

# Sequential Decision Making and Reinforcement Learning

(SDMRL)

Model-free RL

---

Sarah Keren

The Taub Faculty of Computer Science  
Technion - Israel Institute of Technology

# Acknowledgments

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
Methods

Policy Search  
Methods

- David Silver's course on RL:  
*<https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver>*
- Slides by Malte Helmert, Carmel Domshlak, Erez Karpas and Alexander Shleyfman.

## Recap

---

# Anatomy of RL Algorithms

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

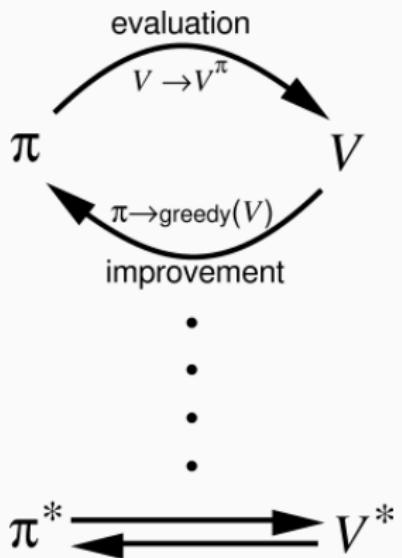
Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods



# Model-Free vs. Model-Based RL

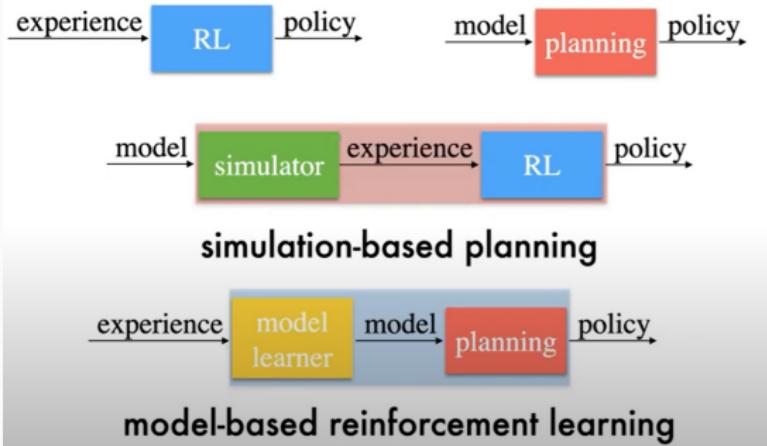


Figure 1: By Michael Littman

<https://www.youtube.com/watch?v=45FKxa3qPHo>

# Model-Free vs. Model-Based RL

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Model Free

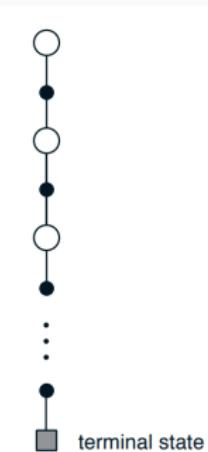
Model Based

## Model Free RL: Monte Carlo

---

# Monte Carlo (MC)

- MC methods learn directly from episodes of experience
- MC is model-free: no knowledge of MDP transitions / rewards
- MC learns from complete episodes: no bootstrapping.
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to episodic MDPs: all episodes must terminate



Reinforcement Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Sarah Keren

# Monte Carlo Simulation-Rollout

- A Monte Carlo simulation is a statistical technique used to analyze the behavior of complex systems and processes that are influenced by random variables.

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Monte Carlo Simulation-Rollout

- A Monte Carlo simulation is a statistical technique used to analyze the behavior of complex systems and processes that are influenced by random variables.
- Each rollout involves simulating a sequence of actions, typically by selecting actions according to some policy (e.g., uniformly at random, or based on a heuristic) until a terminal state is reached.

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Monte Carlo Simulation-Rollout

- A Monte Carlo simulation is a statistical technique used to analyze the behavior of complex systems and processes that are influenced by random variables.
- Each rollout involves simulating a sequence of actions, typically by selecting actions according to some policy (e.g., uniformly at random, or based on a heuristic) until a terminal state is reached.
- Typically, a large number of simulations are used to estimate the probabilities and distributions of different outcomes.

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Monte Carlo Simulation-Rollout

- A Monte Carlo simulation is a statistical technique used to analyze the behavior of complex systems and processes that are influenced by random variables.
- Each rollout involves simulating a sequence of actions, typically by selecting actions according to some policy (e.g., uniformly at random, or based on a heuristic) until a terminal state is reached.
- Typically, a large number of simulations are used to estimate the probabilities and distributions of different outcomes.
- We will use this in different algorithms



Reinforcement Learning  
(SDMRL)  
Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

# Monte-Carlo Policy Evaluation

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- **Reminder:**

- Return is (typically) the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

- Value function is (typically) the expected return:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

- **Objective:** learn  $V_\pi$  from episodes of experience under policy  $\pi$

$$S_1, A_1, R_2, \dots, S_k \sim \pi$$

- Estimate  $V_\pi(s)$  as average total reward of epochs containing  $s$  (calculating from  $s$  to end of epoch).

# Monte-Carlo Policy Evaluation

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Reminder:

- Return is (typically) the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

- Value function is (typically) the expected return:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

- **Objective:** learn  $V_\pi$  from episodes of experience under policy  $\pi$

$$S_1, A_1, R_2, \dots, S_k \sim \pi$$

- Estimate  $V_\pi(s)$  as average total reward of epochs containing  $s$  (calculating from  $s$  to end of epoch).

Monte-Carlo policy evaluation uses empirical mean return instead of expected return. **Why ?**

# Monte-Carlo Policy Evaluation

## Key Idea

Use observed reward-to-go of the state as the direct evidence of the actual expected utility of that state.

- Reward-to-go of a state  $s$  = the sum of the (discounted) rewards from that state until a terminal state is reached
- Two versions to evaluate state  $s$ :
  - The first time-step  $t$  that state  $s$  is visited in an episode
  - Every time-step  $t$  that state  $s$  is visited in an episode
- Increment visit counter  $N(s) \leftarrow N(s) + 1$
- Increment total return  $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return  $V(s) = S(s)/N(s)$
- By law of large numbers,  $V(s) \rightarrow V_\pi(s)$  as  $N(s) \rightarrow \infty$  (averaging the reward-to-go samples will converge to true value at state)

Reinforcement Learning  
(SDMRL)  
Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

# Monte-Carlo Policy Evaluation - Blackjack

- States (200 of them):
  - Current sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a “useable” ace? (yes-no)
- Action stick: Stop receiving cards (and terminate)
- Action twist: Take another card (no replacement)
- Reward for stick:
  - +1 if sum of cards > sum of dealer cards
  - 0 if sum of cards = sum of dealer cards
  - -1 if sum of cards < sum of dealer cards
- Reward for twist:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum of cards < 12



Reinforcement Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

# Monte-Carlo Policy Evaluation

Reinforcement Learning  
(SDMRL)

Sarah Keren

Recap

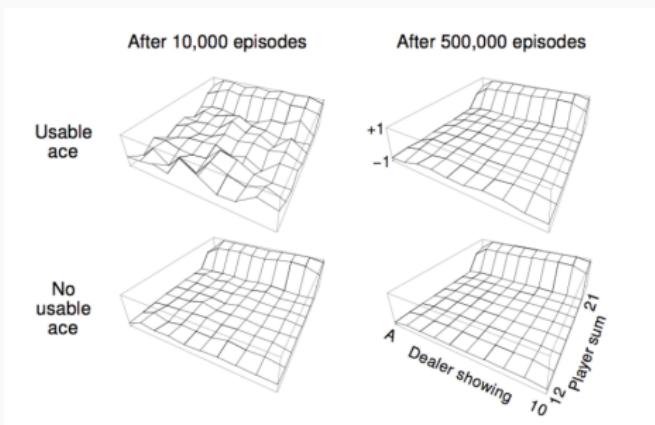
Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

## Blackjack Value Function after Monte-Carlo Learning



# Monte-Carlo Policy Evaluation

Reinforcement Learning  
(SDMRL)

Sarah Keren

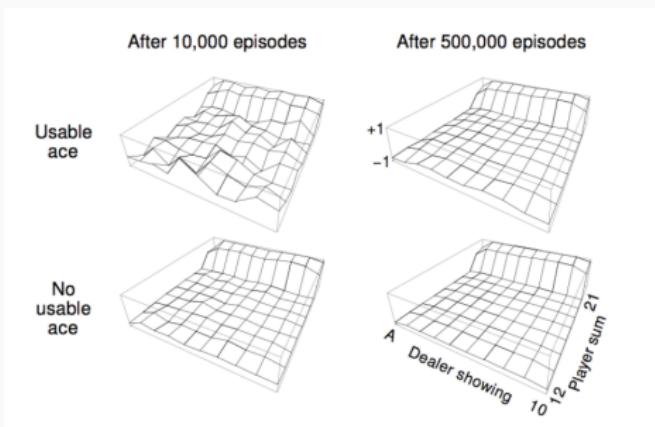
Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods



Policy - stick if sum of cards  $\geq 20$ , otherwise twist.

# Monte-Carlo: Prediction (Evaluation)

Reinforcement  
Learning

(SDMRL)

Sarah Keren

## Prediction:

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  $\pi$

For each state  $s$  appearing in the episode:

$G \leftarrow$  return following the first occurrence of  $s$

Append  $G$  to  $Returns(s)$

$V(s) \leftarrow$  average( $Returns(s)$ )

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Monte-Carlo: Prediction (Evaluation)

Reinforcement  
Learning

(SDMRL)

Sarah Keren

## Prediction:

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  $\pi$

For each state  $s$  appearing in the episode:

$G \leftarrow$  return following the first occurrence of  $s$

Append  $G$  to  $Returns(s)$

$V(s) \leftarrow$  average( $Returns(s)$ )

How can we use this for control ?

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Monte-Carlo: From Prediction (Evaluation) to Control

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Control:

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:

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

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Monte-Carlo: From Prediction (Evaluation) to Control

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Control:

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:

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

Problem with this approach?

# Monte-Carlo: From Prediction (Evaluation) to Control

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Use a random policy to simulate many (!) trajectories
- Compute the q-value of each state-action pair
- Update  $\pi$  by taking the max action.

**Problem with this approach?**

# Monte-Carlo Policy Control - Taxi

Reinforcement Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

```
+-----+  
|R: | : :G|  
| : | : : |  
| : : : : |  
| : | : | :B|  
|Y| : |B: |  
+-----+  
          (North)  
num episodes completed:  1  
total rewards:           -133  
mean rewards per episode: -133.00
```

```
def build_decision_dict(raw_data):  
    # Nested dictionary for: State -> Action -> Reward List  
    state_action_scores = defaultdict(lambda: defaultdict(lambda: []))  
    for trajectory in raw_data:  
        reward_sum = 0  
        # iterate backwards to calculate the return G of each observed state action pair  
        for state, action, reward in reversed(list(zip(trajectory.observations, trajectory.actions, trajectory.rewards))):  
            reward_sum += reward  
            state_action_scores[state][action].append(reward_sum)  
  
    for state, action_values in state_action_scores.items():  
        for action, values_list in action_values.items():  
            # Calculate the mean of all returns for a state action pair  
            state_action_scores[state][action] = np.mean(values_list)  
        # For each state choose the action with the highest mean return  
        state_action_scores[state] = max(state_action_scores[state], key=state_action_scores[state].get)  
    return state_action_scores
```

[https://github.com/CLAIR-LAB-TECHNION/FSTMA-course/blob/main/tutorials/tut03/Monte\\_Carlo.ipynb](https://github.com/CLAIR-LAB-TECHNION/FSTMA-course/blob/main/tutorials/tut03/Monte_Carlo.ipynb)

# Monte-Carlo Control - Taxi

Reinforcement Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

```
env.seed(seed)

policy = calc_final_policy(RandomTraveler, 100, "mcc_100")
evaluate_and_print(policy)
policy = calc_final_policy(RandomTraveler, 1000, "mcc_1000")
evaluate_and_print(policy)
policy = calc_final_policy(RandomTraveler, 10000, "mcc_10000")
evaluate_and_print(policy)
policy = calc_final_policy(RandomTraveler, 100000, "mcc_100000")
evaluate_and_print(policy)
```

100%  10000/10000 [00:01<00:00, 5139.08it/s]

Monte Carlo Control Policy

-----  
total reward over all episodes: -1306708  
mean reward per episode: -130.6708

100%  10000/10000 [00:01<00:00, 5334.19it/s]

Monte Carlo Control Policy

-----  
total reward over all episodes: -1242335  
mean reward per episode: -124.2335

100%  10000/10000 [00:01<00:00, 6415.93it/s]

Monte Carlo Control Policy

-----  
total reward over all episodes: -729155  
mean reward per episode: -72.9155

100%  10000/10000 [00:01<00:00, 7840.29it/s]

Monte Carlo Control Policy

-----  
total reward over all episodes: -1544428  
mean reward per episode: -15.44428

# Generalised Policy Iteration With Monte-Carlo Evaluation

Reinforcement Learning  
(SDMRL)

Sarah Keren

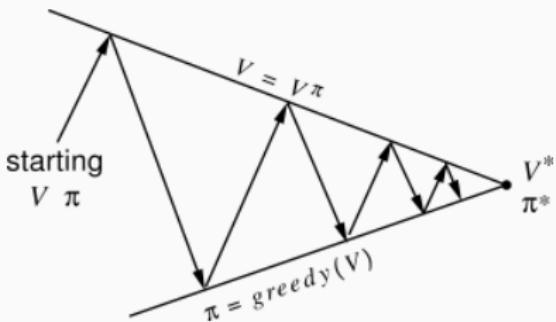
Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods



Policy evaluation: Monte-Carlo policy evaluation,  $V = V_\pi$ ?

Policy improvement: Greedy policy improvement?

# Model-Free Policy Iteration Using Action-Value Function

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

$$\pi'(s) = \arg \max_{a \in A} \sum_{s' \in \mathcal{S}} P(s'|s, a) [R(s, a, s') + \gamma V^\pi(s')] ,$$

$$\pi'(s) = \arg \max_{a \in A} Q(s, a)$$

# Model-Free Policy Iteration Using Action-Value Function

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

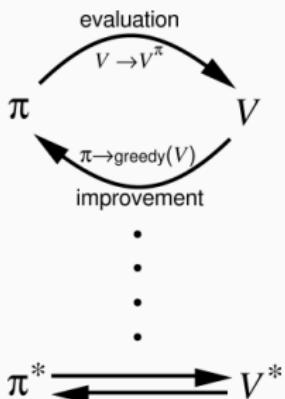
Greedy policy improvement over  $V(s)$  requires model of MDP

$$\pi'(s) = \arg \max_{a \in A} \sum_{s' \in \mathcal{S}} P(s'|s, a) [R(s, a, s') + \gamma V^\pi(s')] ,$$

Greedy policy improvement over  $Q(s, a)$  is model-free

$$\pi'(s) = \arg \max_{a \in A} Q(s, a)$$

# Generalised Policy Iteration With Monte-Carlo Evaluation



- **Policy evaluation:** Monte-Carlo policy evaluation.  $Q_\pi(s, a)$
- **Policy improvement:** Greedy policy improvement. How?

# Example of Greedy Action Selection

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

There are two doors in front of you:

You open the left door and get reward 0

$$V(\text{left}) = 0$$

You open the right door and get reward +1

$$V(\text{right}) = +1$$

You open the right door and get reward +2

$$V(\text{right}) = +2$$

You open the right door and get reward +2

$$V(\text{right}) = +2$$

.

.

.

Are you sure you've chosen the best door?



# $\epsilon$ -Greedy Exploration

- Simplest idea for ensuring continual exploration
- All  $m$  actions are tried with non-zero probability
- With probability  $1 - \epsilon$  choose the greedy action
- With probability  $\epsilon$  choose an action at random

# Policy Improvement Theorem\*

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

## Theorem

Let  $\pi$  and  $\pi'$  be any pair of deterministic policies. If for all  $s \in \mathcal{S}$   $Q_\pi(s, \pi'(s)) \geq \mathcal{V}_\pi(s)$  then for all  $s \in \mathcal{S}$ ,  $\mathcal{V}_{\pi'}(s) \geq \mathcal{V}_\pi(s)$ . i.e.,  $\pi'$  is an improvement over  $\pi$ .

$\pi'$ , that can either exploit or explore, must be at least as good as  $\pi$

# $\epsilon$ -Greedy Policy Improvement\*

## Theorem

For any  $\epsilon$ -greedy policy  $\pi$ , if the  $\epsilon$ -greedy policy  $\pi'$  with respect to  $Q_\pi$  is an improvement, then  $V_{\pi'}(s) \geq V_\pi(s)$

$$Q_\pi(s, \pi'(s)) = \sum_{a \in \mathcal{A}} \pi'(a|s) Q_\pi(s, a)$$

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# $\epsilon$ -Greedy Policy Improvement\*

## Theorem

For any  $\epsilon$ -greedy policy  $\pi$ , if the  $\epsilon$ -greedy policy  $\pi'$  with respect to  $Q_\pi$  is an improvement, then  $V_{\pi'}(s) \geq V_\pi(s)$

$$\begin{aligned} Q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) Q_\pi(s, a) \\ &= \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} Q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} Q_\pi(s, a) \end{aligned}$$

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# $\epsilon$ -Greedy Policy Improvement\*

## Theorem

For any  $\epsilon$ -greedy policy  $\pi$ , if the  $\epsilon$ -greedy policy  $\pi'$  with respect to  $Q_\pi$  is an improvement, then  $V_{\pi'}(s) \geq V_\pi(s)$

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

$$\begin{aligned} Q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) Q_\pi(s, a) \\ &= \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} Q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} Q_\pi(s, a) \\ &\geq \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} Q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1 - \epsilon} Q_\pi(s, a) \end{aligned}$$

# $\epsilon$ -Greedy Policy Improvement\*

## Theorem

For any  $\epsilon$ -greedy policy  $\pi$ , if the  $\epsilon$ -greedy policy  $\pi'$  with respect to  $Q_\pi$  is an improvement, then  $V_{\pi'}(s) \geq V_\pi(s)$

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

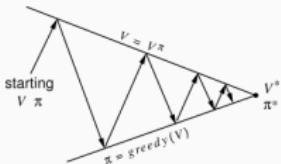
Policy Search  
Methods

$$\begin{aligned} Q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) Q_\pi(s, a) \\ &= \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} Q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} Q_\pi(s, a) \\ &\geq \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} Q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1 - \epsilon} Q_\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s, a) = V_\pi(s) \end{aligned}$$

Where  $m = |\mathcal{A}|$ .

Therefore from policy improvement theorem,  $V_{\pi'}(s) \geq V_\pi(s)$

# Monte-Carlo Policy Iteration



- Policy evaluation: Monte-Carlo policy evaluation.  $Q_\pi(s, a)$
- Policy improvement:  $\epsilon$ -greedy improvement

Is it necessary to wait until the end of execution of all trajectories ?

# Incremental Mean

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

The mean  $\mu_1, \mu_2, \dots$  of a sequence  $x_1, x_2, \dots$  can be computed incrementally,

$$\mu_k = \frac{1}{k} \sum_{j=1}^k x_j$$

$$= \frac{1}{k} \left( x_k + \sum_{j=1}^{k-1} x_j \right)$$

$$= \frac{1}{k} (x_k + (k - 1)\mu_{k-1})$$

$$= \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})$$

# Incremental Monte-Carlo Updates

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Update  $V(s)$  incrementally after episode  $S_1, A_1, R_2, \dots, S_T$
- For each state  $S_t$  with return  $G_t$ :

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

How do we set  $\alpha$  a.k.a **the learning rate** ?

# Monte-Carlo Control

Reinforcement Learning  
(SDMRL)

Sarah Keren

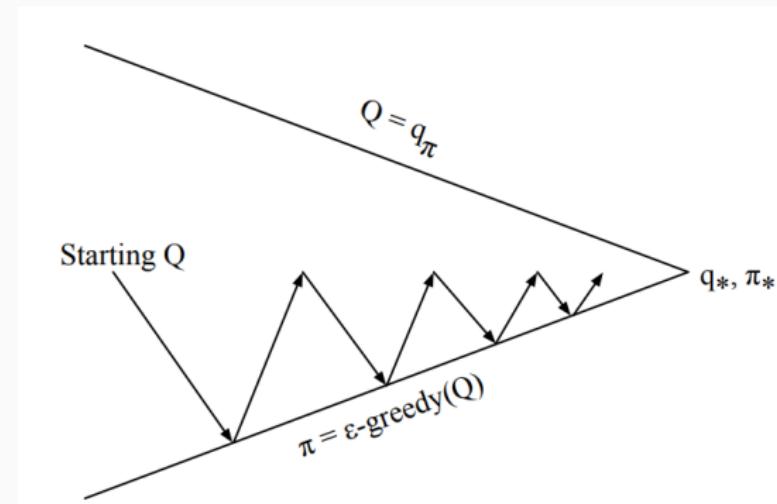
Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods



Every Iteration:

- Policy evaluation: Monte-Carlo policy evaluation.  $Q \approx q_\pi$
- Policy improvement:  $\epsilon$ -greedy improvement

# Monte-Carlo Control

Reinforcement Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

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

Algorithm parameter: small  $\varepsilon > 0$

Initialize:

$\pi \leftarrow$  an arbitrary  $\varepsilon$ -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  $\pi$ :  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

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

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

Append  $G$  to  $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow$  average( $Returns(S_t, A_t)$ )

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

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

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

Will this converge to an optimal policy?

# How do we make sure exploration eventually stops?

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# How do we make sure exploration eventually stops?

- Decay  $\epsilon$  Over Time: e.g., exponential decay  $\epsilon_t = \gamma \cdot \epsilon_{t-1}$  with  $\gamma \in (0, 1)$ .
- After a sufficient number of iterations (when  $Q(s, a)$  has stabilized), switch to a purely greedy policy.

## Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k \rightarrow \infty} N_k(s, a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k \rightarrow \infty} \pi_k(a|s) = \mathbf{1}(a = \arg \max_{a' \in \mathcal{A}} Q_k(s, a'))$$

For example,  $\epsilon$ -greedy is GLIE if  $\alpha$  reduces to zero at  $\epsilon_k = \frac{1}{k}$ .

Recap

Model Free RL:  
Monte CarloTemporal-  
Difference  
LearningComparing Monte  
Carlo and TD  
methodsPolicy Search  
Methods

# GLIE Monte-Carlo Control

Reinforcement Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

- Sample  $k$ th episode using  $\pi : \{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
- For each state  $S_t$  and action  $A_t$  in the episode,

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$$

- Improve policy based on new action-value function

$$\epsilon \leftarrow \frac{1}{k}$$

$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

## Theorem

GLIE Monte-Carlo control converges to the optimal action-value function,  $Q(s, a) \rightarrow Q^*(s, a)$

# Monte-Carlo Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Converge very slowly to correct utilities values (requires a lot of sequences)

$$V_\pi(s) = \mathbb{E}_\pi [G_t \mid S_t = s]$$

- Doesn't exploit Bellman constraints on policy values

$$V_\pi(s) = R(s) + \gamma \sum_{s'} \mathcal{P}(s' | s, \pi(s)) \cdot V_\pi(s')$$

- Can consider estimates that violate this property badly
- How can we incorporate such constraints ?

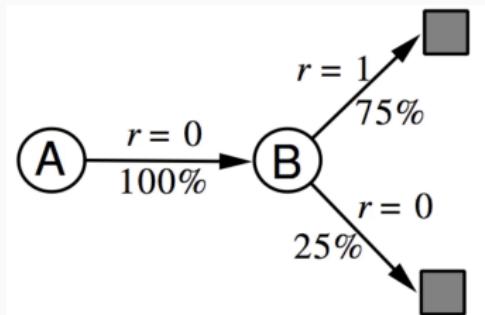
# Temporal-Difference Learning

---

# AB Example

Two states A, B; no discounting; 8 episodes of experience

- A, 0, B, 0
- B, 1
- B, 0



What is  $V(A)$ ,  $V(B)$ ?

# From Monte Carlo to Temporal-Difference (TD) Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

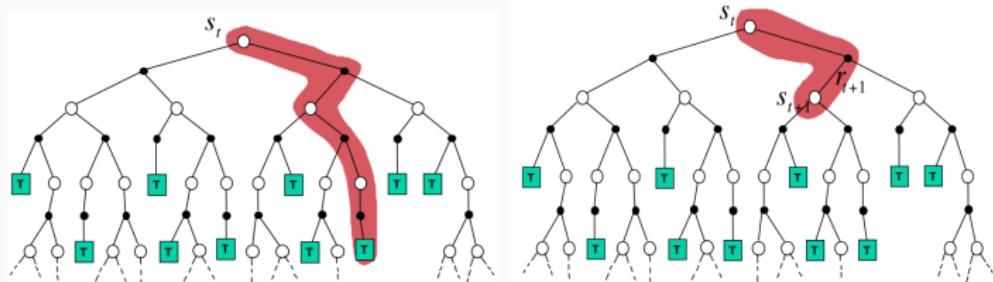
Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods



# Reminder: Bellman Formulas

Bellman equation (for a given deterministic policy):

Multiple variations, e.g.,

$$V_{\pi}(s) = R(s) + \gamma \sum_{s'} \mathcal{P}(s' | s, \pi(s)) \cdot V_{\pi}(s')$$

Bellman equation (for a given stochastic policy):

$$V_{\pi}(s) = \sum_a \pi(a | s) \left[ R(s, a) + \gamma \sum_{s'} \mathcal{P}(s' | s, a) V_{\pi}(s') \right]$$

Using  $Q$

$$Q_{\pi}(s, a) = R(s, a) + \gamma \sum_{s'} \mathcal{P}(s' | s, a) \sum_{a'} \pi(a' | s') Q_{\pi}(s', a')$$

Can we use these in RL?

# Bellman Optimality Equations

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

$$V^*(s) = \max_a \left[ R(s, a) + \gamma \sum_{s'} \mathcal{P}(s' | s, a) V^*(s') \right]$$

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} \mathcal{P}(s' | s, a) \max_{a'} Q^*(s', a')$$

$$V^*(s) = \max_a Q^*(s, a)$$

## Can we use these in RL?

Reminder: during the learning process we see (possibly partial) trajectories of the form:

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

# Temporal-Difference (TD) Learning: Evaluation

- For each transition from  $s$  to  $s'$ , we perform the following update

$$V_\pi(s) := V_\pi(s) + \alpha(R(s) + \gamma V_\pi(s') - V_\pi(s))$$

with  $\alpha$  as the learning rate

How does this move us closer to satisfying the Bellman constraint ?

$$V_\pi(s) = R(s) + \gamma \sum_{s'} \mathcal{P}(s'|s, \pi(s)) \cdot V_\pi(s')$$

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Temporal-Difference (TD) Learning: Evaluation

- For each transition from  $s$  to  $s'$ , we perform the following update

$$V_\pi(s) := V_\pi(s) + \alpha(R(s) + \gamma V_\pi(s') - V_\pi(s))$$

with  $\alpha$  as the learning rate

How does this move us closer to satisfying the Bellman constraint ?

$$V_\pi(s) = R(s) + \gamma \sum_{s'} \mathcal{P}(s'|s, \pi(s)) \cdot V_\pi(s')$$

- $R(s) + \gamma V_\pi(s')$  is a (noisy) sample of utility based on the next state.
- The update maintains a “mean” of (noisy) utility samples
- How do we guarantee convergence to the true values ?

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Temporal-Difference (TD) Learning: Evaluation

- For each transition from  $s$  to  $s'$ , we perform the following update

$$V_\pi(s) := V_\pi(s) + \alpha(R(s) + \gamma V_\pi(s') - V_\pi(s))$$

with  $\alpha$  as the learning rate

How does this move us closer to satisfying the Bellman constraint ?

$$V_\pi(s) = R(s) + \gamma \sum_{s'} \mathcal{P}(s'|s, \pi(s)) \cdot V_\pi(s')$$

- $R(s) + \gamma V_\pi(s')$  is a (noisy) sample of utility based on the next state.
- The update maintains a “mean” of (noisy) utility samples
- How do we guarantee convergence to the true values ?
  - If the learning rate decreases appropriately with the number of samples (e.g.  $\frac{1}{n}$ ), then the utility estimates will converge to true values (non-trivial).

# Temporal-Difference (TD) Learning: Control

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Simplest temporal-difference learning algorithm:  $TD(0)$ 
  - Update value  $V(s_t)$  toward estimated return  $R_{t+1} + \gamma V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

- If we are using  $Q$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

Which policy do we use here?

# Temporal-Difference (TD) Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Do local updates of utility/value function on a per-action basis.

# Temporal-Difference (TD) Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Do local updates of utility/value function on a per-action basis.

## What does TD learn from?

# Temporal-Difference (TD) Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Do local updates of utility/value function on a per-action basis.

**What does TD learn from?** TD methods learn directly from episodes of experience

**Model-free or model-based?**

# Temporal-Difference (TD) Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Do local updates of utility/value function on a per-action basis.

**What does TD learn from?** TD methods learn directly from episodes of experience

**Model-free or model-based?** TD is model-free: no learning of MDP transitions / rewards (doesn't try to estimate entire transition function).

**Is there a need to wait for the episode to end?**

# Temporal-Difference (TD) Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Do local updates of utility/value function on a per-action basis.

**What does TD learn from?** TD methods learn directly from episodes of experience

**Model-free or model-based?** TD is model-free: no learning of MDP transitions / rewards (doesn't try to estimate entire transition function).

**Is there a need to wait for the episode to end?** TD learns from incomplete episodes, by **bootstrapping** - updates a guess towards a guess.

# MC vs. TD

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Goal: learn and optimize value function online from experience
- Incremental every-visit Monte-Carlo
  - Update value  $V(s_t)$  toward actual return  $G_t$   
(typically,  $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$ )

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

- Simplest temporal-difference learning algorithm:  $TD(0)$ 
  - Update value  $V(s_t)$  toward estimated return  $R_{t+1} + \gamma V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

- $R_{t+1} + \gamma V(S_{t+1})$  is called the **TD target**
- $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$  is called the **TD error**

# Going Home Example

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

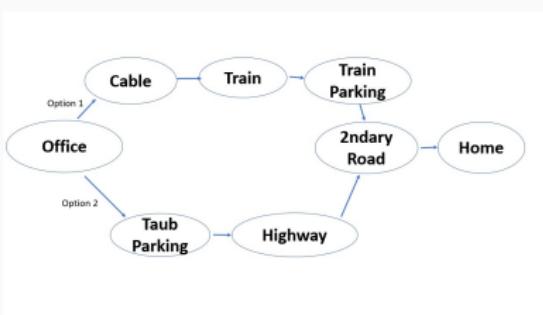
Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods



# Going Home Example

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

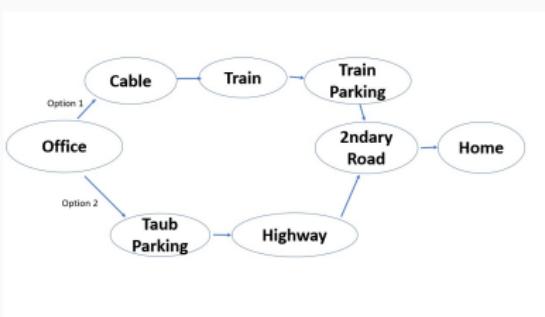
Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods



- $\tau_1 = \text{office}, 20, \text{cable}, 15, \text{train}, 30, \text{train-park}, 5, \text{2ndary}, 15, \text{home}$
- $\tau_2 = \text{office}, 20, \text{cable}, 15, \text{train}, 60, \text{train-park}, 5, \text{2ndary}, 15, \text{home}$
- $\tau_3 = \text{office}, 5, \text{taub-park}, 10, \text{highway}, 60, \text{2ndary}, 15, \text{home}$
- $\tau_4 = \text{office}, 5, \text{taub-park}, 10, \text{highway}, 60, \text{2ndary}, 35, \text{home}$
- $\tau_5 = \text{office}, 5, \text{taub-park}, 10, \text{highway}, 60$

# Going Home Example

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

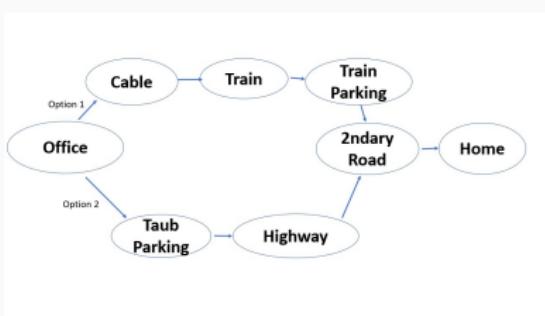
Policy Search  
Methods

$$V_{\pi}(S_t) \leftarrow V_{\pi}(S_t) + \alpha(G_t - V(S_t))$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

$$V_{\pi}(S_t) \leftarrow V_{\pi}(S_t) + \alpha(R(S_t) + \gamma V_{\pi}(S_{t+1}) - V_{\pi}(S))$$

$$\gamma = 1 \quad \alpha = 0.5$$



- $\tau_1$  = office, 20, cable, 15, train, 30, train-park, 5, 2ndary, 15, home
- $\tau_2$  = office, 20, cable, 15, train, 60, train-park, 5, 2ndary, 15, home
- $\tau_3$  = office, 5, taub-park, 10, highway, 60, 2ndary, 15, home
- $\tau_4$  = office, 5, taub-park, 10, highway, 60, 2ndary, 35, home
- $\tau_5$  = office, 5, taub-park, 10, highway, 60

# Going Home Example

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

$$V_{\pi}(S_t) \leftarrow V_{\pi}(S_t) + \alpha(G_t - V(S_t))$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

$$V_{\pi}(S_t) \leftarrow V_{\pi}(S_t) + \alpha(R(S_t) + \gamma V_{\pi}(S_{t+1}) - V_{\pi}(S))$$

$$\gamma = 1 \quad \alpha = 0.5$$

State	0	MC		TD	
		1	2	1	2
Exit office	?	?	?	?	?
Enter Cable	?	?	?	?	?
Enter Taub Park	?	?	?	?	?
Enter Train	?	?	?	?	?
Enter Train Park	?	?	?	?	?
Enter Highway	?	?	?	?	?
Enter 2nd Road	?	?	?	?	?
Enter Home	?	?	?	?	?

- $\tau_1$  = office, 20, cable, 15, train, 30, train-park, 5, 2ndary, 15, home
- $\tau_2$  = office, 20, cable, 15, train, 60, train-park, 5, 2ndary, 15, home
- $\tau_3$  = office, 5, taub-park, 10, highway, 60, 2ndary, 15, home
- $\tau_4$  = office, 5, taub-park, 10, highway, 60, 2ndary, 35, home
- $\tau_5$  = office, 5, taub-park, 10, highway, 60

# From MC to TD Control

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)

# From MC to TD Control

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Works with incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to  $Q(S, A)$
  - Use  $\epsilon$ -greedy policy improvement
  - Update every time-step.

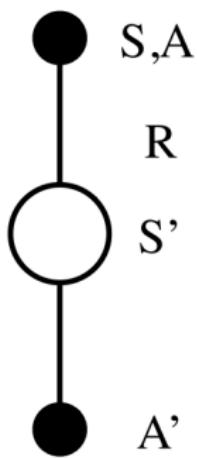
# TD methods: SARSA

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Updating Action-Value Functions with



Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$$

# On-Policy Control With SARSA

Reinforcement Learning  
(SDMRL)

Sarah Keren

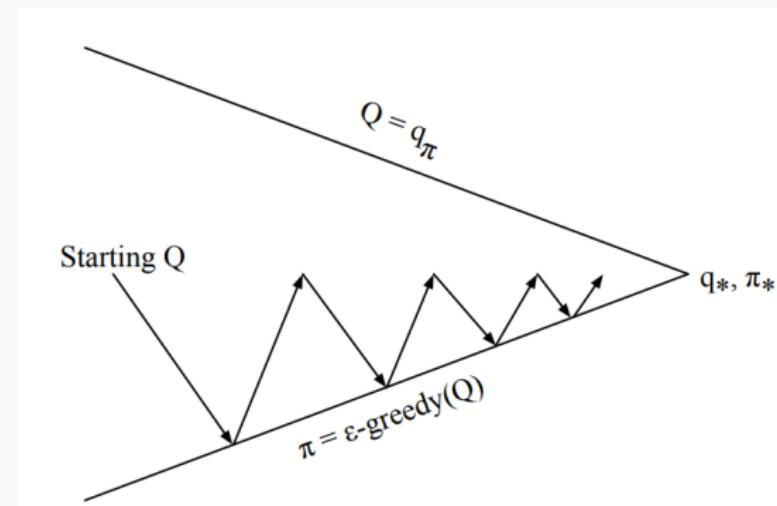
Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods



Every time-step:

- Policy evaluation: Sarsa  $Q \approx q_\pi$
- Policy improvement:  $\epsilon$ -greedy improvement

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$   
Repeat (for each episode):

    Initialize  $S$

    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\varepsilon$ -greedy)

    Repeat (for each step of episode):

        Take action  $A$ , observe  $R, S'$

        Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\varepsilon$ -greedy)

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$

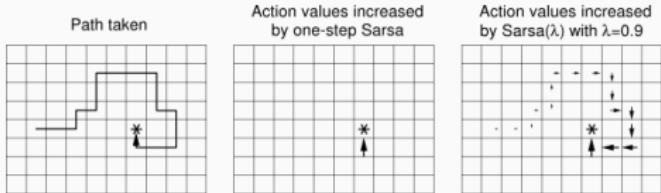
$S \leftarrow S'; A \leftarrow A'$ ;

    until  $S$  is terminal

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$$

# Additional topics:

- N-Step Sarsa
- Sarsa( $\lambda$ )
- Forward and backward view Sarsa
- Convergence proof



Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Off-Policy Learning

- Evaluate target policy  $\pi(a|s)$  to compute  $\mathcal{V}_\pi(s)$  or  $Q_\pi(s, a)$  while following behaviour policy  $\mu(a|s)$

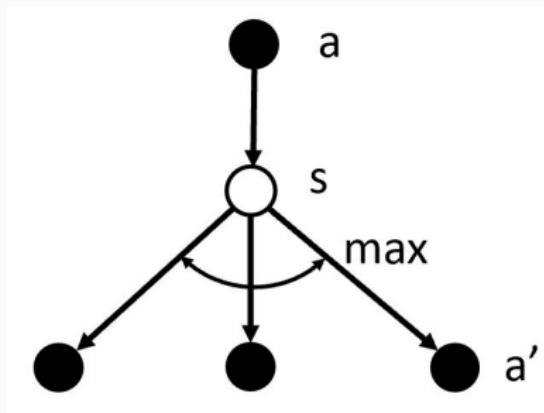
$$\{S_1, A_1, R_2, \dots, S_T\} \sim \mu$$

- Why is this important?
  - Learn from observing humans or other agents
  - Re-use experience generated from old policies  $\pi_1, \pi_2, \dots, \pi_{t1}$
  - Learn about optimal policy while following exploratory policy
  - Learn about multiple policies while following one policy

# Q-Learning

- We now consider off-policy learning of action-values  $Q(s, a)$
- Next action is chosen using the behaviour policy  $A_{t+1} \sim \mu(\cdot | S_t)$  but we update the target policy by considering the possible actions that could be applied to the next state and choose one with maximal value.

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$$



Reinforcement Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

# Off-Policy Control with Q-Learning

Reinforcement Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

- In Q-learning we allow both behaviour and target policies to improve
- The target policy  $\pi$  is greedy w.r.t.  $Q(s, a)$

$$\pi(s) = \arg \max_{a'} Q(s, a')$$

- The behaviour policy  $\mu$  is (e.g.)  **$\epsilon$ -greedy** w.r.t.  $Q(s, a)$
- The Q-learning target then simplifies:

$$R + \gamma Q(S', A')$$

$$= R + \gamma Q(S', \arg \max_a Q(S', a))$$

$$= R + \max_a \gamma Q(S', a)$$

# Q-Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$   
Repeat (for each episode):

    Initialize  $S$

    Repeat (for each step of episode):

        Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\varepsilon$ -greedy)

        Take action  $A$ , observe  $R, S'$

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$ ;

    until  $S$  is terminal

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$$

# Q-Learning

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

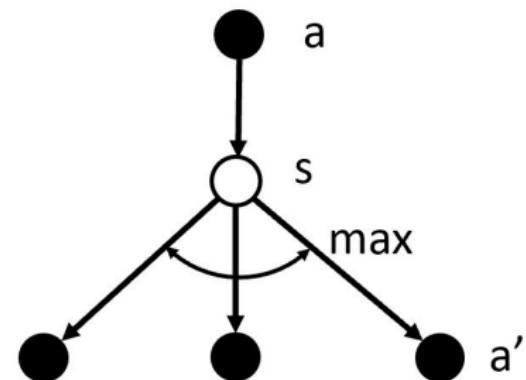
Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$$



## Theorem

Q-learning control converges to the optimal action-value function,  $Q(s, a) \rightarrow Q^*(s, a)$

# SARSA vs. Q-Learning

SARSA:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Q-Learning:

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$$

# SARSA vs. Q-Learning

Reinforcement Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

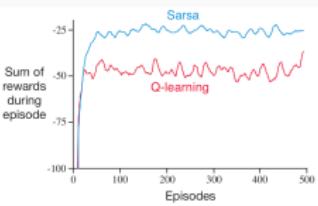
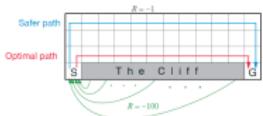
Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$   
Repeat (for each episode):  
    Initialize  $S$   
    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)  
    Repeat (for each step of episode):  
        Take action  $A$ , observe  $R, S'$   
        Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)  
         $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$   
         $S \leftarrow S'; A \leftarrow A'$ ;  
    until  $S$  is terminal

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$   
Repeat (for each episode):  
    Initialize  $S$   
    Repeat (for each step of episode):  
        Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)  
        Take action  $A$ , observe  $R, S'$   
         $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$   
         $S \leftarrow S'$ ;  
    until  $S$  is terminal



## Comparing Monte Carlo and TD methods

---

# MC vs. TD

- TD can learn before knowing the final outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn without the final outcome
  - TD can learn from incomplete sequences
  - MC can only learn from complete sequences
  - TD works in continuing (non-terminating) environments
  - MC only works for episodic (terminating) environments

# Bias/Variance Trade-Off

- Return  $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$  is an **unbiased** estimate of  $V_\pi(S_t)$
- True TD target  $R_{t+1} + \gamma v_\pi(S_{t+1})$  is **unbiased estimate** of  $\pi(S_t)$ .
- TD target  $R_{t+1} + \gamma V(S_{t+1})$  is a **biased** estimate  $\pi(S_t)$ . **why?**
- TD target has much lower variance than the return - **why?**

# Bias/Variance Trade-Off

- Return  $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$  is an **unbiased** estimate of  $V_\pi(S_t)$
- True TD target  $R_{t+1} + \gamma v_\pi(S_{t+1})$  is **unbiased estimate** of  $\pi(S_t)$ .
- TD target  $R_{t+1} + \gamma V(S_{t+1})$  is a **biased** estimate  $\pi(S_t)$ . **why?**
- TD target has much lower variance than the return - **why?**
  - Return depends on many random actions, transitions, rewards
  - TD target depends on one random action, transition, reward

# MC vs. TD (continued)

- MC has high variance, zero bias
  - Good convergence properties
  - (even with function approximation)
  - Not very sensitive to initial value
  - Very simple to understand and use
- TD has low variance, some bias
  - Usually more efficient than MC
  - $\text{TD}(0)$  converges to  $\pi(s)$
  - (but not always with function approximation)
  - More sensitive to initial value

# Random Walk Example

Reinforcement Learning  
(SDMRL)

Sarah Keren

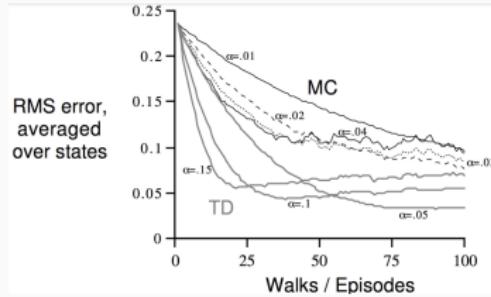
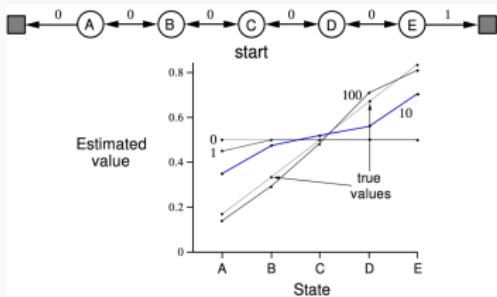
Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods



# MC vs. TD (continued)

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- MC converges to solution with minimum mean-squared error

- Best fit to the observed returns

$$\sum_{k=1}^K \sum_{t=1}^{T_k} (G_t^k - \mathcal{V}(s_t^k))^2$$

- $TD(0)$  converges to solution of max likelihood Markov model

- Solution to the MDP  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$  that best fits the data

$$\mathcal{P}_{s,s'}^a = \frac{1}{N(s,a)} \sum_{k=1}^K \sum_{t=1}^{T_k} 1(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\mathcal{R}_s^a = \frac{1}{N(s,a)} \sum_{k=1}^K \sum_{t=1}^{T_k} 1(s_t^k, a_t^k = s, a) r_t^k$$

# MC vs. TD (continued)

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- TD exploits Markov property
  - Usually more efficient in Markov environments
- MC does not exploit Markov property
  - Usually more effective in non-Markov environments

# Monte-Carlo vs. Temporal-Difference vs. Dynamic Programming Backup

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

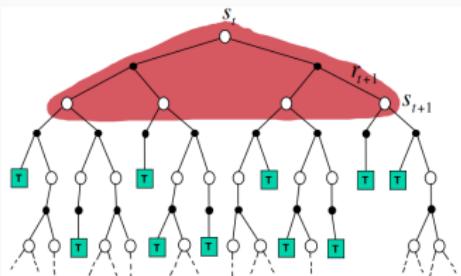
Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

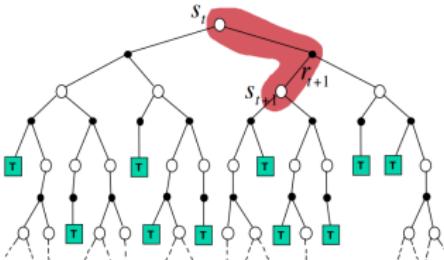
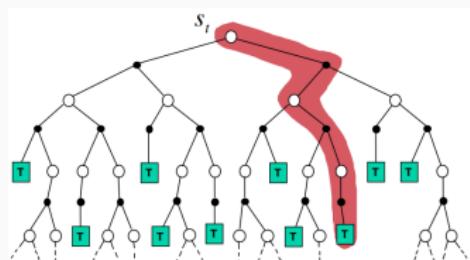
Which is which ?

$$\mathcal{V}(s_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma \mathcal{V}(S_{t+1})]$$



$$\mathcal{V}(S_t) \leftarrow \mathcal{V}(S_t) + \alpha(G_t \mathcal{V}(S_t))$$

$$\mathcal{V}(s_t) \leftarrow \mathcal{V}(s_t) + \alpha(R_{t+1} + \gamma \mathcal{V}(S_{t+1}) - \mathcal{V})$$



# Bootstrapping and Sampling

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Bootstrapping:

- MC
- DP
- TD

- Sampling:

- MC
- DP
- TD

# Bootstrapping and Sampling

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- **Bootstrapping:**
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- **Sampling:**
  - MC samples
  - DP does not sample
  - TD samples

# Unified View of Reinforcement Learning

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

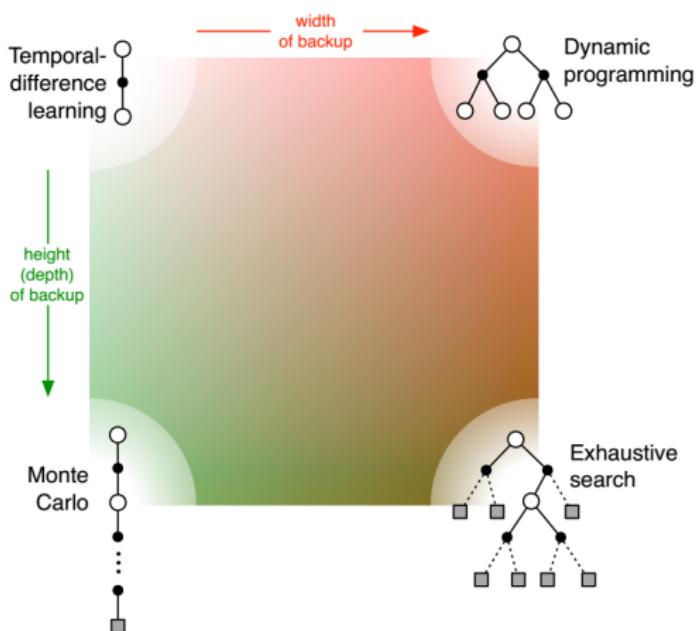
Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

## Unified View



# Extensions and Additional Topics (which we won't cover)

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- n-Step Prediction
- Averaging n-Step Returns ( $TD(\lambda)$ )
- Eligibility Traces and credit assignment:
  - Frequency heuristic: assign credit to most frequent states
  - Recency heuristic: assign credit to most recent states
- Forward vs. backward view

## Policy Search Methods

---

# Policy Search

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Key Idea: Keep Twiddling the policy as long as its performance improves, then stop.
- In some ways, policy search is the simplest of all methods

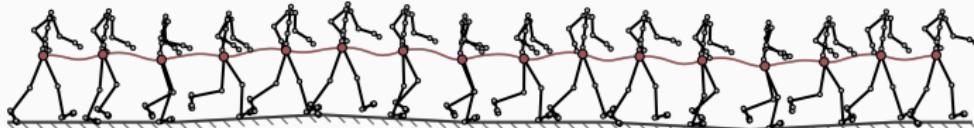
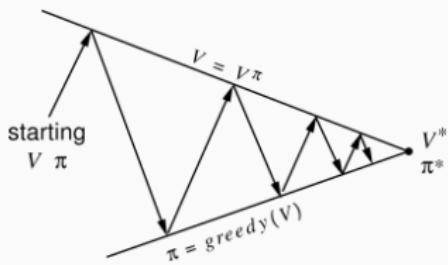


Image from Guided Policy Search by Levine and Koltun, 2013

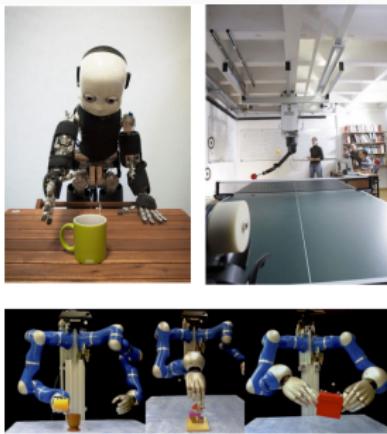
How does this fit within our general structure ?



# Motivation for Policy-based Reinforcement Learning

## Challenges:

- Dimensionality:
  - High-dimensional continuous state and action space
  - Huge variety of tasks
- Real world environments:
  - High-costs of generating data
  - Noisy measurements
- Exploration:
  - Do not damage the robot
  - Need to generate smooth trajectories



From

<https://icml.cc/2015/tutorials/PolicySearch.pdf>

Reinforcement Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-Difference Learning

Comparing Monte Carlo and TD methods

Policy Search Methods

Sarah Keren

# Policy Search

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

Which policy search methods have we already seen ?

Which policy search methods have we already seen ?

- **Policy Iteration:** for known MDPs, we start with a fixed policy and iteratively improve until we reach convergence.
- **Hill Climbing:** maximize (or minimize) a target function  $f(\mathbf{x})$ . At each iteration, adjust a single element in  $\mathbf{x}$  and determine whether the change improves the value of  $f(\mathbf{x})$ .

# Reminder: RL Approaches

Reinforcement Learning  
(SDMRL)

Sarah Keren

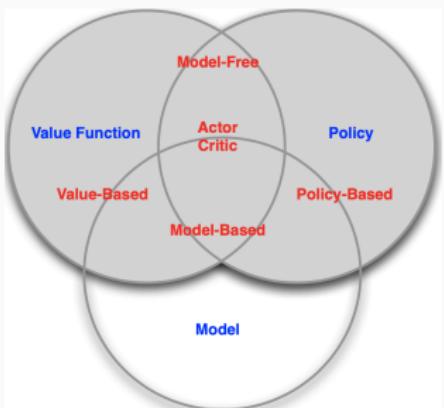
Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods



# Reminder: Value-based Reinforcement Learning:

- Estimate value function: e.g.

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma Q(s', a') - Q(s, a))$$

- Global estimate for all reachable states
- Hard to scale to high-dimensional space
- Approximations might compromise policy quality.

- Estimate global policy: e.g.  $\pi'(s) = \arg \max_{a \in A} Q(s, a)$

- Greedy policy update for all states
- Policy update might get unstable

- Explore the whole state space: e.g.  $\pi(a|s) = \frac{\exp(Q(s,a))}{\sum_a' \exp(Q(s,a'))}$
- Uncorrelated exploration in each step
- (Might damage a robot)

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Policy Search

Reinforcement Learning  
(SDMRL)

Sarah Keren

Recap

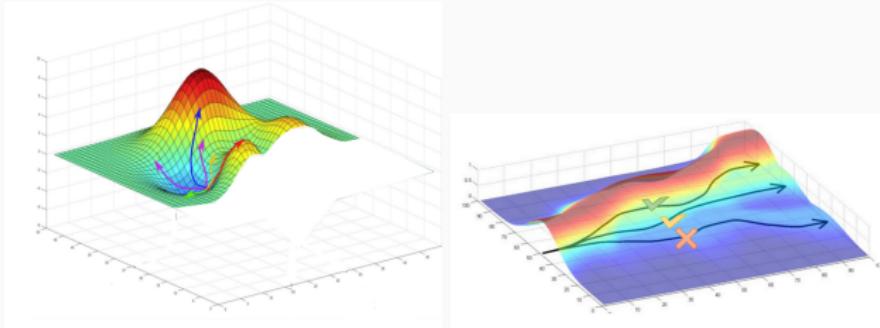
Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Policy search methods directly optimize policy parameters without learning a value function
- Key advantage: Can handle continuous action spaces naturally
- Different from value-based methods like Q-learning



# Policy Representation

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Policies can be represented as:
  - Neural networks:  $\pi_\theta(a|s)$
  - Linear functions:  $\pi_\theta(a|s) = \sigma(\theta^T \phi(s))$
  - Gaussian policies:  $\pi_\theta(a|s) = \mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$

# Parameterized Policies

- A policy  $\pi$  is a function that maps states to actions (or state-action pairs to probabilities).
- We approximate the value or action-value function using parameters  $\Theta$ ,

$$\mathcal{V}_\theta(s) \approx \mathcal{V}_\pi(s)$$

$$Q_\theta(s, a) \approx Q_\pi(s, a)$$

- A policy was generated directly from the value function e.g. using  $\epsilon$ -greedy.
- **Policy search (gradient)** methods directly parametrise the policy  $\pi_\theta(s, a) = \mathcal{P}[a|s, \theta]$
- Both model-free and model-based versions.

Reinforcement  
Learning

(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

# Policy Search Methods<sup>1</sup>

- Use parametrized policy  $a \sim \pi(a|s; \theta)$ ,  $\theta$  - parameter vector.
  - Compact parametrizations for high-dimensional spaces
  - Encode prior knowledge
- Locally optimal solutions e.g.,  $\theta_{new} = \theta_{old} + \alpha \frac{\partial J_\theta}{\partial \theta}$ , where  $J$  is the loss function.
  - Safe policy updates
  - No global value function estimation
- Correlated local exploration  
e.g.:  $\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta)$ 
  - Explore in parameter space
  - Generates smooth trajectories

<https://icml.cc/2015/tutorials/PolicySearch.pdf>

<sup>1</sup>see Deisenroth, Neumann and Peters for A Survey of Policy Search for Robotics, FNT 2013

**REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for  $\pi_*$** 

Input: a differentiable policy parameterization  $\pi(a|s, \theta)$

Algorithm parameter: step size  $\alpha > 0$

Initialize policy parameter  $\theta \in \mathbb{R}^{d'}$  (e.g., to  $\mathbf{0}$ )

Loop forever (for each episode):

Generate an episode  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$ , following  $\pi(\cdot| \cdot, \theta)$

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

$$\begin{aligned} G &\leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \\ \theta &\leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t | S_t, \theta) \end{aligned} \tag{G_t}$$

Figure 2: Taken from Sutton and Barto

# Recap and what next

Reinforcement  
Learning  
(SDMRL)

Sarah Keren

Recap

Model Free RL:  
Monte Carlo

Temporal-  
Difference  
Learning

Comparing Monte  
Carlo and TD  
methods

Policy Search  
Methods

- Spectrum of approaches to model-free RL
- Next: Dealing with large states space and model-based RL

