

# CS 687: Phase Two

---

- Model-free Solution Methods:
  - Monte-Carlo methods (Chapter 5)
  - Temporal-difference learning (Chapter 6)
  - Eligibility traces (Chapter 7)
- How to estimate value functions from simulation?
  - Full trajectory methods
  - Bootstrap methods
  - Hybrid methods

# Homework for Chapter 4

---

- ❑ Problem set 4.1, 4.2, 4.3, 4.5, 4.6, 4.7, 4.9
- ❑ Due Thu 10/17

# Chapter 5: Monte Carlo Methods

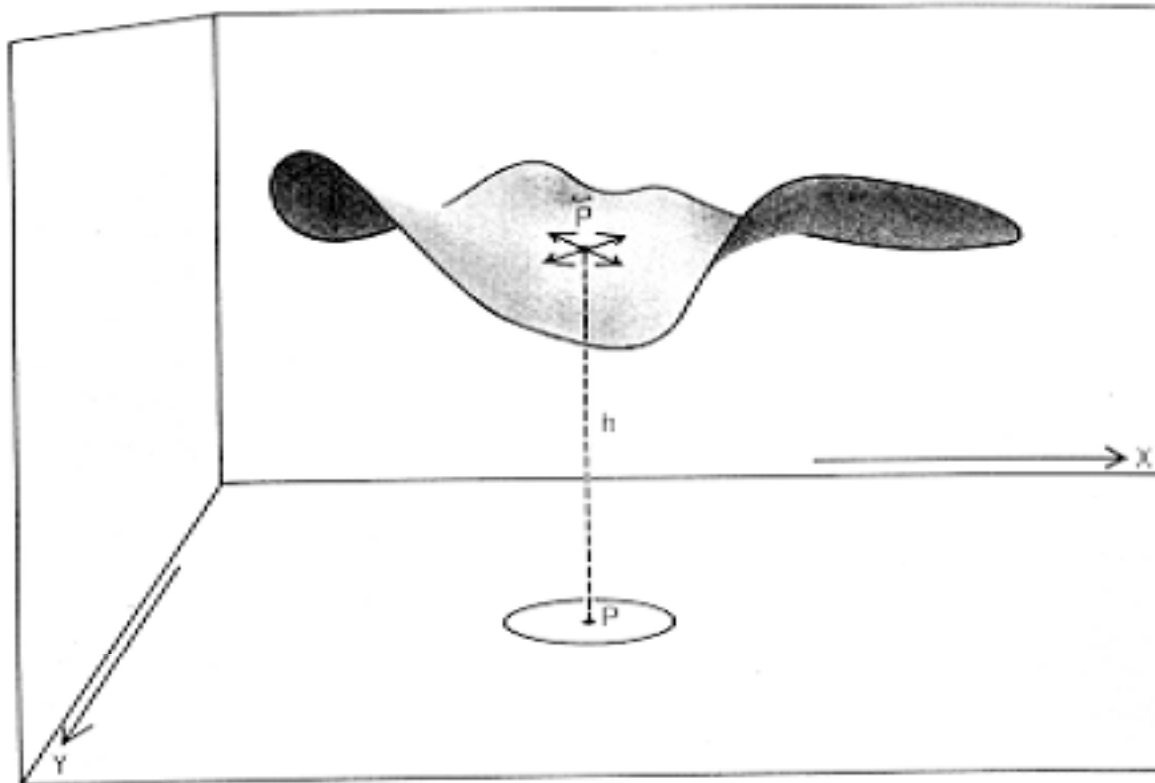
---

- ❑ Monte Carlo methods learn from *complete* sample returns
  - Only defined for *episodic* tasks
- ❑ Monte Carlo methods learn directly from experience
  - *On-line*: No model necessary and still attains optimality
  - *Simulated*: No need for a *full* model

# The Power of Monte Carlo

e.g., Elastic Membrane (Dirichlet Problem)

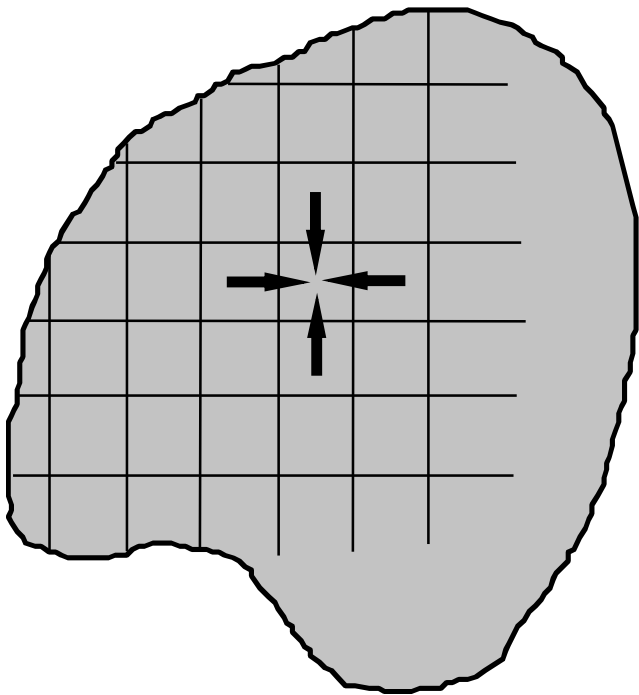
How do we compute the shape of the membrane or bubble?



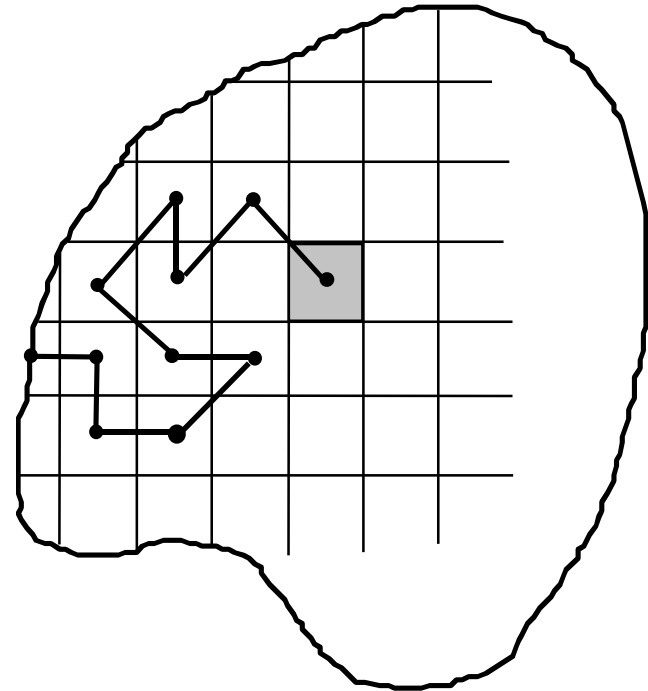
# Two Approaches

---

Relaxation



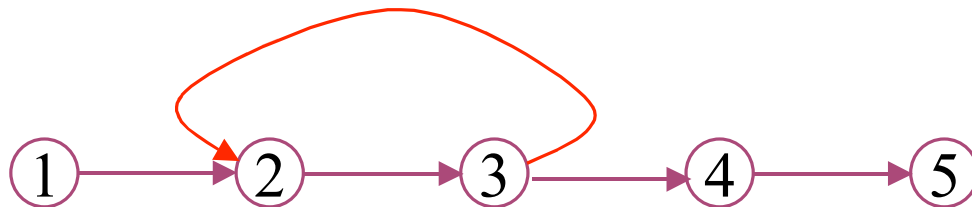
Kakutani's algorithm, 1945



# Monte Carlo Policy Evaluation

---

- ❑ *Goal:* learn  $V^\pi(s)$
- ❑ *Given:* some number of episodes under  $\pi$  which contain  $s$
- ❑ *Idea:* Average returns observed after visits to  $s$



- ❑ *Every-Visit MC:* average returns for *every* time  $s$  is visited in an episode
- ❑ *First-visit MC:* average returns only for *first* time  $s$  is visited in an episode
- ❑ Both converge asymptotically

# First-visit Monte Carlo policy evaluation

---

## □ Initialize

- $\pi$  is the policy to be evaluated
- $V$  = an arbitrary scalar value function
- $Returns(s)$  is an empty list for all states in  $S$

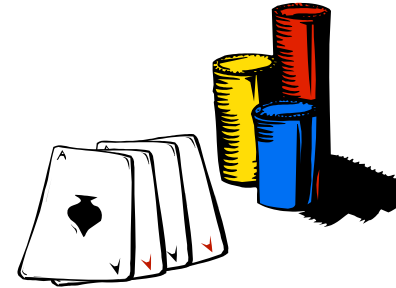
## □ Repeat Forever

- Generate an episode using  $\pi$
- For each state  $s$  appearing in the episode
  - $R$  is the return following the first occurrence of  $s$
  - Append  $R$  to  $Returns(s)$
  - $V(s) = \text{Average}(Returns(s))$

# Blackjack example

---

- ❑ **Object:** Have your card sum be greater than the dealers without exceeding 21.
- ❑ **States** (200 of them):
  - current sum (12-21)
  - dealer's showing card (ace-10)
  - do I have a useable ace?
- ❑ **Reward:** +1 for winning, 0 for a draw, -1 for losing
- ❑ **Actions:** stick (stop receiving cards), hit (receive another card)
- ❑ **Policy:** Stick if my sum is 20 or 21, else hit

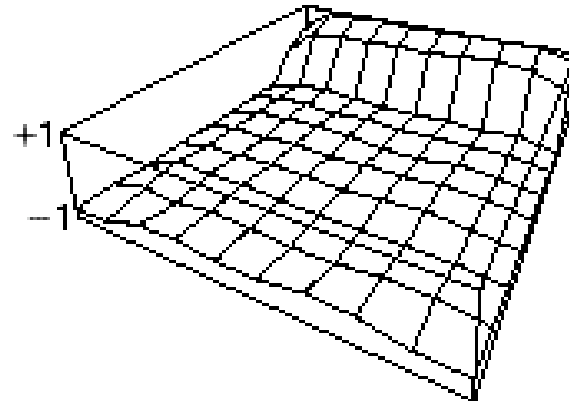
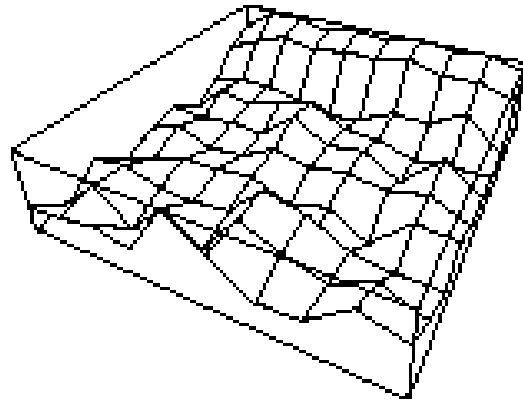


# Blackjack value functions

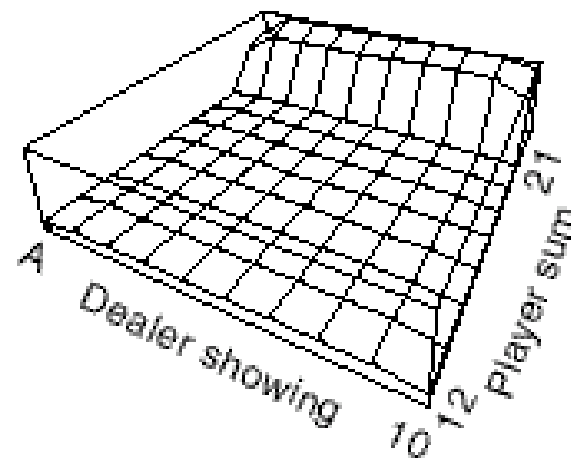
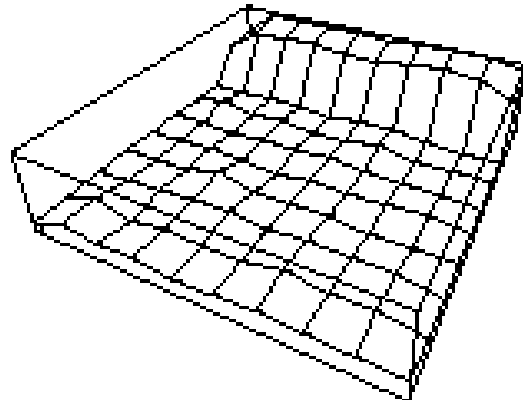
After 10,000 episodes

After 500,000 episodes

Usable  
ace



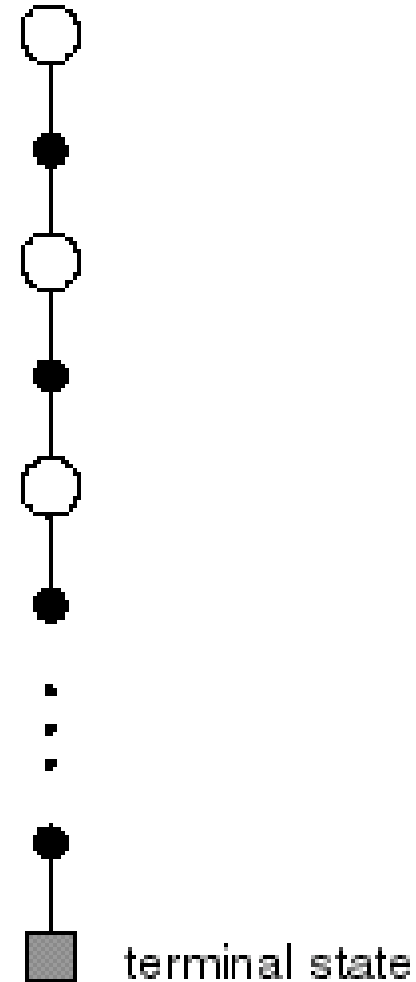
No  
usable  
ace



# Backup diagram for Monte Carlo

---

- ❑ Entire episode included
- ❑ Only one choice at each state (unlike DP)
- ❑ MC does not bootstrap
- ❑ Time required to estimate one state does not depend on the total number of states



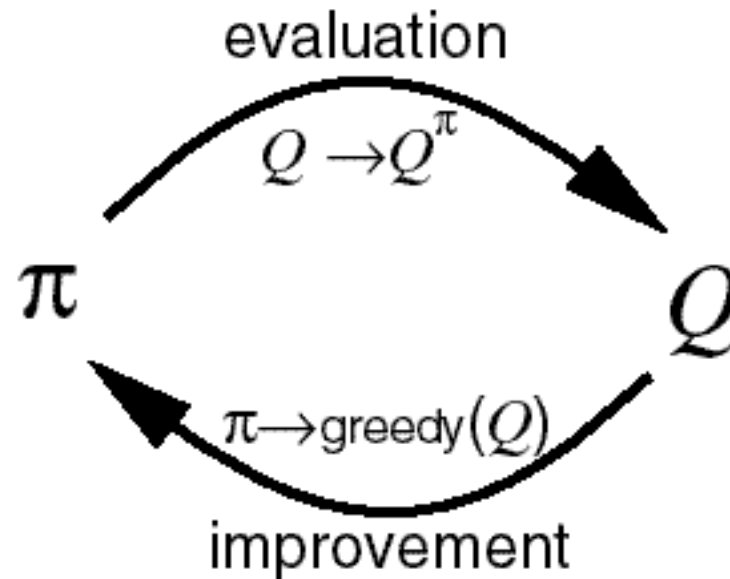
# Monte Carlo Estimation of Action Values (Q)

---

- ❑ Monte Carlo is most useful when a model is not available
  - We want to learn  $Q^*$
- ❑  $Q^\pi(s,a)$  - average return starting from state  $s$  and action  $a$  following  $\pi$
- ❑ Also converges asymptotically *if* every state-action pair is visited
- ❑ *Exploring starts*: Every state-action pair has a non-zero probability of being the starting pair

# Monte Carlo Control

---



- ❑ **MC policy iteration:** Policy evaluation using MC methods followed by policy improvement
- ❑ **Policy improvement step:** greedify with respect to value (or action-value) function

# Convergence of MC Control

---

- Policy improvement theorem tells us:

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

- This assumes exploring starts and infinite number of episodes for MC policy evaluation
- To solve the latter:
  - update only to a given level of performance
  - alternate between evaluation and improvement per episode

# Monte Carlo 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

Fixed point is optimal policy  $\square^*$

Proof is open question

Repeat forever:

(a) Generate an episode using exploring starts and  $\pi$

(b) For each pair  $s, a$  appearing in the episode:

$R \leftarrow$  return following the first occurrence of  $s, a$

Append  $R$  to  $Returns(s, a)$

$Q(s, a) \leftarrow \text{average}(Returns(s, a))$

(c) For each  $s$  in the episode:

$\pi(s) \leftarrow \text{argmax}_a Q(s, a)$

# Monte Carlo Exploring Starts

---

Initialize for all states  $s \in S$ , and actions  $a \in A(s)$

$Q(s,a) \leftarrow$  arbitrary

$\pi(s) \leftarrow$  arbitrary

Returns( $s,a$ )  $\leftarrow$  empty list

Repeat forever:

(a) Generate an episode using ES and  $\pi$

(b) For each pair  $s,a$ , appearing in the episode

$R \leftarrow$  return following first occurrence of  $s,a$

Append  $R$  to Returns( $s,a$ )

$Q(s,a) \leftarrow$  average(Returns( $s,a$ ))

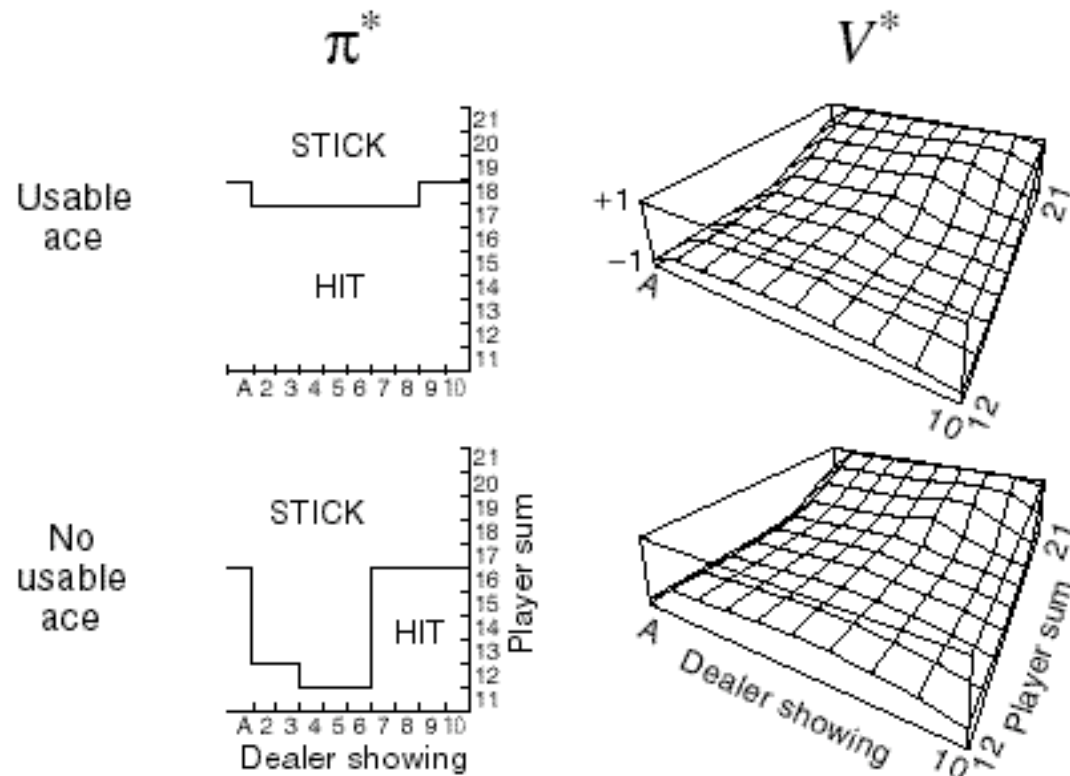
(c) For each  $s$  in the episode:

$\pi(s) \leftarrow \arg \max_a Q(s,a)$

Convergence  
recently proved  
In Tsitsiklis,  
JAIR 2002!

# Blackjack example continued

- Exploring starts
- Initial policy as described before



# On-policy Monte Carlo Control

---

- ❑ *On-policy*: learn about policy currently executing
- ❑ How do we get rid of exploring starts?
  - Need *soft* policies:  $\pi(s,a) > 0$  for all  $s$  and  $a$
  - e.g.  $\epsilon$ -soft policy:

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

non-max

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

greedy

- ❑ Similar to GPI: move policy *towards* greedy policy (i.e.  $\epsilon$ -soft)
- ❑ Converges to best  $\epsilon$ -soft policy

# On-policy MC Control

---

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

$Q(s, a) \leftarrow$  arbitrary

$Returns(s, a) \leftarrow$  empty list

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

Repeat forever:

(a) Generate an episode using  $\pi$

(b) For each pair  $s, a$  appearing in the episode:

$R \leftarrow$  return following the first occurrence of  $s, a$

Append  $R$  to  $Returns(s, a)$

$Q(s, a) \leftarrow \text{average}(Returns(s, a))$

(c) For each  $s$  in the episode:

$a^* \leftarrow \arg \max_a Q(s, a)$

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

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

# Off-policy Monte Carlo control

---

- ❑ Behavior policy generates behavior in environment
- ❑ Estimation policy is policy being learned about
- ❑ Average returns from behavior policy by probability their probabilities in the estimation policy

# Learning about $\gamma$ while following $\pi'$

Suppose we have  $n_s$  returns,  $R_t^i(s)$ , from state  $s$ , each with probability  $p_t^i(s)$  of being generated by  $\pi$  and probability  $p_t^j(s)$  of being generated by  $\pi'$ . Then we can estimate:

$$V^{\pi}(s) = \frac{\sum_{i=1}^{n_s} \frac{p_t^i(s)}{p_t^j(s)} R_t^i(s)}{\sum_{i=1}^{n_s} \frac{p_t^i(s)}{p_t^j(s)}}$$

which depends on the environmental probabilities  $p_t^i(s)$  and  $p_t^j(s)$ . However,

$$p_t^i(s) = \prod_{k=1}^{n(s)-1} \pi(s_k, a_{ik}) P_{s_k s_{k+1}}^{\pi}$$

and

$$\frac{p_t^i(s)}{p_t^j(s)} = \frac{\prod_{k=1}^{n(s)-1} \pi(s_k, a_{ik}) P_{s_k s_{k+1}}^{\pi}}{\prod_{k=1}^{n(s)-1} \pi'(s_k, a_{jk}) P_{s_k s_{k+1}}^{\pi'}} = \prod_{k=1}^{n(s)-1} \frac{\pi(s_k, a_{ik})}{\pi'(s_k, a_{jk})}$$

Thus the weight needed,  $p_t^i(s)/p_t^j(s)$ , depends only on the two policies and not at all on the environmental dynamics.

# Off-policy MC control

---

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

$Q(s, a) \leftarrow$  arbitrary

$N(s, a) \leftarrow 0$  ; Numerator and

$D(s, a) \leftarrow 0$  ; Denominator of  $Q(s, a)$

$\pi \leftarrow$  an arbitrary deterministic policy

Repeat forever:

(a) Select a policy  $\pi'$  and use it to generate an episode:

$$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T, s_T$$

(b)  $\tau \leftarrow$  latest time at which  $a_t \neq \pi(s_t)$

(c) For each pair  $s, a$  appearing in the episode after  $\tau$ :

$t \leftarrow$  the time of first occurrence (after  $\tau$ ) of  $s, a$

$$w \leftarrow \frac{\gamma^{t-\tau-1}}{\sum_{k=\tau}^{T-1} \gamma^{k-\tau-1}}$$

$$N(s, a) \leftarrow N(s, a) + wR_t$$

$$D(s, a) \leftarrow D(s, a) + w$$

$$Q(s, a) \leftarrow \frac{N(s, a)}{D(s, a)}$$

(d) For each  $s \in \mathcal{S}$ :

$$\pi(s) \leftarrow \operatorname{argmax}_a Q(s, a)$$

# Incremental Implementation

---

- MC can be implemented incrementally
  - saves memory
- Compute the weighted average of each return

$$V_n = \frac{\sum_{k=1}^n w_k R_k}{\sum_{k=1}^n w_k}$$

non-incremental

$$V_{n+1} = V_n + \frac{w_{n+1}}{W_{n+1}} [R_{n+1} - V_n]$$

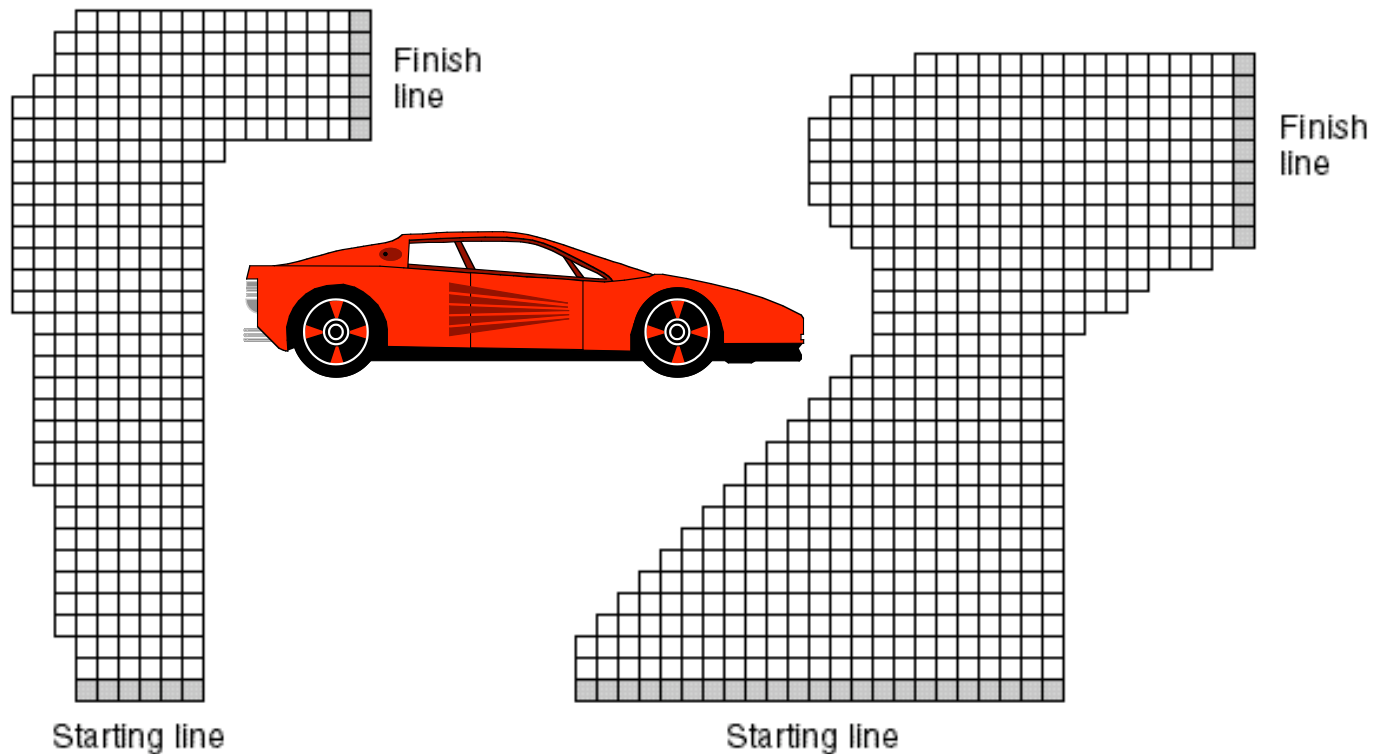
$$W_{n+1} = W_n + w_{n+1}$$

$$V_0 = W_0 = 0$$

incremental equivalent

# Racetrack Exercise

- ▣ *States*: grid squares, velocity horizontal and vertical
- ▣ *Rewards*: -1 on track, -5 off track
- ▣ *Actions*: +1, -1, 0 to velocity
- ▣  $0 < \text{Velocity} < 5$
- ▣ Stochastic: 50% of the time it moves 1 extra square up or right



# Summary

---

- ❑ MC has several advantages over DP:
  - Can learn directly from interaction with environment
  - No need for full models
  - No need to learn about ALL states
  - Less harm by Markovian violations (later in book)
- ❑ MC methods provide an alternate policy evaluation process
- ❑ One issue to watch for: maintaining sufficient exploration
  - exploring starts, soft policies
- ❑ No bootstrapping (as opposed to DP)