compare alternatives, we need to estimate the value of all the actions from each state, not just the one we currently favor.
- This is the general problem of maintaining exploration

Monte Carlo Estimation of Action Values

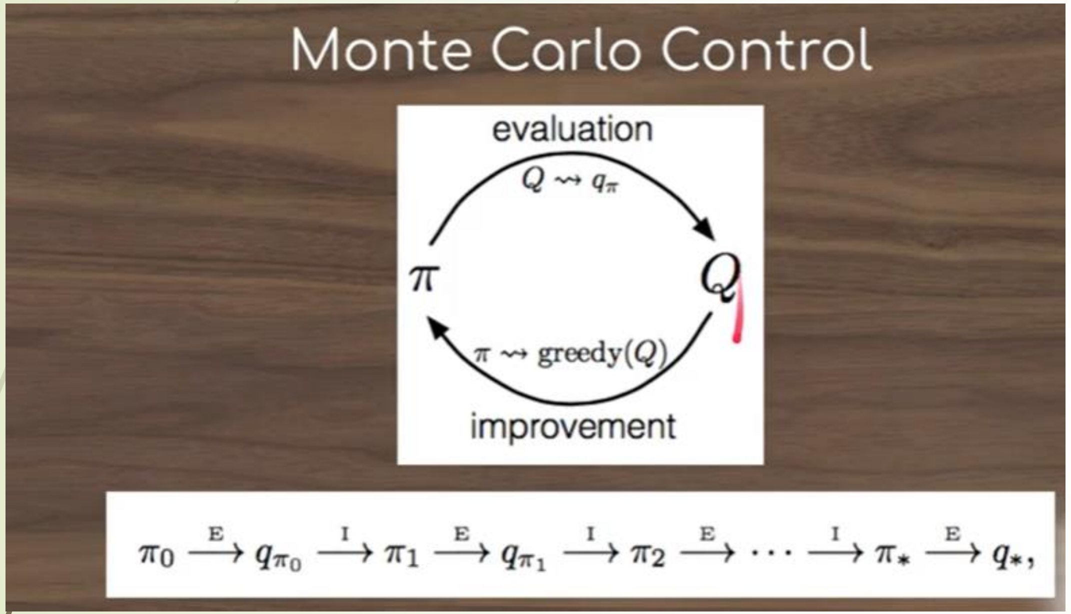
- For policy evaluation to work we need to make sure that we visit all actions in every state.
- One way to go about this, is to specify that every episode will start in some state–action pair, and that every pair has a probability > 0 to be selected as the start pair.
- This guarantees that all state–action pairs will be visited an infinite number of times in the limit of an infinite number of episodes.
- We call this the assumption of exploring starts.

Monte Carlo Control

- The overall idea of MC control is to proceed according to the same pattern as in the DP, that is, according to the idea of generalized policy iteration (GPI).
- In GPI one maintains both an approximate policy and an approximate value function.
- The value function is repeatedly altered to more closely approximate the value function for the current policy, and the policy is repeatedly improved with respect to the current value function:



Monte Carlo Control


Monte Carlo Control
evaluation
$Q \rightsquigarrow q_\pi$
π
Q
$\pi \rightsquigarrow \text{greedy}(Q)$
improvement

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} q_*,$$

- let us consider a Monte Carlo version of classical policy iteration.
- In this method, we perform alternating complete steps of policy evaluation and policy improvement, beginning with an arbitrary policy π0 and ending with the optimal policy and optimal action-value function as shown in figure.

Policy Improvement in MC

- Policy improvement is done by making the policy greedy with respect to the current value function.
- In this case we have an action-value function, and therefore no model is needed to construct the greedy policy.
- For any action value function q, the corresponding greedy policy is the one that, for each s ∈ S, deterministically chooses an action with maximal action-value:

$$\pi(s) = \arg\max_a q(s, a).$$

Policy Improvement in MC

- Policy improvement then can be done by constructing each π_{k+1} as the greedy policy with respect to $q_{\pi k}$.
- The policy improvement theorem then applies to π_k and π_{k+1} for all s ∈ S,

$$\begin{aligned}
\pi'(s) &= \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}$$

$$\begin{aligned}
q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\
&= \max_a q_{\pi_k}(s,a) \\
&\geq q_{\pi_k}(s, \pi_k(s)) \\
&= v_{\pi_k}(s).
\end{aligned}$$

Policy Improvement in MC

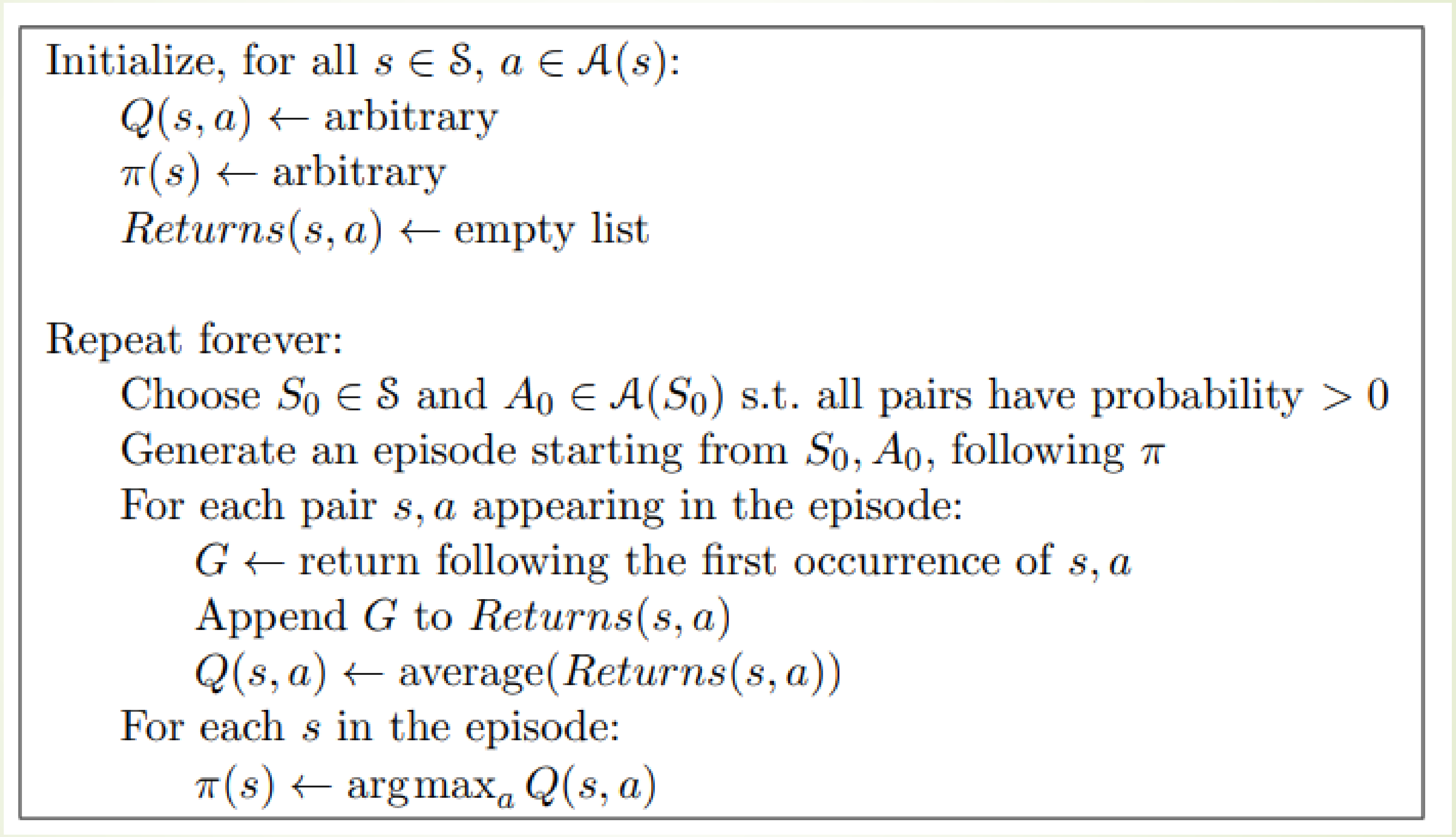
$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\ &= \max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &= v_{\pi_k}(s). \end{aligned}$$

- The theorem assures us that each πk+1 is uniformly better than πk, or just as good as πk, in which case they are both optimal policies.
- This in turn assures us that the overall process converges to the optimal policy and optimal value function.
- In this way Monte Carlo methods can be used to find optimal policies given only sample episodes and no other knowledge of the environment's dynamics.

Monte Carlo with Exploring Starts

- For Monte Carlo policy evaluation it is natural to alternate between evaluation and improvement on an episode-by-episode basis.
- After each episode, the observed returns are used for policy evaluation, and then the policy is improved at all the states visited in the episode.
- A complete simple algorithm along these lines is called as Monte Carlo ES, for Monte Carlo with Exploring Starts

Monte Carlo with Exploring Starts



```
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \leftarrow$ arbitrary
    $\pi(s) \leftarrow$ arbitrary
    $Returns(s,a) \leftarrow$ empty list

Repeat forever:
    Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability $> 0$
    Generate an episode starting from $S_0, A_0$, following $\pi$
    For each pair $s, a$ appearing in the episode:
        $G \leftarrow$ return following the first occurrence of $s, a$
        Append $G$ to $Returns(s,a)$
        $Q(s,a) \leftarrow$ average$(Returns(s,a))$
    For each $s$ in the episode:
        $\pi(s) \leftarrow \arg\max_a Q(s,a)$
```

A Monte Carlo control algorithm assuming exploring starts and that episodes always terminate for all policies.

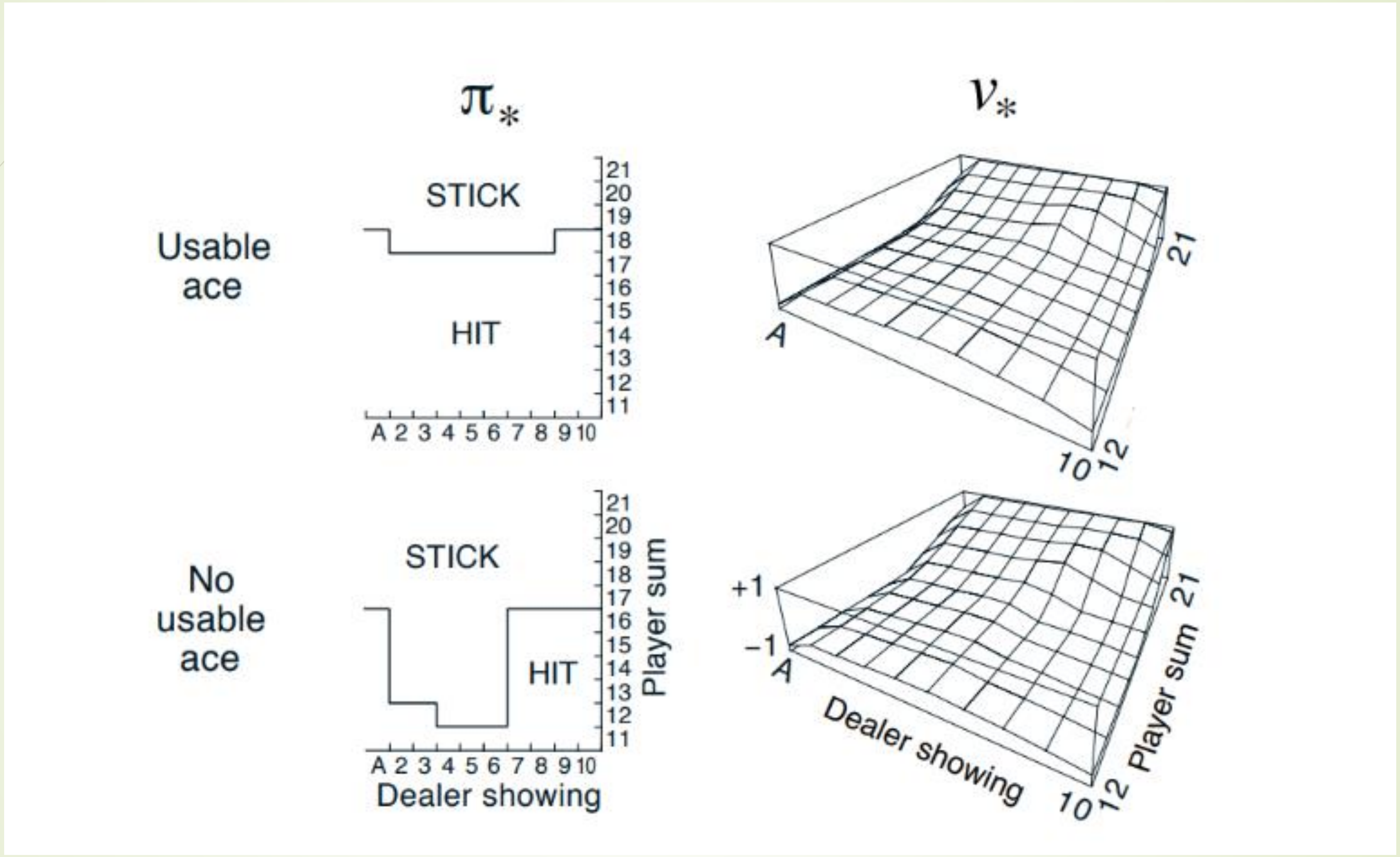
Monte Carlo with Exploring Starts

- In Monte Carlo ES, all the returns for each state–action pair are accumulated and averaged, irrespective of what policy was in force when they were observed.
- It is easy to see that Monte Carlo ES cannot converge to any suboptimal policy. If it did, then the value function would eventually converge to the value function for that policy, and that in turn would cause the policy to change.
- Stability is achieved only when both the policy and the value function are optimal.
- Convergence to this optimal fixed point seems inevitable as the changes to the action-value function decrease over time

Solution of Blackjack Problem

- It is straightforward to apply Monte Carlo ES to blackjack.
- The episodes are all simulated games, so it is easy to arrange for exploring starts that include all possibilities.
- In this case one simply picks the dealer's cards, the player's sum, and whether or not the player has a usable ace, all at random with equal probability.
- As the initial policy we use the policy evaluated in the previous blackjack example, that which sticks only on 20 or 21.
- The initial action-value function can be zero for all state–action pairs.

Solution of Blackjack Problem



The optimal policy and state-value function for blackjack, found by Monte Carlo ES. The state-value function shown was computed from the action-value function found by Monte Carlo ES.

Monte Carlo Control without Exploring starts

How can we avoid the unlikely assumption of exploring starts?

- The only general way to ensure that all actions are selected infinitely often is for the agent to continue to select them.
- There are two approaches to ensuring this, resulting in what we call on-policy methods and off-policy methods.
- **On-policy** methods attempt to evaluate or improve the policy that is used to make decisions, whereas **off-policy** methods evaluate or improve a policy different from that used to generate the data.
- The Monte Carlo ES method is an example of an on-policy method.
- **On-policy** methods: we try to evaluate or improve the policy that we have.
- **Off-policy** methods: we have 2 policies, and we try to evaluate or improve one of them, and use the other for directions.

On- Policy Monte Carlo Control

- In on-policy MC control methods the policy is generally ***soft*** meaning that the probability for taking action ***a*** in state **s** is >0 for all ***a*** in ***A(s)*** and ***s*** in ***S.***
- But over time the policy shifts gradually closer and closer to a deterministic policy.
- One way to define such a policy is using ***epsilon-greedy policies,*** which means that most of the time the action with the highest estimated value is selected, and with a probability of *epsilon* a random action is selected.
- By behaving this way a constant exploration is guaranteed.

On- Policy Monte Carlo Control

- That is, the policy is designed in a way that all nongreedy action are given the minimal probability of selection which is:

$$\frac{\epsilon}{|A(s)|}$$

- And the remaining bulk of the probability is given to the greedy action:

$$1 - \epsilon + \frac{\epsilon}{|A(s)|}$$

On- Policy Monte Carlo Control

- The epsilon-greedy policies are examples of epsilon-soft policies and defined as policies for which

$$\pi(a|s) \geq \frac{\epsilon}{|A(s)|}$$

- for all states and actions, for some *epsilon* > 0. Among epsilon-soft policies, epsilon-greedy policies are in some sense those that are closest to greedy.

On- Policy Monte Carlo Control

- The overall idea of on-policy Monte Carlo control is same as that of GPI.
- As in Monte Carlo ES, we use first-visit MC methods to estimate the action-value function for the current policy.
- Without the assumption of exploring starts, however, we cannot simply improve the policy by making it greedy with respect to the current value function, because that would prevent further exploration of nongreedy actions.
- Fortunately, GPI does not require that the policy be taken all the way to a greedy policy, only that it be moved toward a greedy policy.
- In on-policy method we will move it only to an ε-greedy policy.
- For any ε-soft policy, π, any ε-greedy policy with respect to q_π is guaranteed to be better than or equal to π.

On- Policy Monte Carlo Control

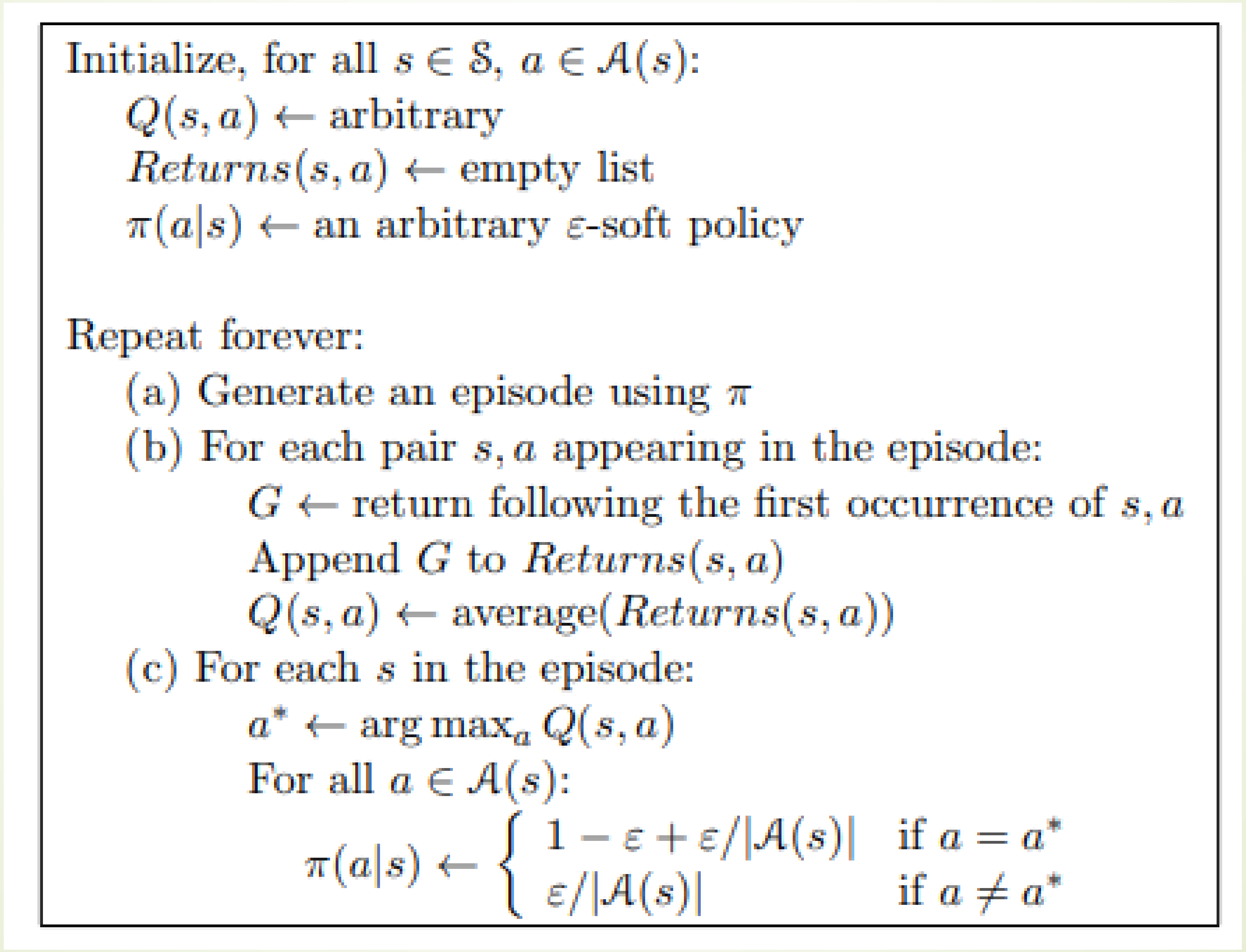
- Let v∗ and q∗ denote the optimal value functions for the new environment. Then a policy π is optimal among ε-soft policies if and only if vπ = v∗.
- From the definition of v∗ we know that it is the unique solution to

$$\begin{aligned}\widetilde{v}_*(s) &= (1-\varepsilon)\max_a \widetilde{q}_*(s,a) + \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a \widetilde{q}_*(s,a)\\ &= (1-\varepsilon)\max_a \sum_{s',r} p(s',r|s,a)\Big[r+\gamma\widetilde{v}_*(s')\Big]\\ &\quad + \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a\sum_{s',r} p(s',r|s,a)\Big[r+\gamma\widetilde{v}_*(s')\Big].\end{aligned}$$

When equality holds and the ε-soft policy π is no longer improved, then we get:

$$\begin{aligned}v_\pi(s) &= (1-\varepsilon)\max_a q_\pi(s,a) + \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a)\\ &= (1-\varepsilon)\max_a \sum_{s',r} p(s',r|s,a)\Big[r+\gamma v_\pi(s')\Big]\\ &\quad + \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a\sum_{s',r} p(s',r|s,a)\Big[r+\gamma v_\pi(s')\Big].\end{aligned}$$

On- Policy First Visit Monte Carlo Control (for ε-soft policies)



```
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \leftarrow$ arbitrary
    $Returns(s,a) \leftarrow$ empty list
    $\pi(a|s) \leftarrow$ an arbitrary $\varepsilon$-soft policy

Repeat forever:
    (a) Generate an episode using $\pi$
    (b) For each pair $s, a$ appearing in the episode:
            $G \leftarrow$ return following the first occurrence of $s, a$
            Append $G$ to $Returns(s,a)$
            $Q(s,a) \leftarrow$ average$(Returns(s,a))$
    (c) For each $s$ in the episode:
            $a^* \leftarrow \arg\max_a Q(s,a)$
            For all $a \in \mathcal{A}(s)$:
                $\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$
```

Off-Policy Prediction via Importance Sampling

- So far we have considered methods for estimating the value functions for a policy given an infinite supply of episodes generated using that policy.
- Suppose now that all we have are episodes generated from a different policy.
- That is, suppose we wish to estimate v_π or q_π, but all we have are episodes following another policy μ, where μ != π.
- We call π the target policy because learning its value function is the target of the learning process, and we call μ the behavior policy because it is the policy controlling the agent and generating behavior.
- The overall problem is called off-policy learning because it is learning about a policy given only experience "off" (not following) that policy.

Off-Policy Prediction via Importance Sampling

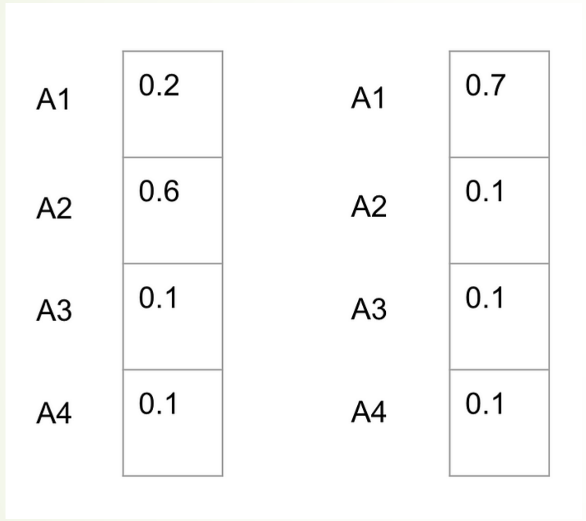
- In order to use episodes from μ to estimate values for π, we must require that every action taken under π is also taken, at least occasionally, under μ.
- That is, we require that π(a|s) > 0 implies μ(a|s) > 0. This is called the assumption of coverage.
- It follows from coverage that μ must be stochastic in states where it is not identical to π.
- The target policy π, on the other hand, may be deterministic, and, in fact, this is a case of particular interest.
- Typically, the target policy is the deterministic greedy policy with respect to the current action-value function estimate.
- This policy we hope becomes a deterministic optimal policy while the behavior policy remains stochastic and more exploratory, for example, an ε-greedy policy.

Importance Sampling

- Importance sampling is a general technique for estimating expected values under one distribution given samples from another.
- We apply this technique to off-policy learning by weighting returns according to the relative probability of their trajectories occurring under the target and behavior policies, called the importance-sampling ratio.

Importance Sampling

- Consider the 2 vectors below:



A1	0.2	A1	0.7
A2	0.6	A2	0.1
A3	0.1	A3	0.1
A4	0.1	A4	0.1

π (left) and b (right)

Looking at the target policy π, we see that we have 20% chance of taking action A1, where the behavior policy b have 70% chance of taking A1. These kind of different probabilities for actions is what we need to weight the actions differently in π (left) and b (right)

Importance Sampling

- So let's see how importance sampling is used to align the probabilities of 2 policies, so that we can update π even though we're behaving using b.

$$\rho_t^T = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} \mu(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$$

- Looking at the formula for the importance-sampling ratio, it is the product of taking a certain action given a certain state from the target policy compared to the behavior policy.

Importance Sampling

- Given a starting state St , the probability of the subsequent state–action trajectory, At , St+1, At+1, . . . , ST , occurring under any policy π is

$$\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),$$

- Where p is the state-transition probability function. Thus, the relative probability of the trajectory under the target and behavior policies (the importance-sampling ratio) is

$$\rho_t^T = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} \mu(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}.$$

- Note that although the trajectory probabilities depend on the MDP's transition probabilities, which are generally unknown, all the transition probabilities cancel and drop out.
- The importance sampling ratio ends up depending only on the two policies and not at all on the MDP.

Off-Policy Prediction via Importance Sampling

- A Monte Carto algorithm uses a batch of observed episodes following policy μ to estimate $V\pi(s)$.
- It is convenient here to number time steps in a way that increases across episode boundaries.
- That is, if the first episode of the batch ends in a terminal state at time 100, then the next episode begins at time $t = 101$.
- This enables us to use time-step numbers to refer to particular steps in particular episodes.
- In particular, we can define the set of all time steps in which state s is visited, denoted $T(s)$.
- This is for an every-visit method; for a first-visit method, $T(s)$ would only include time steps that were first visits to s within their episode.

Off-Policy Prediction via Importance Sampling

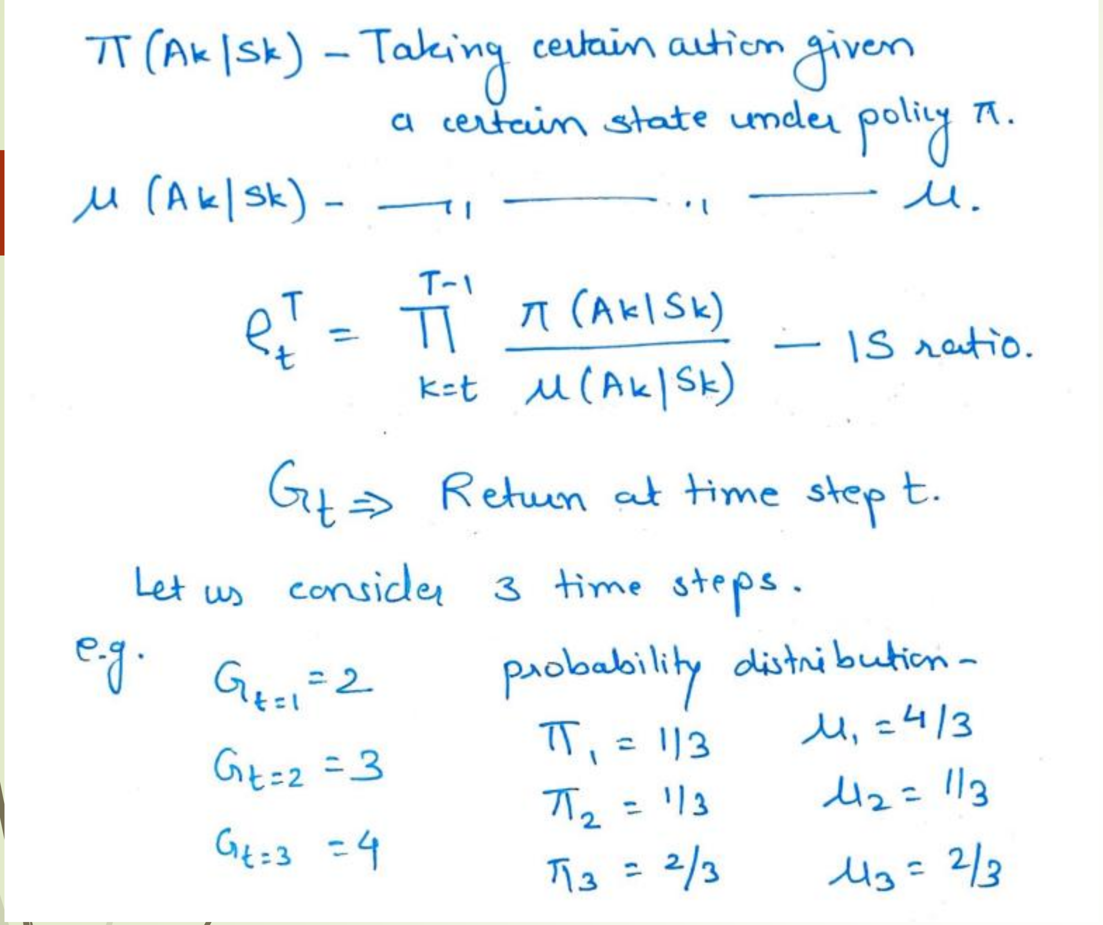
- Generally speaking, there are two ways to calculate the state-value with importance sampling.
- **Ordinary importance sampling,** where the ration is multiplied by the Return and averaged by *T(s) (*the set of all time steps in which state s is visited):

$$V(s) = \frac{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)} G_t}{|\mathcal{T}(s)|}.$$

- Or **weighted importance sampling**, which is a weighted average:

$$V(s) = \frac{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)} G_t}{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)}},$$

let T(t) denote the first time of termination following time t, and Gt denote the return after t up through T(t). Then {Gt}t∈T(s) are the returns that pertain to state s, and {ρ T(t) t }t∈T(s) are the corresponding importance-sampling ratios



$\pi(A_k|S_k)$ – Taking certain action given a certain state under policy π.

$\mu(A_k|S_k)$ – —"— —"— — μ.

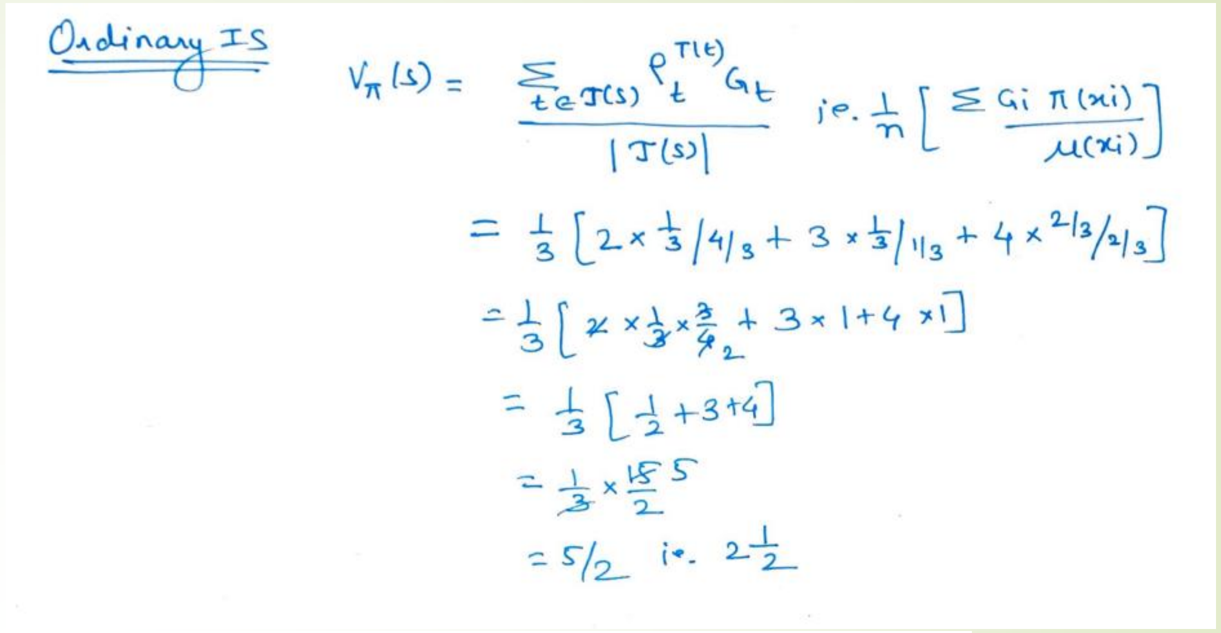
$$\rho_t^T = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)} \text{ – IS ratio.}$$

$G_t \Rightarrow$ Return at time step t.

Let us consider 3 time steps.

e.g.

	probability distribution –	
$G_{t=1} = 2$	$\pi_1 = 1/3$	$\mu_1 = 4/3$
$G_{t=2} = 3$	$\pi_2 = 1/3$	$\mu_2 = 1/3$
$G_{t=3} = 4$	$\pi_3 = 2/3$	$\mu_3 = 2/3$



Ordinary IS

$$V_\pi(s) = \frac{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)} G_t}{|\mathcal{T}(s)|} \quad \text{i.e. } \frac{1}{n}\left[\sum \frac{G_i\, \pi(x_i)}{\mu(x_i)}\right]$$

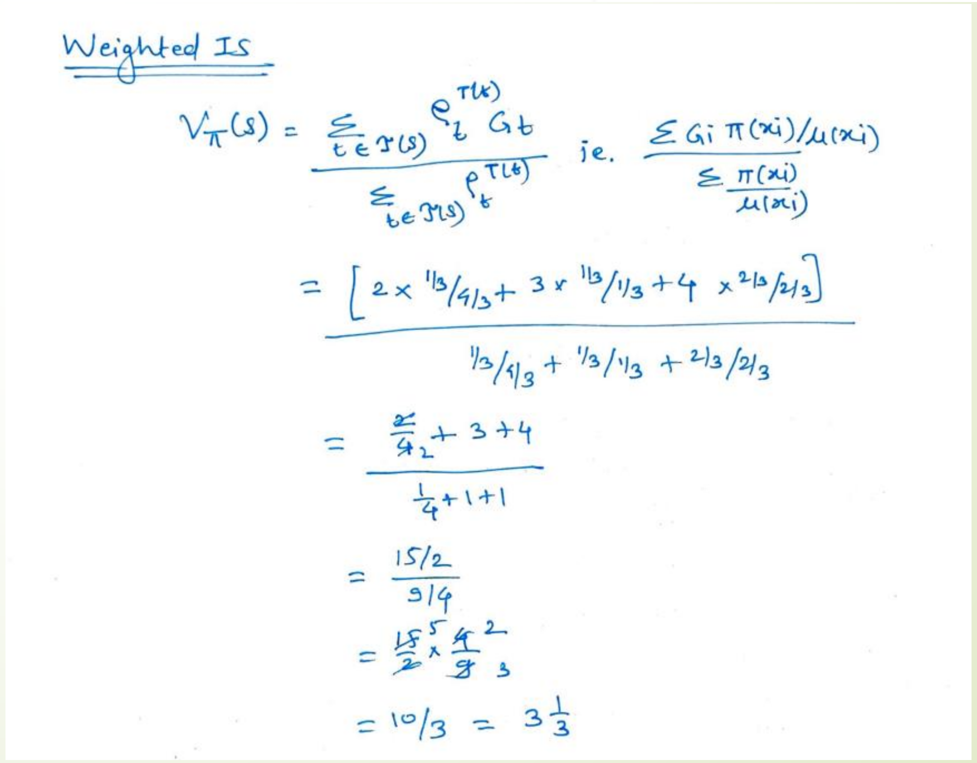
$$= \frac{1}{3}\left[2 \times \tfrac{1}{3}/\tfrac{4}{3} + 3 \times \tfrac{1}{3}/\tfrac{1}{3} + 4 \times \tfrac{2}{3}/\tfrac{2}{3}\right]$$

$$= \frac{1}{3}\left[\cancel{2} \times \frac{1}{\cancel{3}} \times \frac{\cancel{3}}{\cancel{4}_2} + 3 \times 1 + 4 \times 1\right]$$

$$= \frac{1}{3}\left[\frac{1}{2} + 3 + 4\right]$$

$$= \frac{1}{\cancel{3}} \times \frac{\cancel{15}\,5}{2}$$

$$= 5/2 \text{ i.e. } 2\tfrac{1}{2}$$



Weighted IS

$$V_\pi(s) = \frac{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)} G_t}{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)}} \quad \text{i.e. } \frac{\sum G_i\, \pi(x_i)/\mu(x_i)}{\sum \frac{\pi(x_i)}{\mu(x_i)}}$$

$$= \frac{\left[2 \times \tfrac{1}{3}/\tfrac{4}{3} + 3 \times \tfrac{1}{3}/\tfrac{1}{3} + 4 \times \tfrac{2}{3}/\tfrac{2}{3}\right]}{\tfrac{1}{3}/\tfrac{4}{3} + \tfrac{1}{3}/\tfrac{1}{3} + \tfrac{2}{3}/\tfrac{2}{3}}$$

$$= \frac{\frac{\cancel{2}}{\cancel{4}_2} + 3 + 4}{\frac{1}{4} + 1 + 1}$$

$$= \frac{15/2}{9/4}$$

$$= \frac{\cancel{15}^{5}}{\cancel{2}} \times \frac{\cancel{4}^{2}}{\cancel{9}_{3}}$$

$$= 10/3 = 3\tfrac{1}{3}$$

Importance Sampling

- Formally, the difference between the two kinds of importance sampling for the first-visit methods is expressed in their biases and variances.
- **Ordinary importance** sampling is unbiased whereas **weighted importance** sampling is biased (but the bias converges asymptotically to zero).
- On the other hand, the variance of **ordinary importance sampling** is in general unbounded because the variance of the ratios can be unbounded, whereas in the **weighted estimator** the largest weight on any single return is one.
- In practice, the weighted estimator usually has a dramatically lower variance and is strongly preferred.

Consider single return as G = 3.

<u>WIS</u>

$$V_\pi(s) = \cancel{\tfrac{1}{3}}\frac{\left[3 \times {}^{1/3}/_{4/3} + 3 \times {}^{1/3}/_{1/3} + 3 \times {}^{2/3}/_{2/3}\right]}{{}^{1/3}/_{4/3} + {}^{1/3}/_{1/3} + {}^{2/3}/_{2/3}}$$

$$= \frac{\frac{3}{4} + 3 + 3}{\frac{1}{4} + 1 + 1}$$

$$= \frac{3\,\cancel{(\frac{1}{4}+1+1)}}{\cancel{(\frac{1}{4}+1+1)}}$$

$$= \underline{\underline{3}} \Rightarrow \underline{\underline{G}}$$

Consider single return G = 4.

& say ratio ⇒ 10

then <u>OIS</u> $V_\pi(s) = \dfrac{4\times10 + 4\times10 + 4\times10}{3}$

$$= \frac{\cancel{3}\times 40}{\cancel{3}}$$

$= \underline{10 \text{ times } G.}$

Incremental Implementation

- Monte Carlo prediction methods can be implemented incrementally, on an episode-by-episode basis using techniques we learned for bandit problems where we averaged rewards, but in Monte Carlo methods we average returns.
- In all other respects exactly the same methods can be used for on-policy Monte Carlo methods.
- For off-policy Monte Carlo methods, we need to separately consider those that use ordinary importance sampling and those that use weighted importance sampling.

Incremental Implementation

- In ordinary importance sampling, the returns are scaled by the importance sampling ratio $\rho_t^{T(t)}$, then simply averaged.
- For these methods we can again use the incremental methods we learned earlier but using the scaled returns in place of the rewards
- This leaves the case of off-policy methods using weighted importance sampling.
- Here we have to form a weighted average of the returns, and a slightly different incremental algorithm is required.

Incremental Implementation

- Suppose we have a sequence of returns G1, G2, . . . , Gn−1, all starting in the same state and each with a corresponding random weight Wi (e.g., Wi = $\rho_t^{T(t)}$).

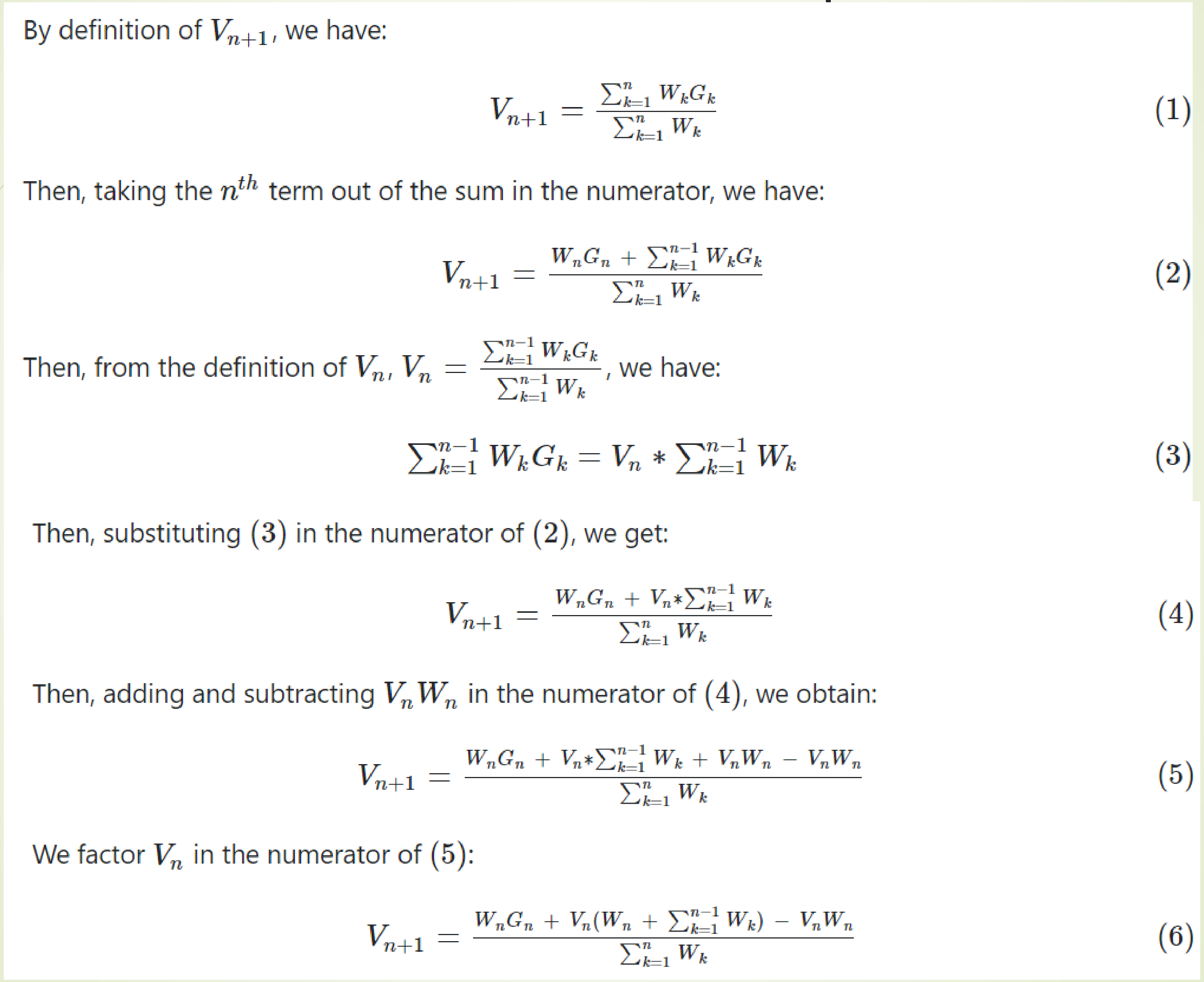
- We wish to form the estimate $$V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \qquad n \geq 2,$$

- and keep it up-to-date as we obtain a single additional return Gn. In addition to keeping track of Vn, we must maintain for each state the cumulative sum Cn of the weights given to the first n returns. The update rule for Vn is

$$V_{n+1} = V_n + \frac{W_n}{C_n}\Big[G_n - V_n\Big], \qquad n \geq 1,$$ and $$C_{n+1} = C_n + W_{n+1},$$

where C0 = 0 (and V1 is arbitrary and thus need not be specified).

Derivation of incremental update rule



By definition of V_{n+1}, we have:

$$V_{n+1} = \frac{\sum_{k=1}^{n} W_k G_k}{\sum_{k=1}^{n} W_k} \tag{1}$$

Then, taking the n^{th} term out of the sum in the numerator, we have:

$$V_{n+1} = \frac{W_n G_n + \sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n} W_k} \tag{2}$$

Then, from the definition of V_n, $V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}$, we have:

$$\sum_{k=1}^{n-1} W_k G_k = V_n * \sum_{k=1}^{n-1} W_k \tag{3}$$

Then, substituting (3) in the numerator of (2), we get:

$$V_{n+1} = \frac{W_n G_n + V_n * \sum_{k=1}^{n-1} W_k}{\sum_{k=1}^{n} W_k} \tag{4}$$

Then, adding and subtracting $V_n W_n$ in the numerator of (4), we obtain:

$$V_{n+1} = \frac{W_n G_n + V_n * \sum_{k=1}^{n-1} W_k + V_n W_n - V_n W_n}{\sum_{k=1}^{n} W_k} \tag{5}$$

We factor V_n in the numerator of (5):

$$V_{n+1} = \frac{W_n G_n + V_n (W_n + \sum_{k=1}^{n-1} W_k) - V_n W_n}{\sum_{k=1}^{n} W_k} \tag{6}$$

Derivation of incremental update rule

We simplify, taking into account that the denominator $\sum_{k=1}^{n} W_k = W_n + \sum_{k=1}^{n-1} W_k$, we get:

$$V_{n+1} = \frac{W_n G_n + V_n(W_n + \sum_{k=1}^{n-1} W_k) - V_n W_n}{\sum_{k=1}^{n} W_k} \longrightarrow V_{n+1} = V_n + \frac{W_n G_n - W_n V_n}{\sum_{k=1}^{n} W_k} \tag{7}$$

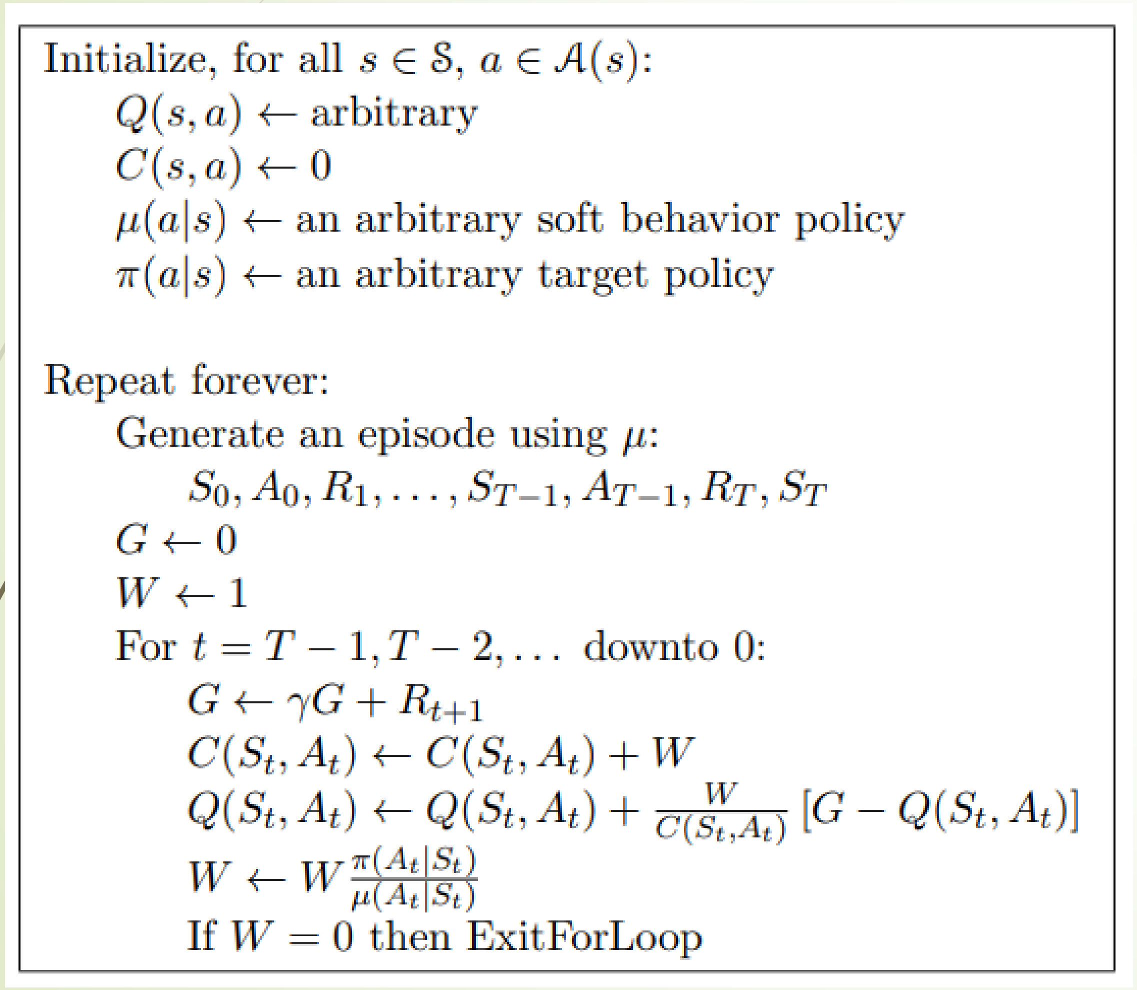
Further rearrangements of the terms give us:

$$V_{n+1} = V_n + \frac{W_n}{\sum_{k=1}^{n} W_k}[G_n - V_n] \tag{8}$$

Finally, by definition of C_n as the cumulative sum of the weights up to time n, we get the desired incremental update equation:

$$V_{n+1} = V_n + \frac{W_n}{C_n}[G_n - V_n] \tag{9}$$

An incremental every-visit MC policy-evaluation algorithm, using weighted importance sampling



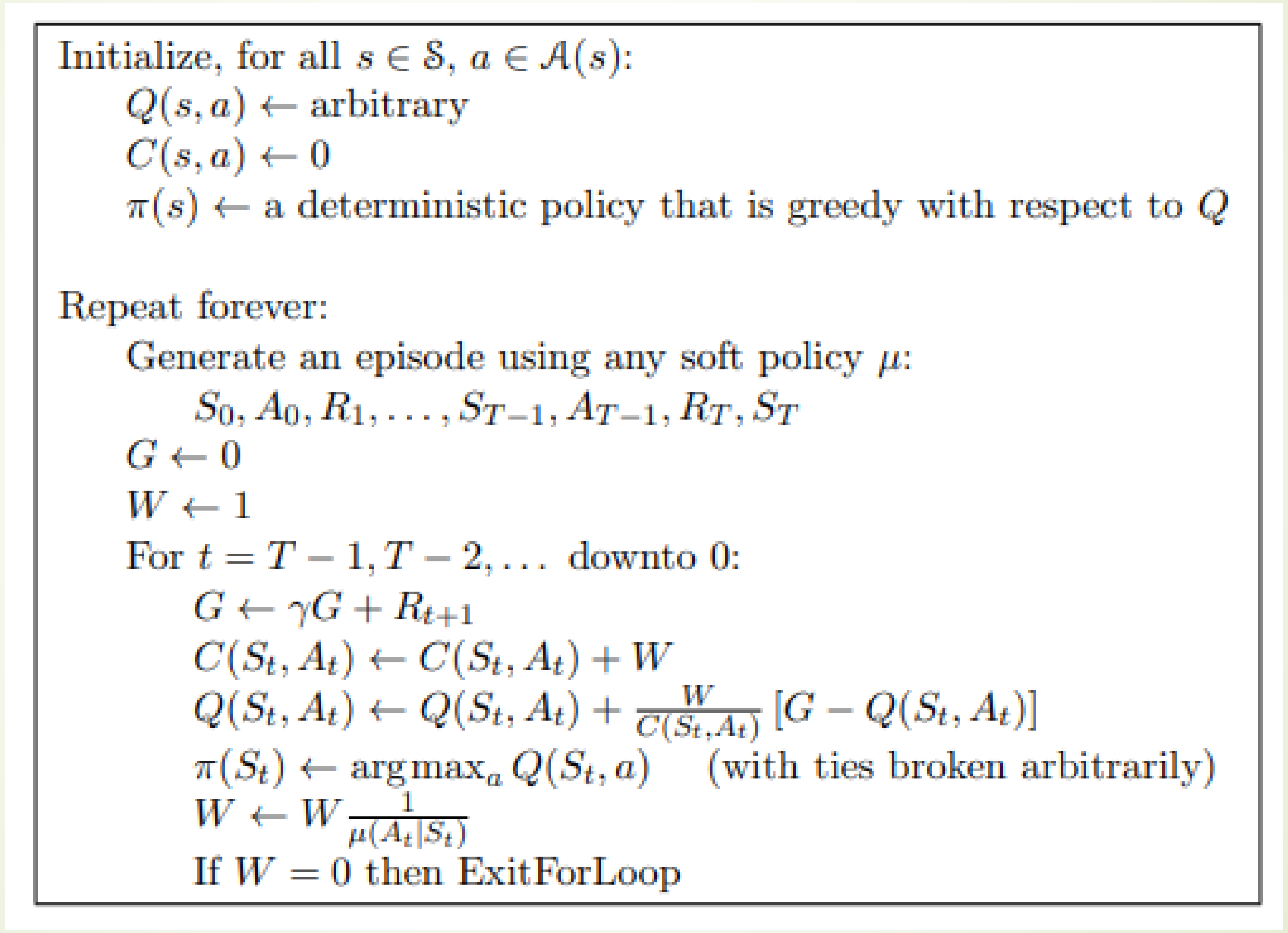
```
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \leftarrow$ arbitrary
    $C(s,a) \leftarrow 0$
    $\mu(a|s) \leftarrow$ an arbitrary soft behavior policy
    $\pi(a|s) \leftarrow$ an arbitrary target policy

Repeat forever:
    Generate an episode using $\mu$:
        $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T, S_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    For $t = T-1, T-2, \dots$ downto 0:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}$
        If $W = 0$ then ExitForLoop
```

- The algorithm is nominally for the off-policy case, using weighted importance sampling, but applies as well to the on-policy case just by choosing the target and behavior policies as the same.
- The approximation Q converges to qπ (for all encountered state–action pairs) even though all actions are selected according to a potentially different policy, μ. In the on-policy case (π = μ), W is always 1

Off-Policy Monte Carlo Control

- Off-policy Monte Carlo control methods follow the behavior policy while learning about and improving the target policy.
- These techniques requires that the behavior policy has a nonzero probability of selecting all actions that might be selected by the target policy (coverage).
- To explore all possibilities, we require that the behavior policy be soft (i.e., that it select all actions in all states with nonzero probability).

Off-Policy Monte Carlo Control



```
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \leftarrow$ arbitrary
    $C(s,a) \leftarrow 0$
    $\pi(s) \leftarrow$ a deterministic policy that is greedy with respect to $Q$

Repeat forever:
    Generate an episode using any soft policy $\mu$:
        $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    For $t = T-1, T-2, \ldots$ downto 0:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)}[G - Q(S_t, A_t)]$
        $\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$    (with ties broken arbitrarily)
        $W \leftarrow W \frac{1}{\mu(A_t|S_t)}$
        If $W = 0$ then ExitForLoop
```

Two Assumptions in MC

1. **Infinite number of episodes solution**

- One is to **hold firm to the idea** of approximating q_{π_k} in each policy evaluation. However, it is also likely to require far too many episodes to be useful in practice on any but the smallest problems.
- Another one is **similar to the idea of GPI**. On each evaluation step we move the value function toward q_{π_k}, but we **do not expect to actually get close except over many steps**. One extreme form of the idea is to alternatively apply policy improvement and policy evaluation.

Two Assumptions in MC

2. Exploring Starts Solution

Exploring starts is a good method but cannot be applied to every task, such as self-driving cars. There are two alternative solutions:

- On-policy control methods attempt to **evaluate or improve the policy that is used to make decisions**.
- Off-policy control methods evaluate or **improve a policy different from that used to generate the data**.

Advantages and Limitations of MC

- **Advantages** :

1. zero bias
2. Good convergence properties (even with function approximation)
3. Not very sensitive to initial value
4. Very simple to understand and use

- **Limitations** :

1. MC must wait until end of episode before return is known
2. MC has high variance
3. MC can only learn from complete sequences
4. MC only works for episodic (terminating) environments

References

- Chapter 5 from Sutton and Barto
- https://www.youtube.com/watch?v=PljDuynF-j0 (game theory)

- https://www.youtube.com/watch?v=3686U57JnLs (simulation)
- https://www.youtube.com/watch?v=r9l9fsdNYPY&t=223s

- https://medium.com/data-science-in-your-pocket/monte-carlo-for-reinforcement-learning-with-example-1754439dd628

- https://eecs.wsu.edu/~taylorm/2010_cs414/414_19.pdf