

Introduction to Reinforcement Learning

Part 5:

Temporal-Difference Learning

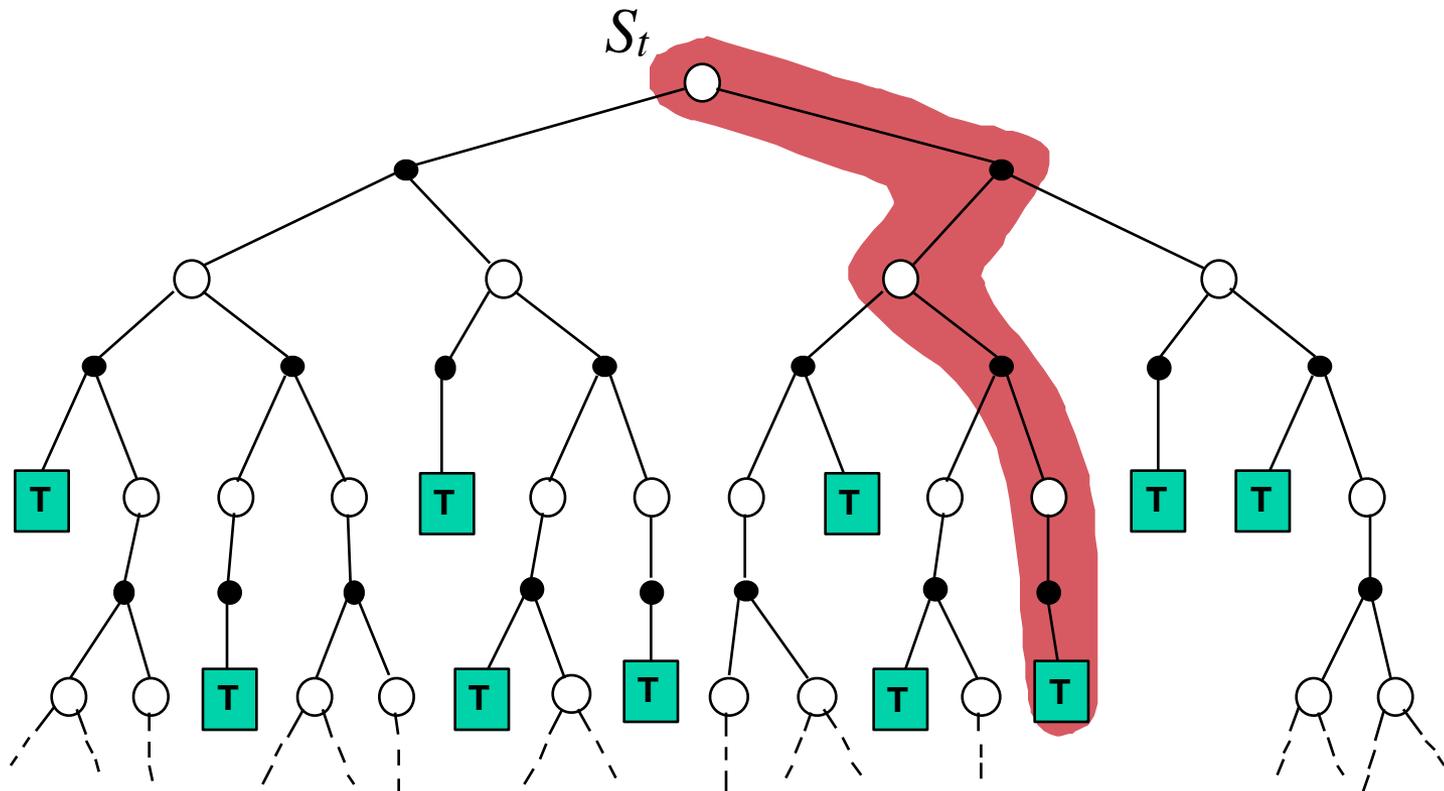
What everybody should know about

Temporal-difference (TD) learning

- Used to learn value functions without human input
- Learns a guess from a guess
- Applied by Samuel to play Checkers (1959) and by Tesauro to beat humans at Backgammon (1992-5) and Jeopardy! (2011)
- Explains (accurately models) the brain reward systems of primates, rats, bees, and many other animals (Schultz, Dayan & Montague 1997)
- Arguably solves Bellman's "curse of dimensionality"

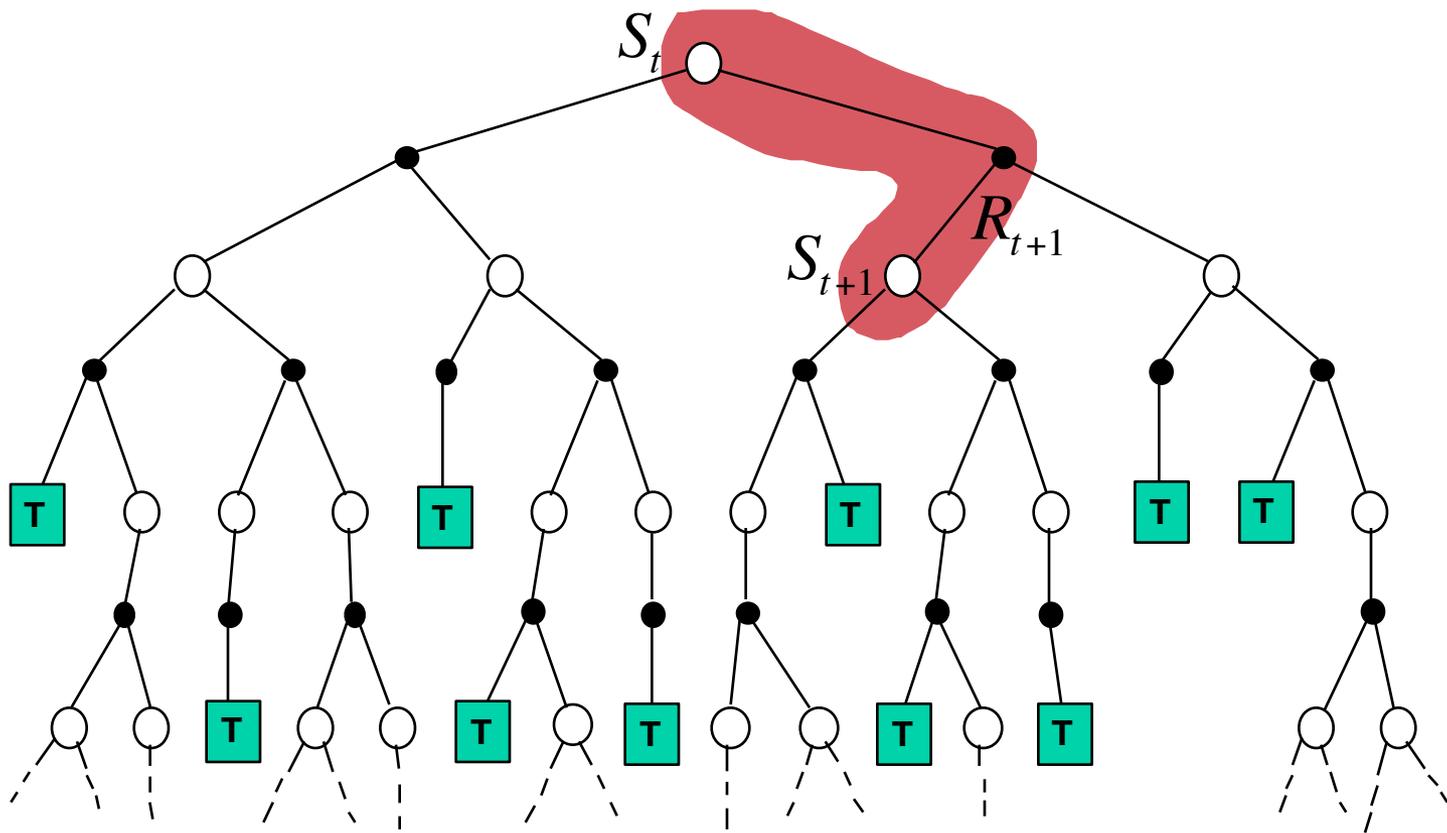
Simple Monte Carlo

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



Simplest TD Method

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



TD Prediction

Policy Evaluation (the prediction problem):

for a given policy π , compute the state-value function v_π

Recall: Simple every-visit Monte Carlo method:

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

target: the actual return after time t

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

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

target: an estimate of the return

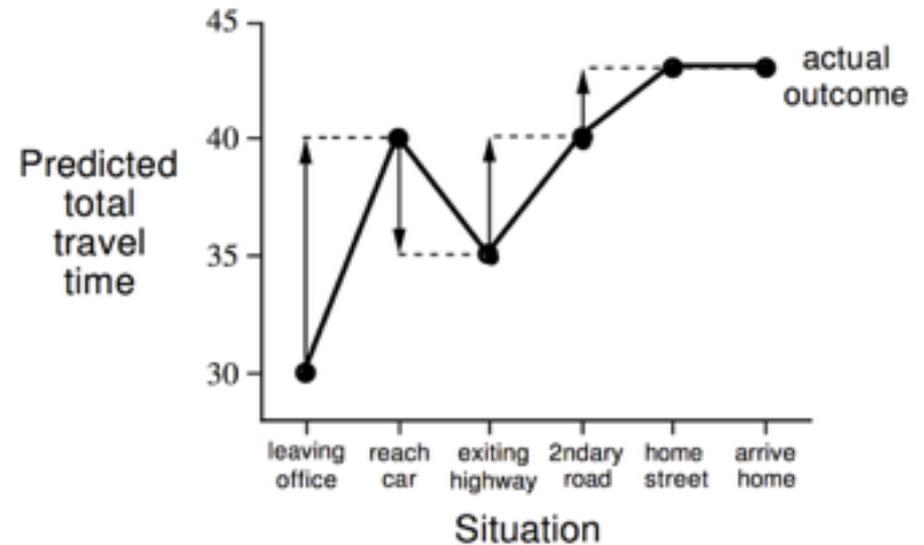
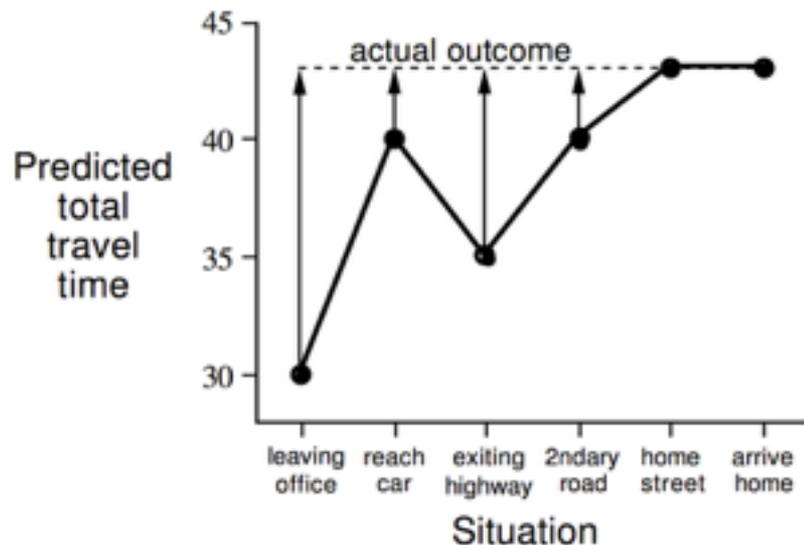
Example: Driving Home

| State | Elapsed Time (minutes) | Predicted Time to Go | Predicted Total Time | |
|--------------------|-----------------------------------|---------------------------------|---------------------------------|------|
| leaving office | 0 | 30 | 30 | |
| | (5) | | | bad |
| reach car, raining | 5 | 35 | 40 | |
| | (15) | | | good |
| exit highway | 20 | 15 | 35 | |
| | (10) | | | bad |
| behind truck | 30 | 10 | 40 | |
| | (10) | | | bad |
| home street | 40 | 3 | 43 | |
| | (3) | | | ok |
| arrive home | 43 | 0 | 43 | |

Driving Home

Changes recommended by Monte Carlo methods ($\alpha=1$)

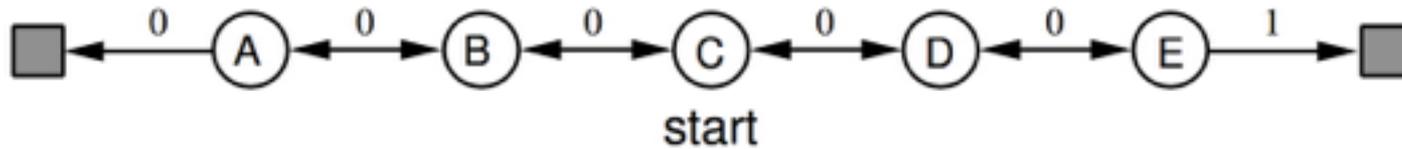
Changes recommended by TD methods ($\alpha=1$)



Advantages of TD Learning

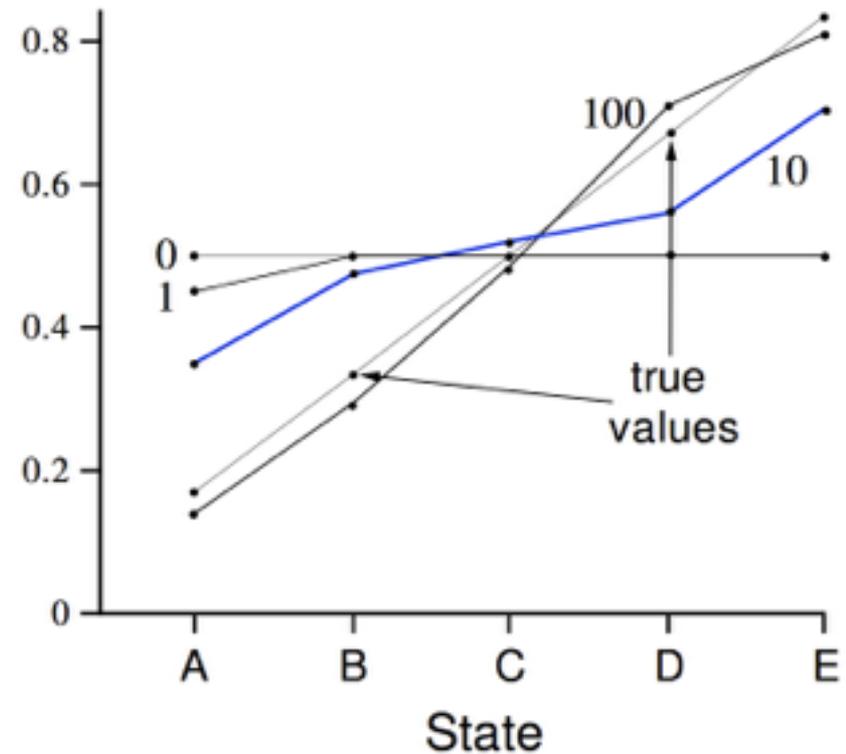
- TD methods do not require a model of the environment, only experience
- TD, but not MC, methods can be fully incremental
 - You can learn **before** knowing the final outcome
 - Less memory
 - Less peak computation
 - You can learn **without** the final outcome
 - From incomplete sequences
- Both MC and TD converge (under certain assumptions to be detailed later), but which is faster?

Random Walk Example

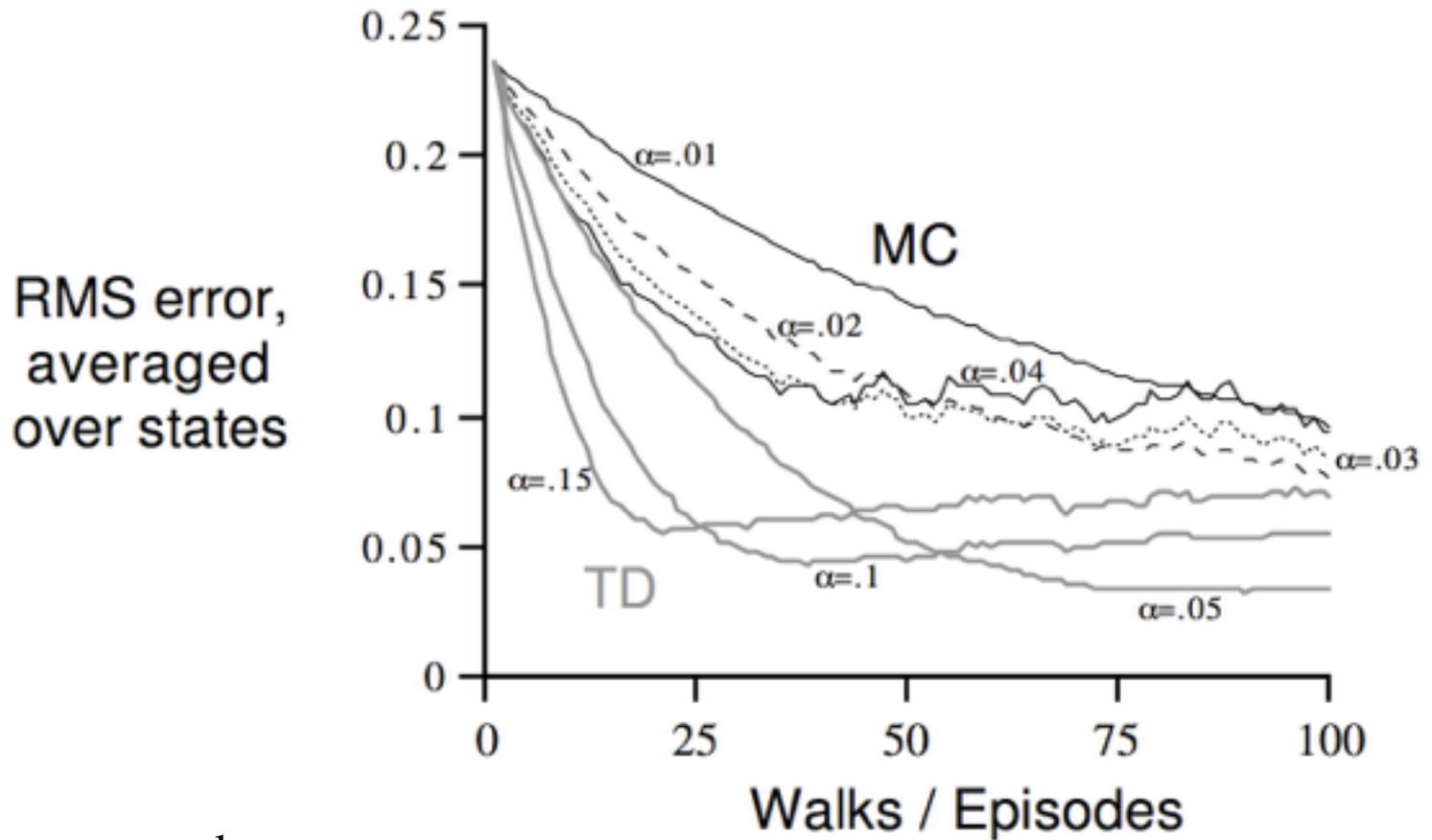


Estimated
value

Values learned by TD(0) after
various numbers of episodes



TD and MC on the Random Walk



Data averaged over
100 sequences of episodes

Batch Updating in TD and MC methods

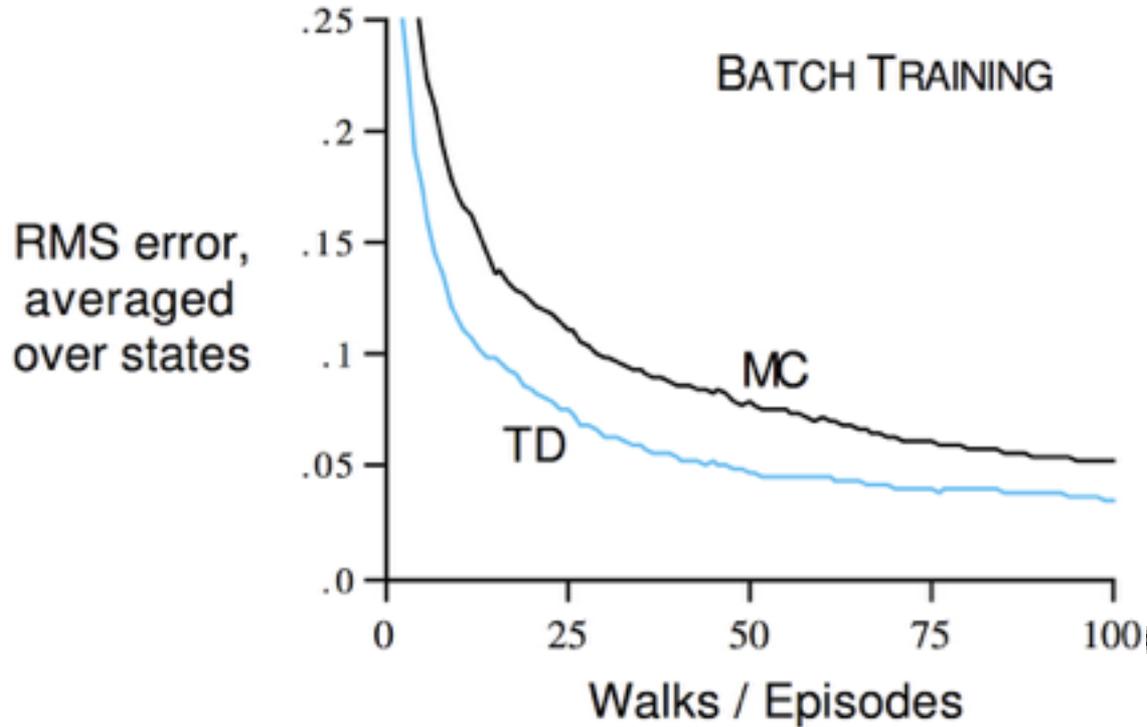
Batch Updating: train completely on a finite amount of data, e.g., train repeatedly on 10 episodes until convergence.

Compute updates according to TD(0), but only update estimates after each complete pass through the data.

For any finite Markov prediction task, under batch updating, TD(0) converges for sufficiently small α .

Constant- α MC also converges under these conditions, **but to a difference answer!**

Random Walk under Batch Updating



After each new episode, all previous episodes were treated as a batch, and algorithm was trained until convergence. All repeated 100 times.

You are the Predictor

Suppose you observe the following 8 episodes:

A, 0, B, 0

B, 1

B, 1

B, 1

B, 1

B, 1

B, 1

B, 0

$V(B)?$ 0.75

$V(A)?$ 0?

You are the Predictor

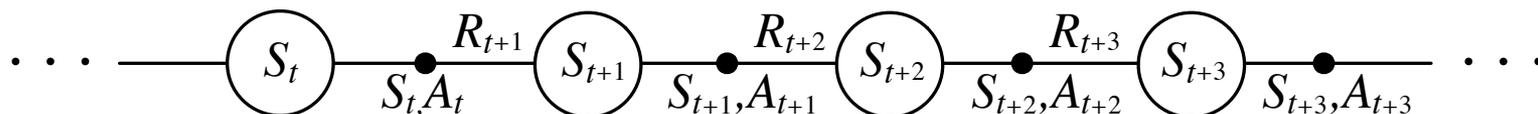
- The prediction that best matches the training data is $V(A)=0$
 - This **minimizes the mean-square-error** on the training set
 - This is what a batch Monte Carlo method gets
- If we consider the sequentiality of the problem, then we would set $V(A)=.75$
 - This is correct for the **maximum likelihood** estimate of a Markov model generating the data
 - i.e, if we do a best fit Markov model, and assume it is exactly correct, and then compute what it predicts (how?)
 - This is called the **certainty-equivalence estimate**
 - This is what TD(0) gets

Summary (so far)

- Introduced *one-step tabular model-free TD methods*
- These methods bootstrap and sample, combining aspects of DP and MC methods
- TD prediction
- If the world is truly Markov, then TD methods will learn faster than MC methods
- MC methods have lower error on past data, but higher error on future data
- Extend prediction to control by employing some form of GPI
 - On-policy control: Sarsa
 - Off-policy control: Q-learning

Learning An Action-Value Function

Estimate q_π for the current policy π



After every transition from a nonterminal state, S_t , do this:

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

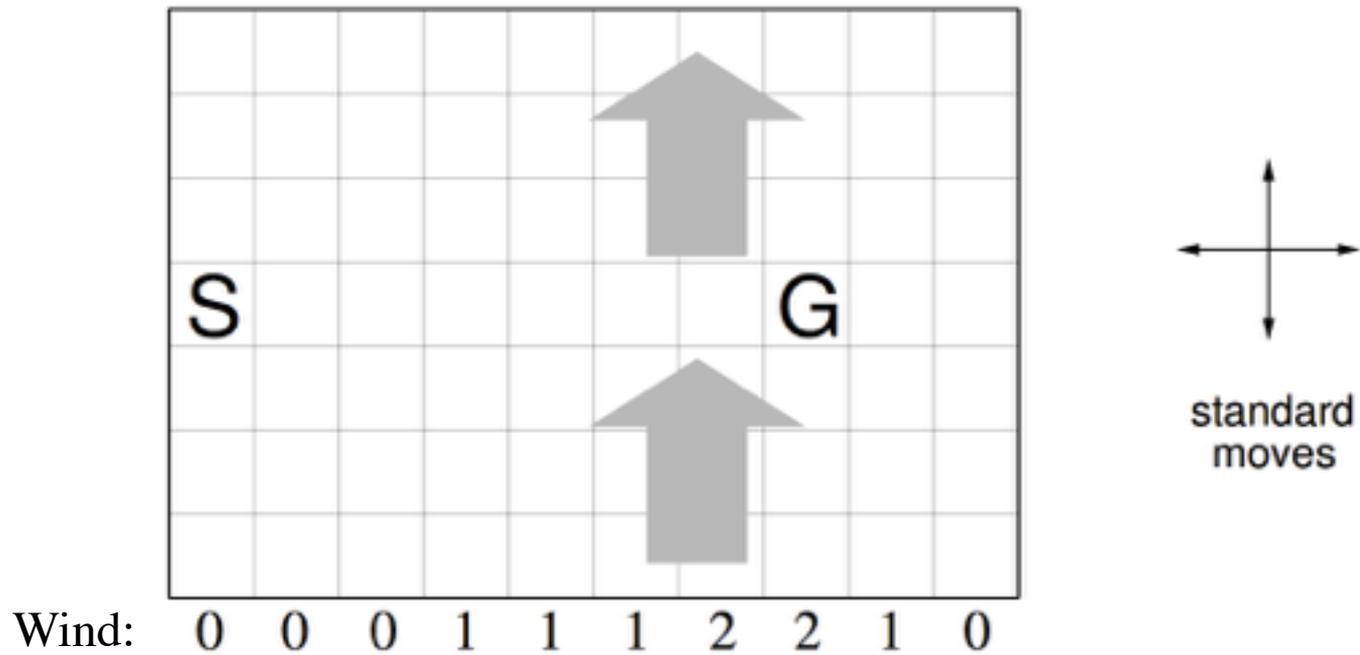
If S_{t+1} is terminal, then define $Q(S_{t+1}, A_{t+1}) = 0$

Sarsa: On-Policy TD Control

Turn this into a control method by always updating the policy to be greedy with respect to the current estimate:

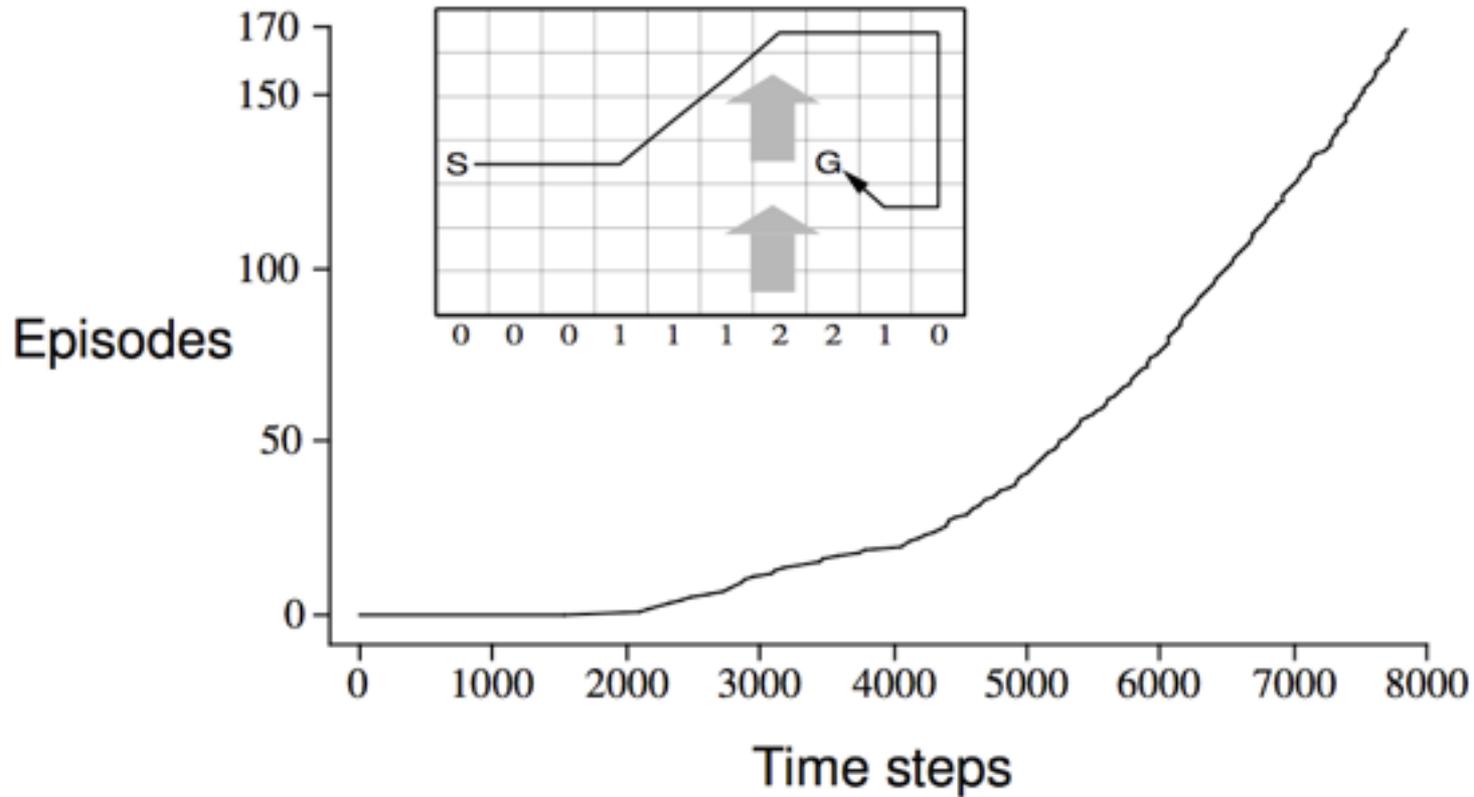
```
Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$ 
Repeat (for each episode):
  Initialize  $S$ 
  Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
  Repeat (for each step of episode):
    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
```

Windy Gridworld



undiscounted, episodic, reward = -1 until goal

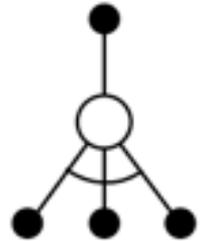
Results of Sarsa on the Windy Gridworld



Q-Learning: Off-Policy TD Control

One-step Q-learning:

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



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

Summary

- Introduced *one-step tabular model-free TD methods*
- These methods bootstrap and sample, combining aspects of DP and MC methods
- TD prediction
- If the world is truly Markov, then TD methods will learn faster than MC methods
- MC methods have lower error on past data, but higher error on future data
- Extend prediction to control by employing some form of GPI
 - On-policy control: *Sarsa*
 - Off-policy control: *Q-learning*