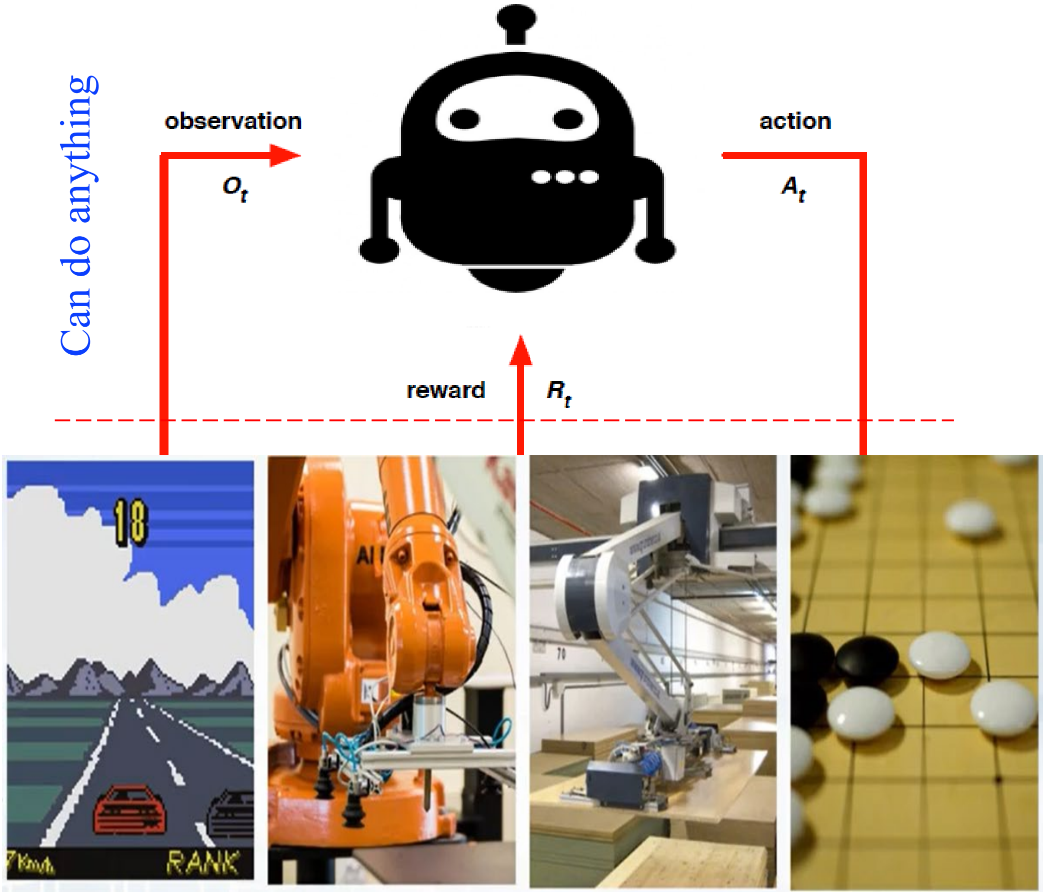
5. Monte-Carlo methods

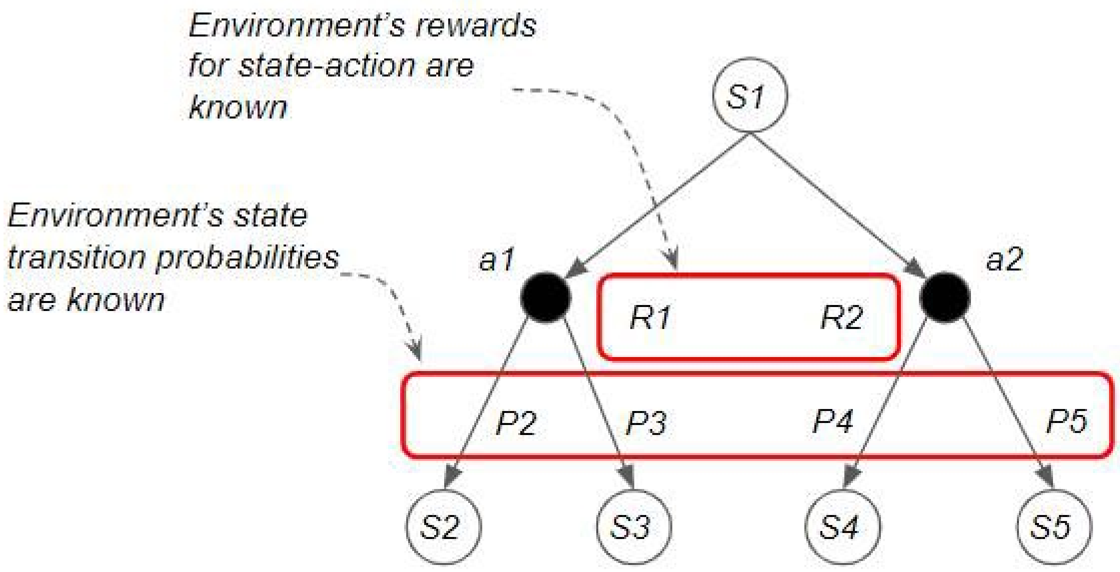
Model-based vs model-free learning

If we know the model of the environment $p(r, s'|s, a)$ for all r, s', s, a, we can derive the value function or the value-state function.

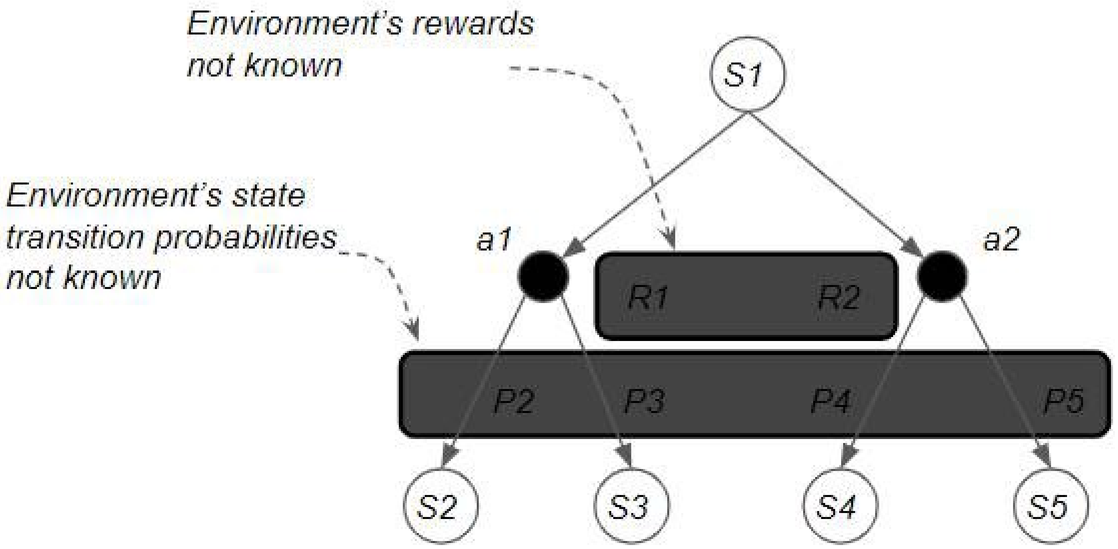
However, this cannot always be assumed, the environment may be a blackbox that we cannot model…

Can do anything





Model-based RL



Model-free RL

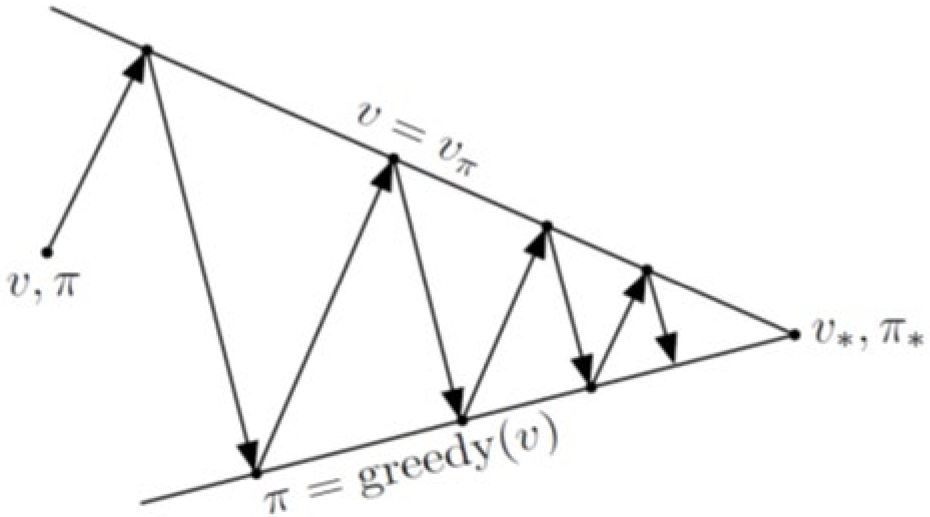
Figures by Ketan Doshi

Why Monte-Carlo methods?

- Is there any way you can apply dynamic programming here?
    - Learn $p(r, s'|s, a)$
    - Get rid of it, and use observations and estimates
- MC is a learning method for estimating value functions and discovering optimal policies.
- MC requires only experience (sample sequences of states, actions, and rewards from actual or simulated interaction with an environment).
- MC uses the simplest possible idea: value = mean return
- In this chapter we define MC methods only for episodic tasks:
    – Experience is divided into episodes
    – All episodes eventually terminate no matter what actions are selected
- So, MC can thus be incremental in an episode-by-episode sense, but not in a step-by-step (online) sense.

- We will adopt the concept of GPI seen in dynamic programming:
  - Policy Prediction
  - Policy Improvement

  to face the control problem and its solution.

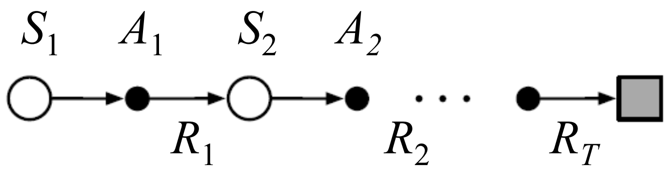


MC Prediction

- Learning the state-value function for a given policy.
- Reminder: the value of a state is the expected cumulative future discounted reward starting from that state

$$v_\pi(s) = E_\pi\{G_t | S_t = s\}$$

- Observe many episodes $S_1A_1R_1S_2A_2R_2$ … under a given policy π.
- Extract those that contain (s, a), obtain G for each episode and average them to get the expectation.



MC Prediction

Two approaches…

- In the **First-visit MC method,** $v_\pi(s)$ is estimated as the average of the returns following first visits to s.

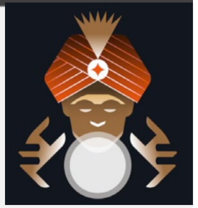
- In the **Every-visit MC method**, the returns are averaged following all visits to s.

First-Visit MC Policy Evaluation

The first time-step t that state s is visited in an episode:

– Increment counter $N(s) \leftarrow N(s) + 1$
– Increment total return $R(s) \leftarrow R(s) + G_t$
– Value is estimated by mean return $V(s) = S(s)/N(s)$
– By law of large numbers we can expect, $V(s) \rightarrow v(s)$ as $N(s) \rightarrow \infty$

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

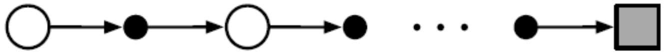


Initialize:
 $\pi \leftarrow$ policy to be evaluated
 $V \leftarrow$ an arbitrary state-value function
 $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:
 Generate an episode using π
 For each state s appearing in the episode:
  $G \leftarrow$ the return that follows the first occurrence of s
  Append G to $Returns(s)$
  $V(s) \leftarrow \text{average}(Returns(s))$

Advantages of MC methods

- The **backup diagram** shows only those sampled states on one episode. It goes all the way to the end of the episode.



- The estimated values for each state are independent, i.e. the estimate for one state does not build upon the estimate of any other state, unlike in DP.
- The **computational expense** of estimating the value of a single state is independent of the number of states.
- Ability **to learn from actual experience** (or from simulated experience).

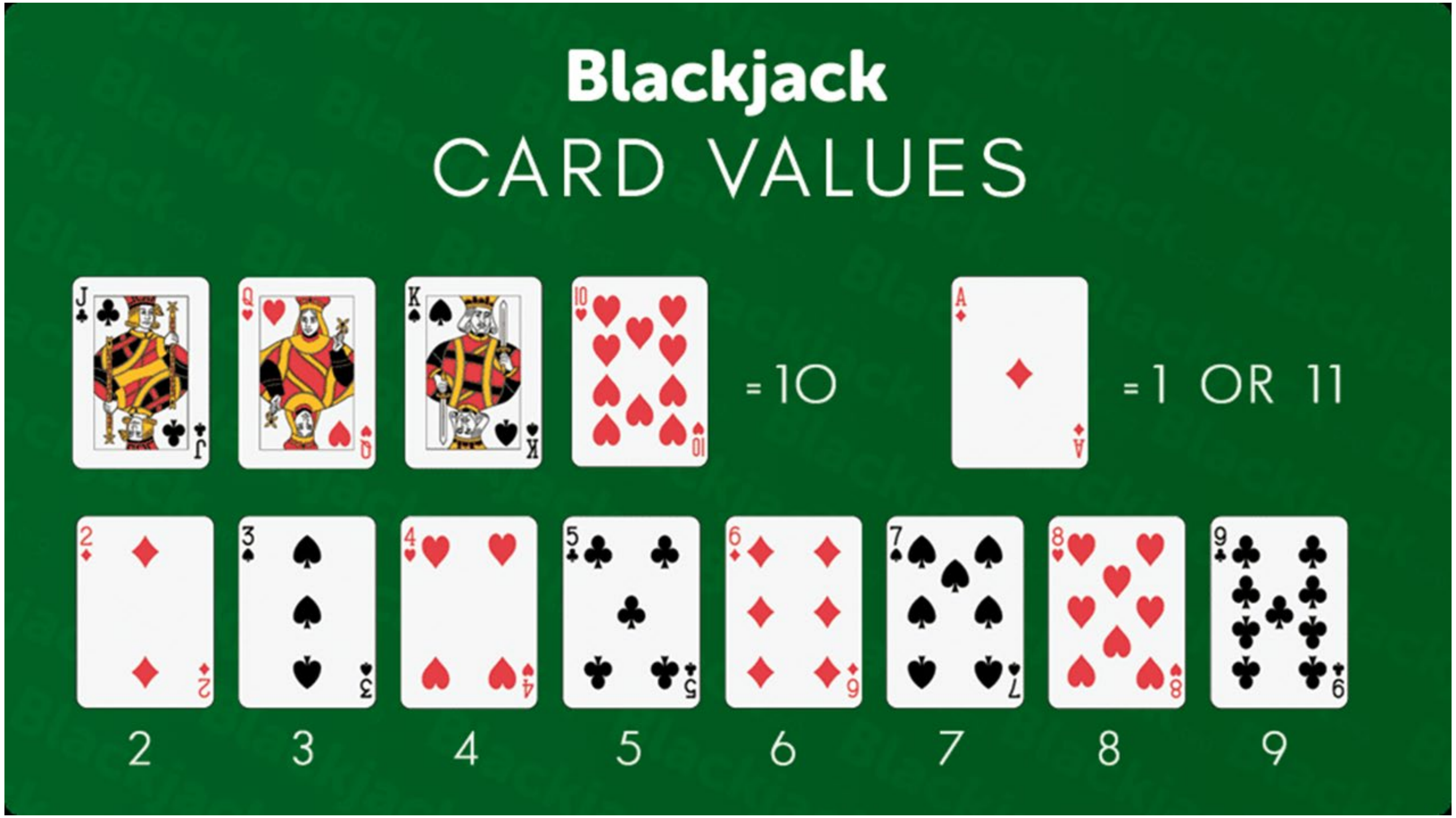
MC estimation of action-values

What is preferable, to learn $v_\pi(s)$ or $q_\pi(s,a)$? For policy optimization, remember $v_\pi(s)$ is useless without $p(s'|s,a)$…

- Primary goal for MC: **to estimate $q^*(s, a)$**
- The policy evaluation problem for action values is to estimate $q_\pi(s, a)$, under policy π :
  - The **every-visit MC** method estimates the value of a state-action pair as the average of the returns that have followed all the visits to it.
  - The **first-visit MC** method averages the returns following the first time in each episode that the state was visited and the action was selected.

Example 5.1: Blackjack (I)

- **Objective:** 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.



From Sutton & Barto, Reinforcement Learning: An Introduction, 1998

Example 5.1: Blackjack (II)

Let's play a game

- A player competes against the dealer.
- 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) **natural**, he then wins (or draw if dealer also has a natural)
- 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.
- 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.

Example 5.1: Blackjack (III)

Playing blackjack is naturally formulated as an **episodic finite MDP**.

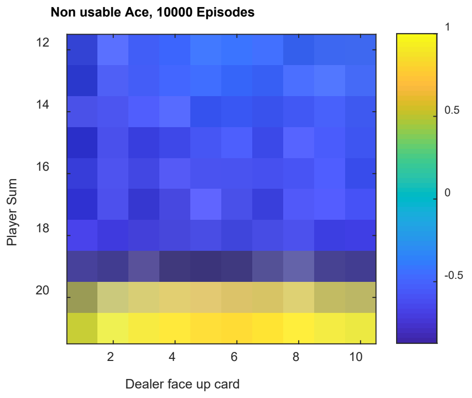
- Each game of blackjack is an episode.
- Rewards of +1, -1, and 0 are given for winning, losing, and drawing, respectively.
- We do not discount ($\gamma = 1$)
- The player's actions are to hit or to stick.
- The states depend on the player's cards and the dealer's showing card.
- The player makes decisions on the basis of three variables:
  - If he holds a usable ace (it counts as 11 without going bust)
  - His current sum (12:21)
  - The dealer's one showing card (1:10)
- This makes for a total of $N_s = 200$ states.
- **Dealer policy**, stick if sum is 17 or greater, hit otherwise.
- **Player policy** π, stick if sum is 20 or 21, hit otherwise.

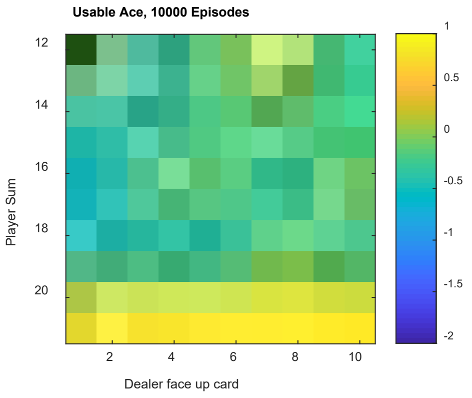
Example 5.1: Blackjack (IV)

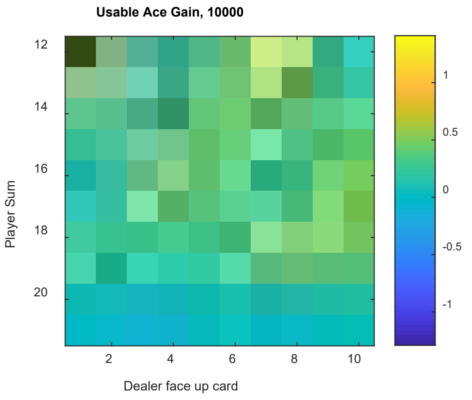
MC prediction for evaluation of v_π

Comparative with/without ace

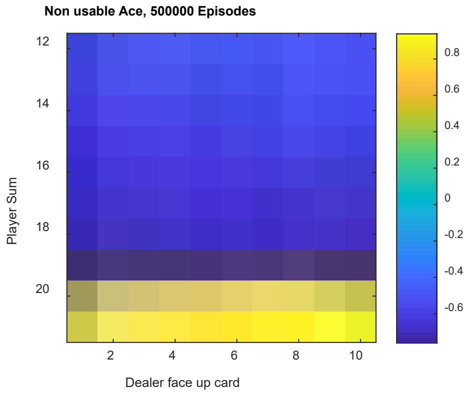
10000 games

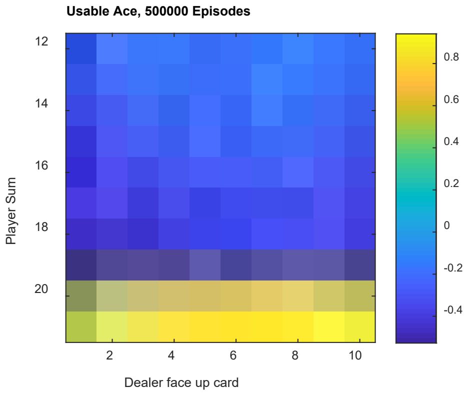


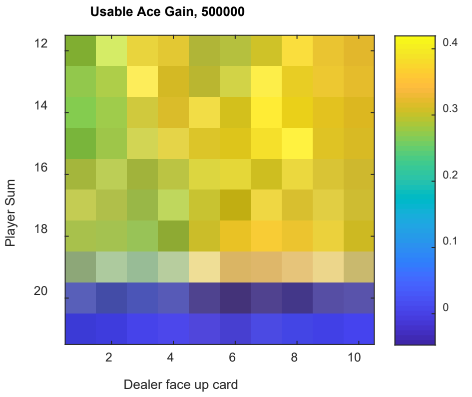




500000 games







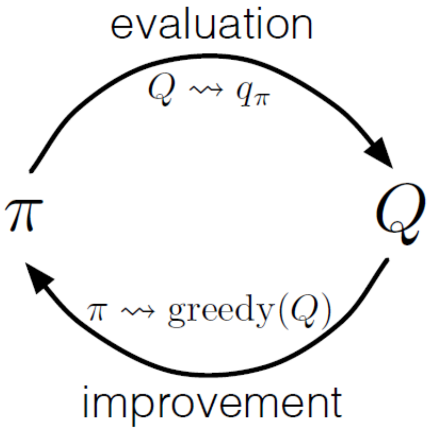
NON usable ace **Usable ace**

MC control

It is the MC version of classical policy iteration.

- GPI: To proceed according to the same pattern as in DP.

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



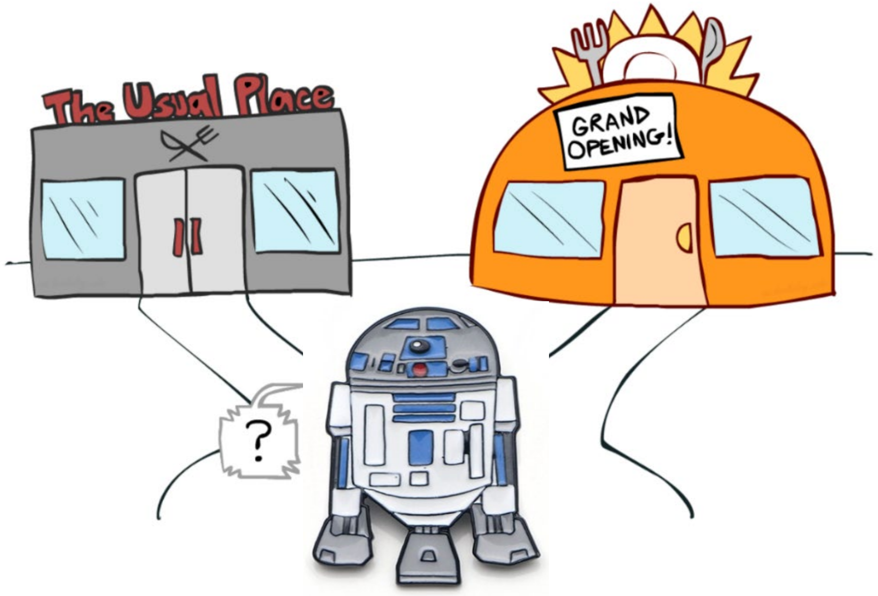
- Policy improvement step: $\pi(s) = \arg\max_a q(s,a)$

- Measurements and assumptions are made to obtain bounds on the magnitude and probability of error in the estimates.
- Sufficient steps have to be taken during each policy evaluation to assure that these bounds are sufficiently small.

Exploration vs. exploitation

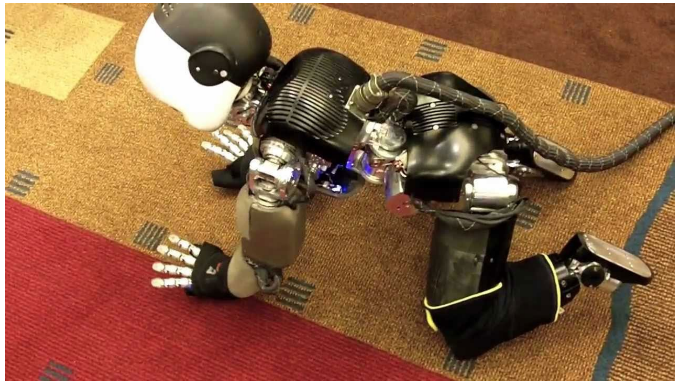
But, if our agent always takes the best actions from his current knowledge…

- How will he ever learn that other actions may be better than his current best one?
- How can we guarantee that all (s,a) pairs are visited?



Exploration vs. exploitation

If a robot gets a reward when crawling, it may get stuck and never learn to walk:



If it always explores possible actions, it will rarely walk and the reward at convergence may get too low.

Exploration vs. exploitation

Problem if π is a deterministic policy: following π one will observe returns only for one of the actions from each state.

How to modify our current learning so that agent does not get stuck in poor local optima? Dedicate effort to explore actions.

Let us solve it by exploring starts: consider all possible starting pairs (s, a) and generate many episodes. Average returns once all episodes have been observed.

MC control by exploring starts

Using the MC policy evaluation principles, it is natural to alternate between evaluation and improvement on an episode-by-episode basis…

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

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 π
  For each pair s, a appearing in the episode:
    $G \leftarrow$ the return that follows the first occurrence of s, a
    Append G to $Returns(s,a)$
    $Q(s,a) \leftarrow \text{average}(Returns(s,a))$ ← Action-value function estimation by averaging
  For each s in the episode:
    $\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

Example 5.1: Blackjack (V)

MC Control. Finding the optimum policy by exploring starts…

`repeat until` $a_\pi(s)$ does not change

- Initialize for all s: $Q(s, a) = 0$
- `repeat` for a number of episodes N

  Generate an episode starting from $(s_0, a_0(s_0))$
  - Choose s_0 (e.g. dealer deals cards and gamer plays until his sum ≥ 12)
  - Choose starting arbitrary policy $a_0(s_0)$: Hit/Stick equiprobable for all s_0
  - Note that all pairs $(s_0, a_0(s_0))$ have probability > 0

  Continue the episode following policy $a_\pi(s)$

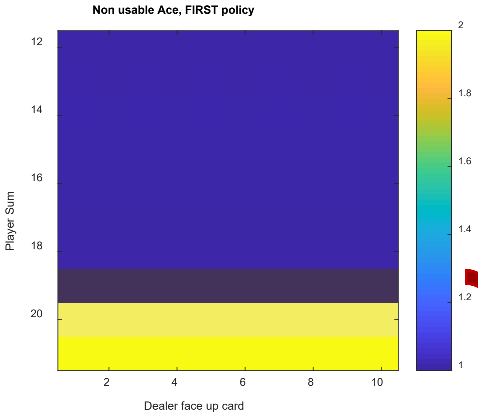
  `For each` pair $(s_0, a_0(s_0))$ and $(s, a_\pi(s))$ appearing in the episode, add G to Returns (s, a) list
- $Q(s, a) =$ `average`(Returns(s, a))
- `For each` s in the episode update actions: $a_\pi(s) = \arg\max_a Q(s, a)$

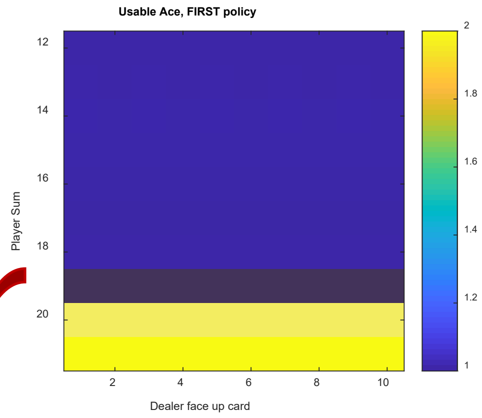
`end`

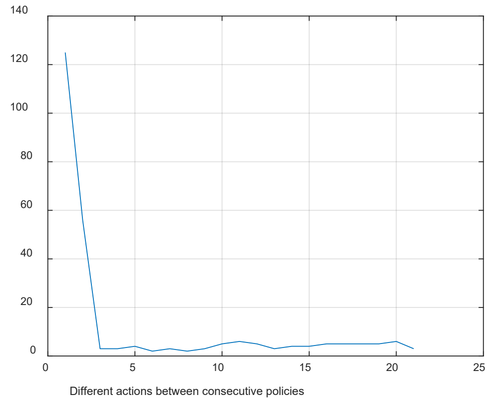
Example 5.1: Blackjack (VI)

MC control by exploring starts

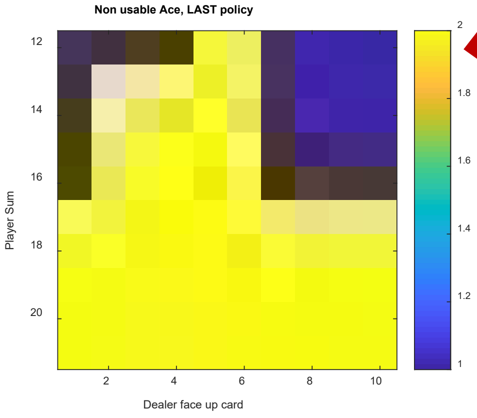
First policy: player hits until sum $\geq$ 20

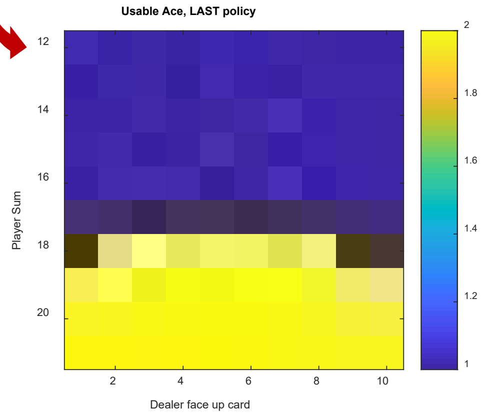






Last policy: Optimum policy





Number of different actions 1(Hit) or 2(Stick) between consecutive policies should become zero

NON usable ace

Usable ace

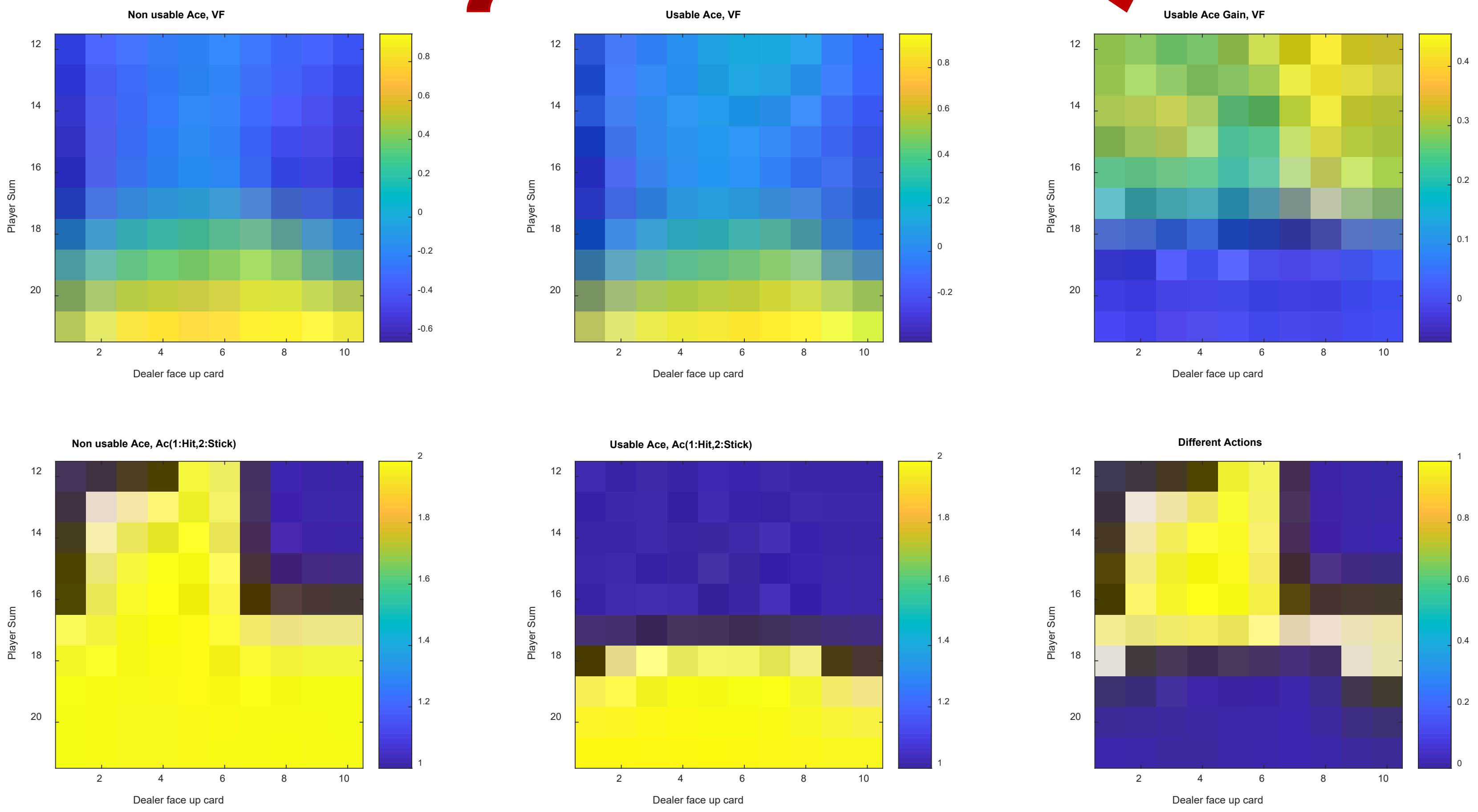
Example 5.1: Blackjack (VII)

MC control by exploring starts

Comparative with/without ace

Value function

Optimum policy
1: Hit
2: Stick



NON usable ace

Usable ace

MC control without exploring starts

Assuming all pairs (s,a) are accessible at the start of an episode is unrealistic in many applications, so let us define alternate strategies that allow exploration…

- On-policy method

- Off-policy method

MC control without exploring starts: on-policy

On-policy method: the policy is generally soft, meaning that $\pi(a|s) > 0$ for all s in $\mathcal{S}$ and for all a in $\mathcal{A}(s)$.

- It gradually shifts closer and closer to a deterministic optimal policy
- ε–greedy policy:

$$\pi(a \mid s) = \begin{cases} 1 - \varepsilon\left(1 - 1/m\right) & \text{if } a^* = \arg\max_{a \in \mathcal{A}} Q(s,a) \\ \varepsilon / m & \text{otherwise} \end{cases}$$

where $m = |\mathcal{A}(s)|$ is number of actions for state s.

MC control without exploring starts: on-policy

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

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 ε-soft policy

Repeat forever:
- (a) Generate an episode using π
- (b) For each pair s, a appearing in the episode:
  - $G \leftarrow$ the return that follows the first occurrence of s, a
  - Append G to $Returns(s, a)$
  - $Q(s, a) \leftarrow \text{average}(Returns(s, a))$
- (c) For each s in the episode:
  - $A^* \leftarrow \arg\max_a Q(s, a)$ (with ties broken arbitrarily)
  - For all $a \in \mathcal{A}(s)$:

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

All action-state pairs can be visited

MC control without exploring starts: on-policy

If we want to converge (and reduce variance) to the optimal policy we need to reduce exploration, so the value of ε could be reduced progressively.

However, some level of exploration is needed in case the environment is changing!

MC control without exploring starts: off-policy

Off-policy method requires the concept of importance sampling:

For two policies that we will use in parallel…

- **Target policy** $\pi(a|s)$: it is learned about and eventually becomes the optimal policy.
- **Behavior policy** $b(a|s)$: it is more exploratory and is used to generate behavior (e.g. following an ε-greedy policy).

we can determine the performance of $\pi(a|s)$ by evaluating the performance of $b(a|s)$ provided that $\pi(a|s) > 0$ if $b(a|s) > 0$

Off-policy methods:

- Greater variance and slower to converge.
- More powerful and general, external expertise can be plugged in $b(a|s)$.

MC control without exploring starts: off-policy

Let us define the relative probabilities of a sequence under the two policies:

- Given a starting state S_t, the probability of the subsequent state-action trajectory $A_t\, S_t\, A_{t+1}\, S_{t+1} \ldots S_T$ occurring under any policy π is

$$\begin{aligned}
&\Pr\left\{A_t S_{t+1} A_{t+1} \ldots S_T \mid S_t, A_{t:T-1} \sim \pi\right\} \\
&\quad = \pi\left(A_t \mid S_t\right) p\left(S_{t+1} \mid S_t, A_t\right) \pi\left(A_{t+1} \mid S_{t+1}\right) \cdots p\left(S_T \mid S_{T-1}, A_{T-1}\right) \\
&\quad = \prod_{k=t}^{T-1} \pi\left(A_k \mid S_k\right) p\left(S_{k+1} \mid S_k, A_k\right)
\end{aligned}$$

- Relative probability of the trajectory under the target and behavior policies

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi\left(A_k \mid S_k\right) p\left(S_{k+1} \mid S_k, A_k\right)}{\prod_{k=t}^{T-1} b\left(A_k \mid S_k\right) p\left(S_{k+1} \mid S_k, A_k\right)} = \prod_{k=t}^{T-1} \frac{\pi\left(A_k \mid S_k\right)}{b\left(A_k \mid S_k\right)}$$

It is assumed that these probabilities are known

MC control without exploring starts: off-policy

The algorithm uses a batch of observed episodes following policy $b(a|s)$ to estimate $v_\pi(s)$.

- $\mathcal{T}(s)$ Set of all time steps (first or every time step) in which state s is visited in all episodes
- $T(t)$ First time of termination following t
- G_t Return after t up through $T(t)$
- $\{G_t\}_{t \in \mathcal{T}(s)}$ Returns associated to state s
- $\{\rho_{t:T(t)-1}\}_{t \in \mathcal{T}(s)}$ Importance sampling ratios

Ordinary Importance Sampling

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

Weighted Importance Sampling

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Incremental implementation

- Monte Carlo prediction methods can be implemented incrementally, on an episode-by-episode basis.
- For MC-off policy weighted importance sampling: suppose we have a sequence of returns $G_1 G_2 \ldots G_{n-1}$, all starting in the same state and each with a corresponding random weight W_i. As we want to form:

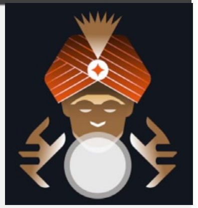
$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k} \quad n \geq 2 \qquad \left(\text{e.g. } W_k = \rho_t^{T(t)}\right)$$

the update rule for V_n is

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}\left[G_n - V_n\right] \quad n \geq 1 \qquad C_{n+1} \doteq C_n + W_{n+1}$$

Complete episode-by-episode incremental algorithm for MC policy evaluation:

Off-policy MC prediction, for estimating $Q \approx q_\pi$



Input: an arbitrary target policy π

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
- $Q(s,a) \leftarrow$ arbitrary
- $C(s,a) \leftarrow 0$

Repeat forever:
- $b \leftarrow$ any policy with coverage of π
- Generate an episode using b:
  - $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
- $G \leftarrow 0$
- $W \leftarrow 1$
- For $t = T-1, T-2, \ldots$ down to 0:
  - $G \leftarrow \gamma G + R_{t+1}$
  - $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
  - $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
  - If $W = 0$ then exit For loop

Incremental evaluation of *q*(*a*,*s*)

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

- $Q(s, a) \leftarrow$ arbitrary
- $C(s, a) \leftarrow 0$
- $\pi(s) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)

Repeat forever:

- $b \leftarrow$ any soft policy
- Generate an episode using b:
  - $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
- $G \leftarrow 0$
- $W \leftarrow 1$
- For $t = T-1, T-2, \ldots$ down to 0:
  - $G \leftarrow \gamma G + R_{t+1}$
  - $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
  - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
  - If $A_t \neq \pi(S_t)$ then exit For loop
  - $W \leftarrow W \frac{1}{b(A_t|S_t)}$

Incremental evaluation

GPI principle for control

On-policy vs Off-policy

Agent takes actions inside an environment

Another agent takes actions in an environment, your agent is trained on those recorded trajectories. The optimal policy is obtained from the interaction between *b*(s|a) and the environment

On-policy	Off-policy
Agent can pick actions	Agent can't pick actions
Most obvious setup	Learning with exploration, playing without exploration
Agent always follows his own policy	Learning from expert (expert is imperfect)

Summary

- Advantages over DP methods:
  - Learn optimal behavior directly from interaction with the environment
  - They can be used with simulation or sample models
  - Easy and efficient
  - They may be less harmed by violations of the Markov property
- GPI:
  - Policy Evaluation: Average many returns that start in the state.
  - Control: Approximate action-value function.
- MC intermix policy evaluation and policy improvement steps:
  - Episode-by-episode basis
  - They can be incrementally implemented on an episode-by-episode basis.
- Maintaining sufficient exploration is an issue in MC prediction:
  - On-policy: To assume that episodes begin with state/action pairs randomly selected to cover all possibilities.
  - Off-policy: To learn the value function of a target policy from data generated by a different behavior policy.