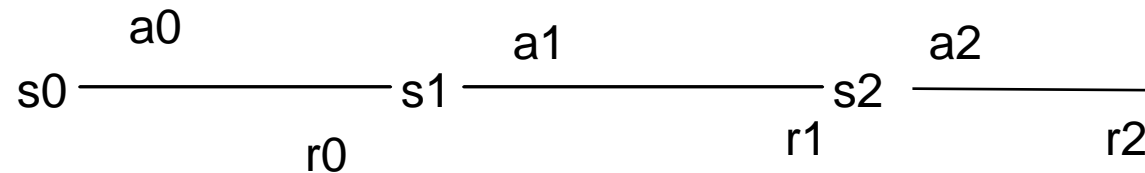
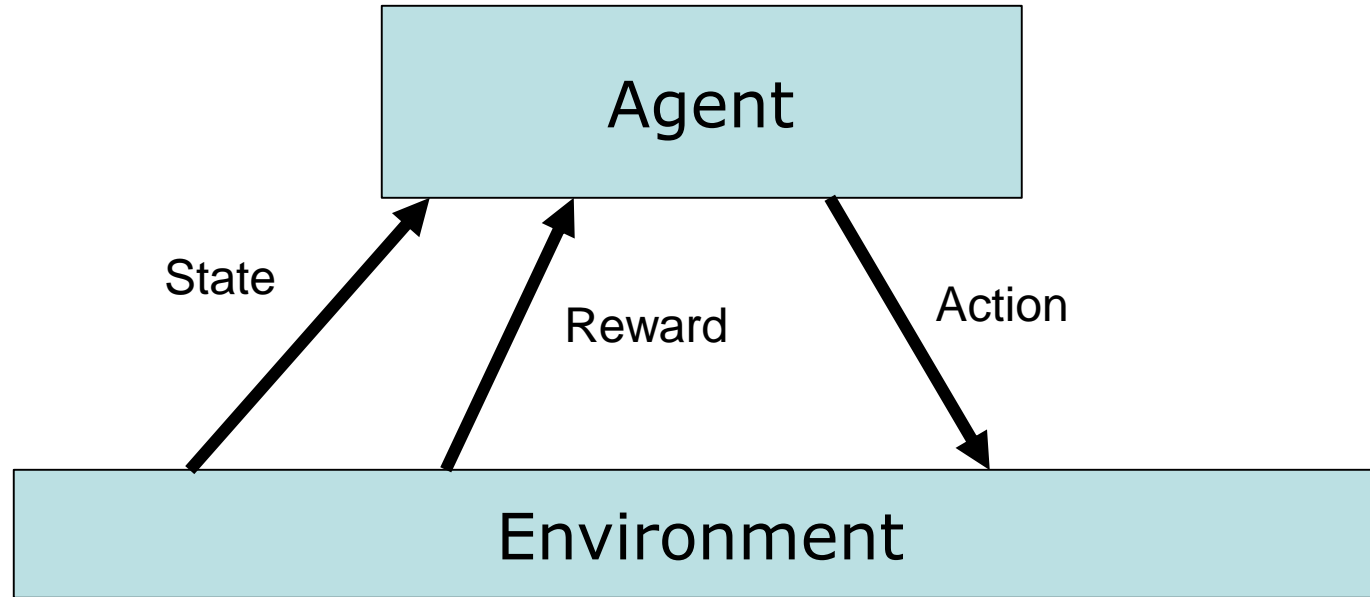


Introduction to Artificial Intelligence

CS 486/686

University of Waterloo

Introduction



Goal: Learn to choose actions that maximize $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$, where $0 < \gamma < 1$

Reinforcement Learning Characteristics

Delayed Reward

Credit assignment problem

Exploration and exploitation

Possibility that a state is only partially observable

Life-long learning

RL Model

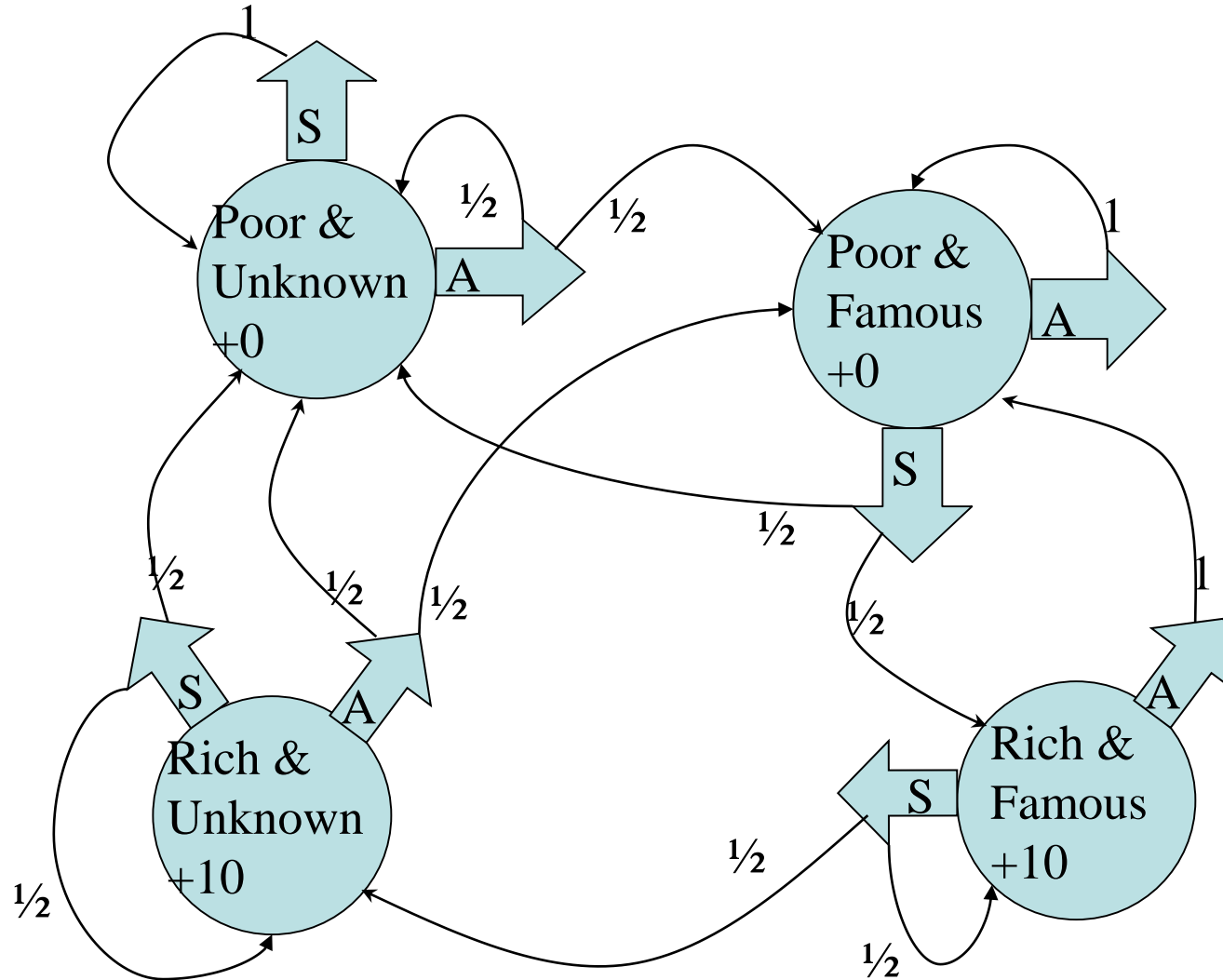
Set of States, S

Set of Actions, A

Set of reward signals, R

Rewards might be delayed

Markov Decision Process



MDPs and RL

If we were given the MDP then we could compute the optimal policy

$$\pi^*(S_i) = \arg \max_a \left[R_i + \gamma \sum_{j=1}^n P(S_j|S_i, a) V^*(S_j) \right]$$

$$V^*(S_i) = R_i + \gamma \sum_{j=1}^n P(S_j|S_i, \pi^*(S_i)) V^*(S_j)$$

In RL we are not given the model (rewards/transition probabilities)

Prediction problem: learn V^* 's or V^π

Control problem: learn π^*

All RL methods are driven by values

- Recall the discounted sum of future rewards:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots$$

- Value of a state, given a policy π

$$V^\pi(s) = \mathbb{E}\{G_t | S_t = s, A_{t:\infty} \sim \pi\}$$

- Value of a state-action pair, given a policy π

$$Q^\pi(s, a) = \mathbb{E}\{G_t | S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\}$$

All RL methods are driven by values

Optimal value of a state

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

Optimal value of a state-action pair

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

Optimal policy: π is an optimal policy if and only if

$$\pi^*(s, a) > 0 \text{ only where } Q^*(s, a) = \max_b Q^*(s, b) \forall s \in \mathcal{S}$$

That is, π^* is optimal iff it is greedy with respect to Q^*

4 Value Functions

	State Values	Action Values
Prediction Problem	V^π	Q^π
Control Problem	V^*	Q^*

These are theoretical objects, and are distinct from their estimates V^t and Q^t

Bellman Optimality Equations

$$\begin{aligned} V^*(s) &= \max_a Q^{\pi^*}(s, a) \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma V^*(s_{t+1} | S_t = s, A_t = a)] \\ &= \max_a \sum_{s', r'} p(s', r | s, a) [r + \gamma V^*(s')] \end{aligned}$$

$$\begin{aligned} Q^*(s, a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') | S_t = s, A_t = a] \\ &= \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} Q^*(s', a')] \end{aligned}$$

Another way of distinguishing RL methods

- **Model-Based**

- Learn the model of the environment (i.e. when done, we know the underlying MDP)

- **Model-Free**

- Never explicitly learn the model (i.e. we never track probability with which we transition between states)

RL and Prediction

- Let's consider a simple problem

3	r	r	r	+1
2	u		u	-1
1	u	l	l	l
	1	2	3	4

$$\gamma = 1$$

$r_i = -0.04$ for non-terminal states

We do not know the transition probabilities

$(1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (2,3) \rightarrow (3,3) \rightarrow (4,3)_{+1}$

$(1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (2,3) \rightarrow (3,3) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (4,3)_{+1}$

$(1,1) \rightarrow (2,1) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (4,2)_{-1}$

What is the value, $V^n(s)$ of being in state s ?

RL and Prediction: Value Estimation

3	r	r	r	+1
2	u		u	-1
1	u	l	l	l
	1	2	3	4

$$\gamma = 1$$

$r_i = -0.04$ for non-terminal states

$(1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (2,3) \rightarrow (3,3) \rightarrow (4,3)_{+1}$
 $(1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (2,3) \rightarrow (3,3) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (4,3)_{+1}$
 $(1,1) \rightarrow (2,1) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (4,2)_{-1}$

What is the value, $V^*(s)$ of being in state s ?


$$V^\pi(S) = E[\sum_{t=0}^{\infty} \gamma^t R(S_t)]$$

Asynchronous Dynamic Programming

$\gamma = 1$

3	r	r	r	+1
2	u		u	-1
1	u	l	l	l
	1	2	3	4

$r_i = -0.04$ for non-terminal states

$$V^\pi(s_i) = r(s_i) + \gamma \sum_j P_{ij}^\pi V^\pi(s_j)$$


$(1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (2,3) \rightarrow (3,3) \rightarrow (4,3)_{+1}$
 $(1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (2,3) \rightarrow (3,3) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (4,3)_{+1}$
 $(1,1) \rightarrow (2,1) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (4,2)_{-1}$

$P_{(1,3)(2,3)}^r = 2/3$
 $P_{(1,3)(1,2)}^r = 1/3$

} Use this information in the Bellman equation

Prediction: Temporal Difference (TD)

- TD is considered to be a bootstrapping and sampling method
- Bootstrapping: update involves an estimate of the value function
 - TD and dynamic programming bootstrap
 - Direct utility estimation (a variant of a Monte Carlo method) did not
- Sampling: update does not involve an expected value
 - TD and direct utility estimation samples
 - Dynamic programming does not sample

Prediction: The Simplest TD Method TD(0)

- The simplest temporal-difference method TD(0):

$$V(S_t) \leftarrow V(S_t) + \alpha \left[\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{target}} - V(S_t) \right]$$

target: an estimate of the return

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated

Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$)

Repeat (for each episode):

Initialize S

Repeat (for each step of episode):

$A \leftarrow$ action given by π for S

Take action A , observe R, S'

$V(S) \leftarrow V(S) + \alpha [R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

until S is terminal

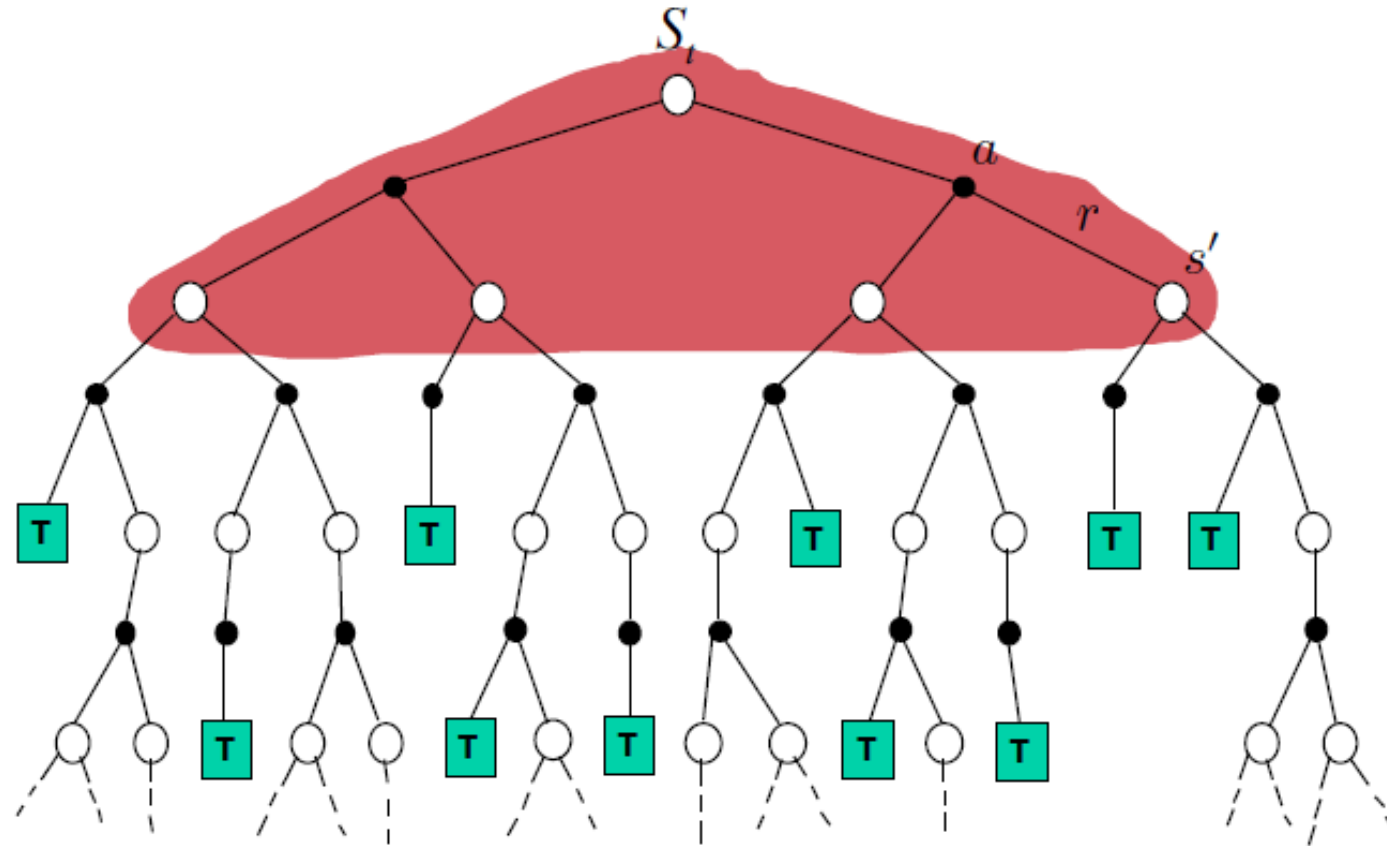
Agent program

Environment program

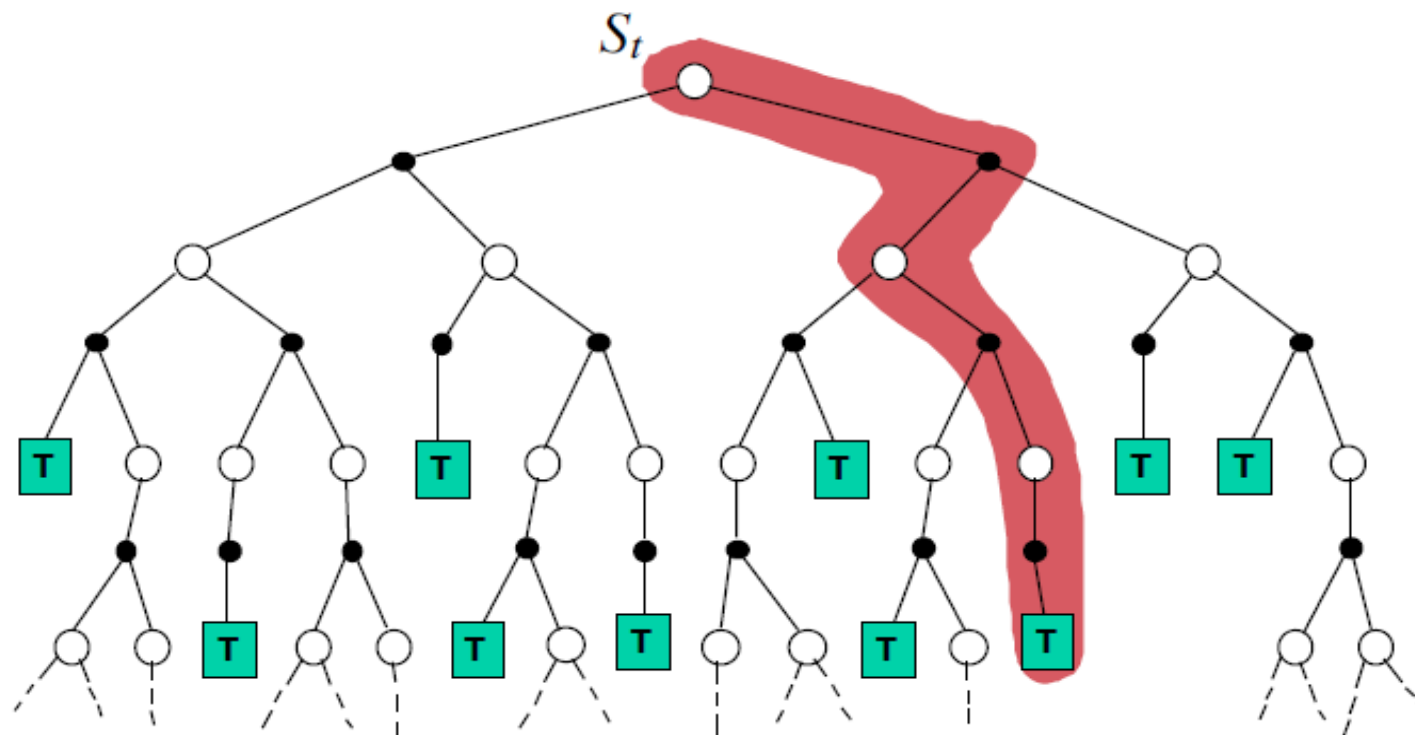
Experiment program

Dynamic Programming

$$V(S_t) \leftarrow E_{\pi} [R_{t+1} + \gamma V(S_{t+1})] = \sum_a \pi(a|S_t) \sum_{s',r} p(s',r|S_t,a) [r + \gamma V(s')]$$

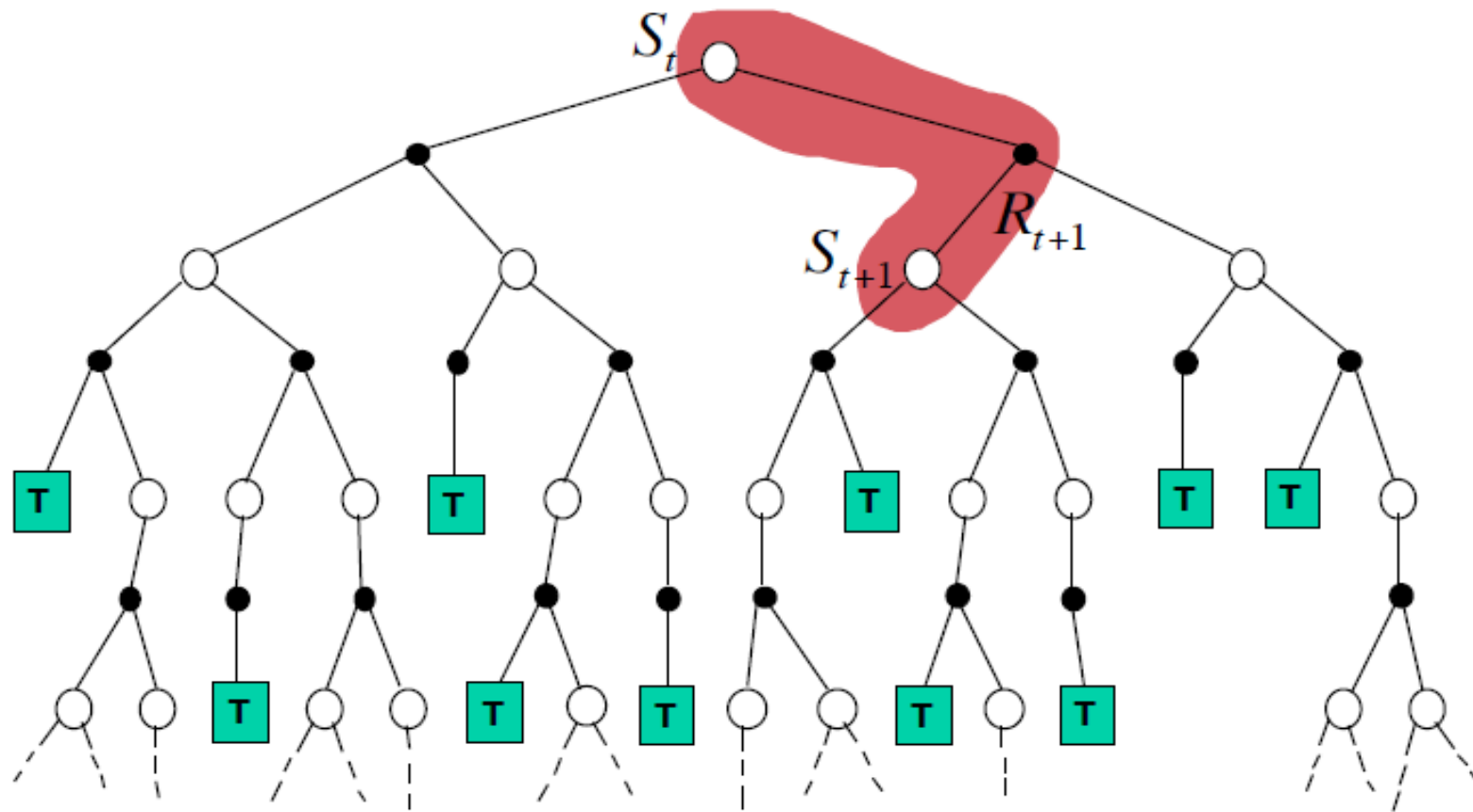


Direct Utility Estimation (and other MC variants)



TD(0)

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



Example: Driving Home (Sutton and Barto)

Driving home:

Each day you drive home

Your goal is to try and predict how long your commute will take at particular stages

When you leave your office you note the time, day, and other relevant information

Policy Evaluation or prediction task

Example: Driving Home

<i>State</i>	<i>Elapsed Time (minutes)</i>	<i>Predicted Time to Go</i>	<i>Predicted Total Time</i>
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Example: Driving Home

Rewards = 1 per step

Discount = 1

G_t = time to go from state S_t

$V(S_t)$ = expected time to get home from S_t

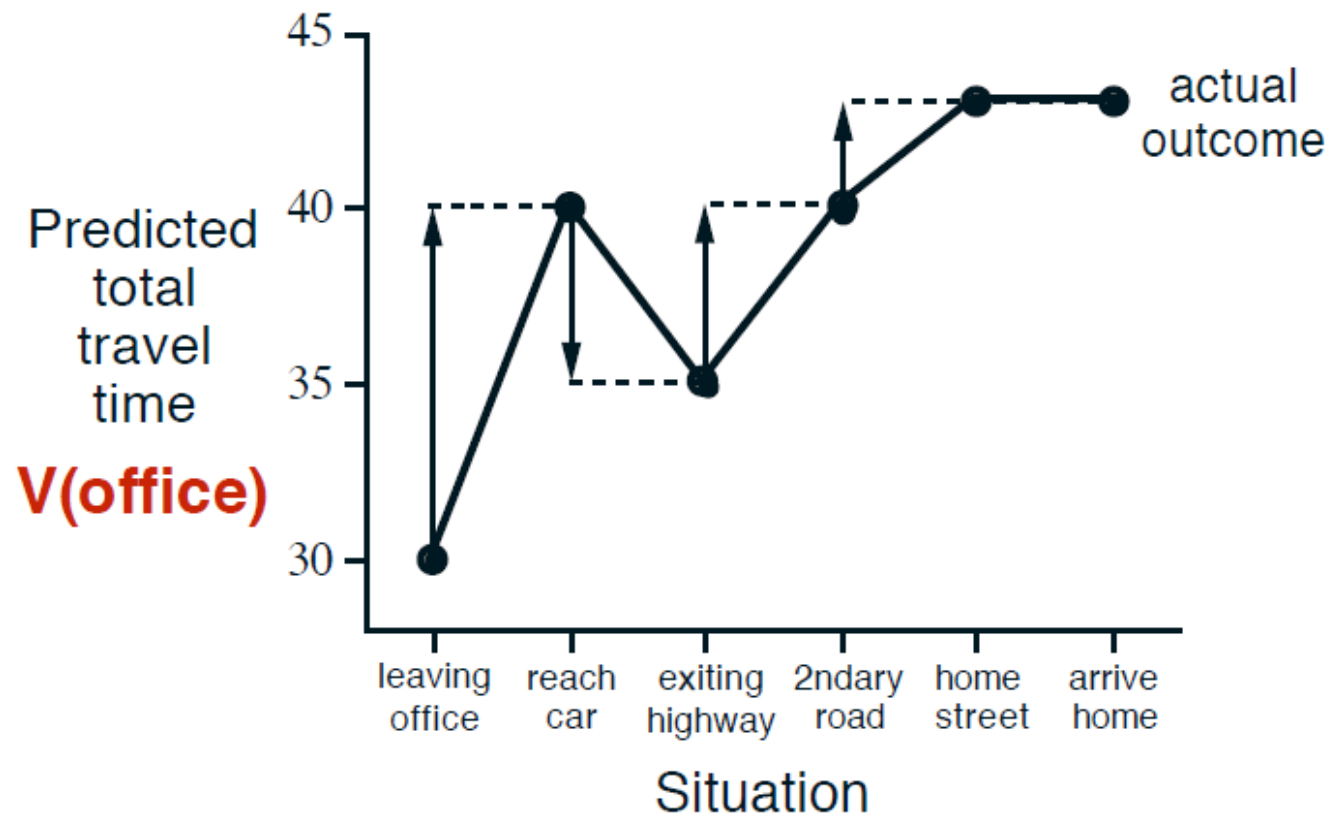
Goal: update the prediction of total time leaving from office, while driving home

Driving Home

<i>State</i>	<i>Elapsed Time</i> <i>(minutes)</i>	R	V(s) <i>Predicted</i> <i>Time to Go</i>	V(office) <i>Predicted</i> <i>Total Time</i>
leaving office, friday at 6	0	5	30	30
reach car, raining	5	15	35	40
exiting highway	20	10	15	35
2ndary road, behind truck	30	10	10	40
entering home street	40	3	3	43
arrive home	43		0	43

- Task: update the value function as we go, based on observed elapsed time—Reward column

Changes Recommended by TD(0) (alpha=1)



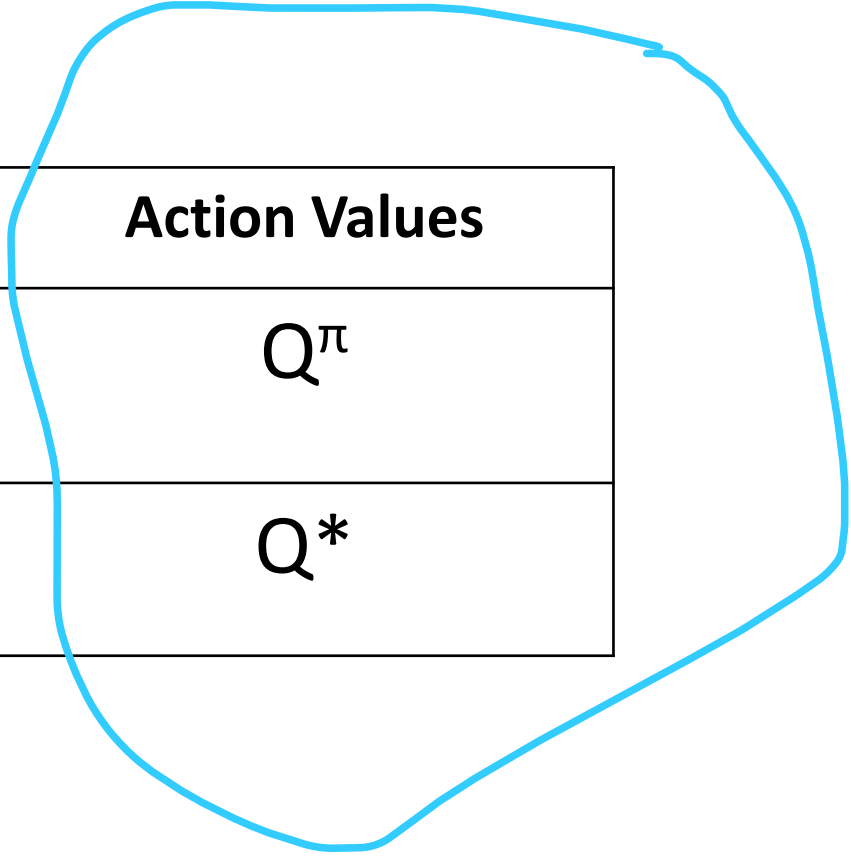
TD(λ)

- **Idea:** You can update from the whole training sequence not just a single transition

$$V^\pi(s_i) \rightarrow V^\pi(s_i) + \alpha \sum_{m=i}^{\infty} \lambda^{m-i} [r(s_m) + \gamma V^\pi(s_{m+1}) - V^\pi(s_m)]$$

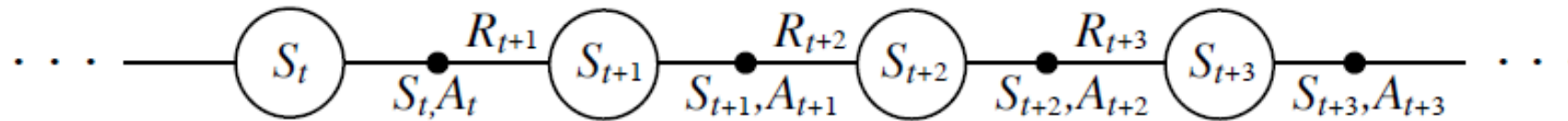
Learning Action-Value Functions

	State Values	Action Values
Prediction Problem	V^π	Q^π
Control Problem	V^*	Q^*



Learning Action-Values

Estimate Q^π from current policy π



After every transition from a non-terminal state, S_t , do

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

If S_{t+1} is terminal, then $Q(S_{t+1}, A_{t+1})=0$

From Learning Action-Values to Control

We can take action-value prediction problem and change it to a control problem by always updating the policy to be greedy with respect to the current estimates

Different types of policies:

Behavioral Policy: used to generate actions and gather data

Learning Policy: target policy to learn

On-Policy Learning: Behavior = Learning

Off-Policy Learning: Behavior \neq Learning

Sarsa: On-Policy TD Control

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

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., ϵ -greedy)

Repeat (for each step of episode):

Take action A , observe R, S'

Choose A' from S' using policy derived from Q (e.g., ϵ -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-Learning: Off-Policy TD Control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

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., ϵ -greedy)

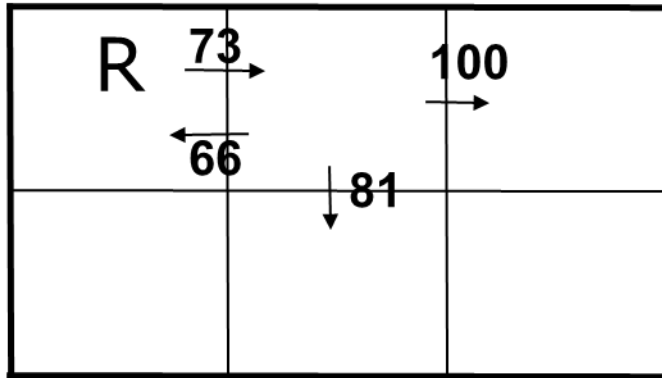
Take action A , observe R, S'

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

$S \leftarrow S'$;

until S is terminal

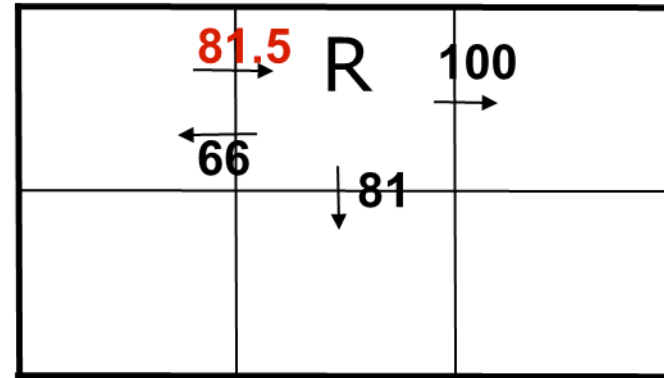
Example: Q-Learning



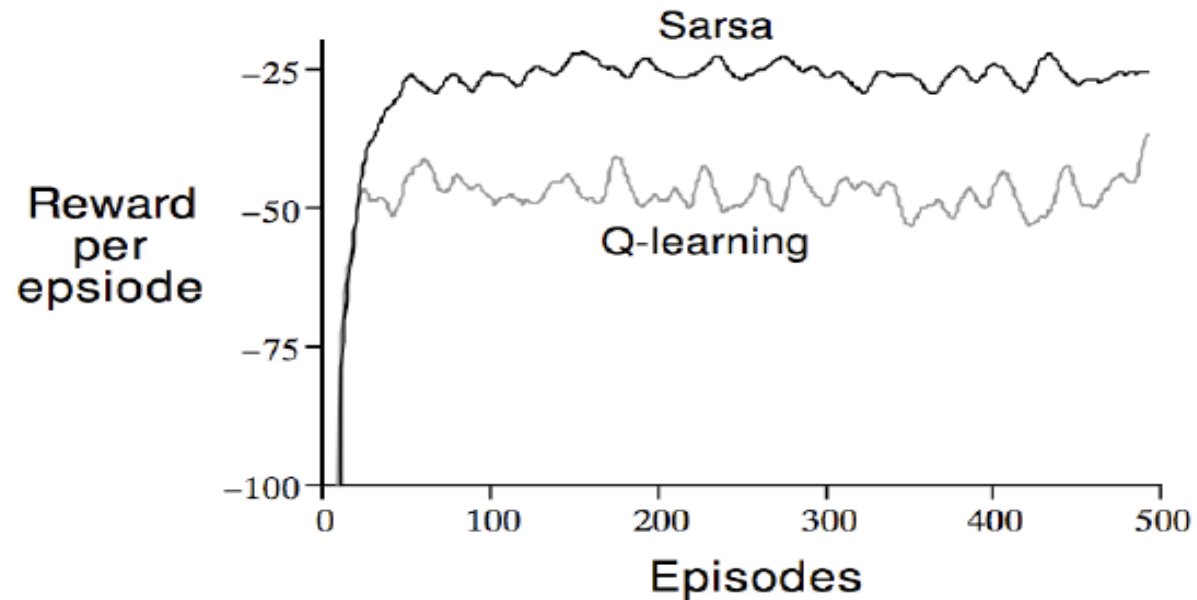
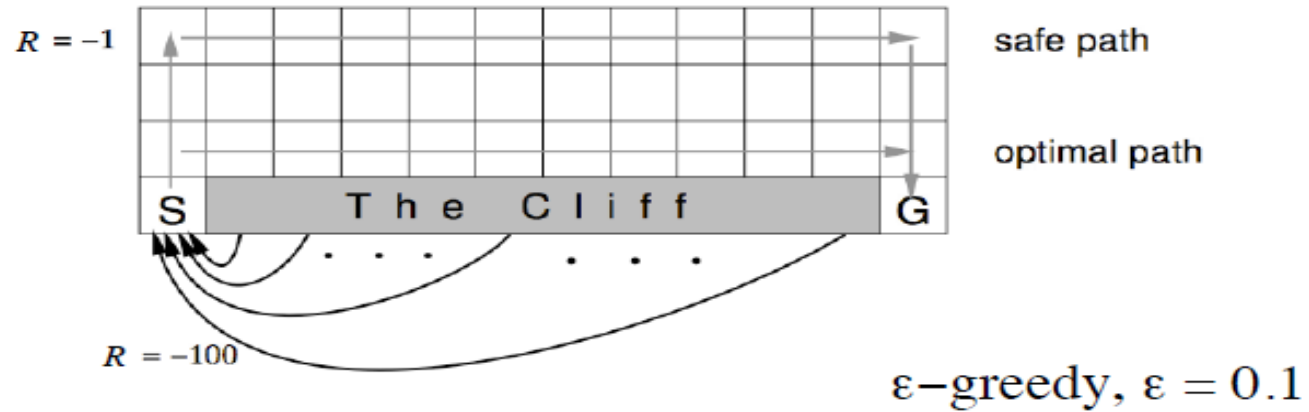
$r=0$ for non-terminal states

$\gamma=0.9$

$\alpha = 0.5$



Example: Sarsa vs Q-Learning



Summary

Basic RL model

Prediction Problem vs Control Problem

Temporal Difference Updates

Prediction TD(0)

Control

Sarsa: On-Policy

Q-Learning: Off-Policy