

# Chapter 6: Temporal Difference Learning

---

Objectives of this chapter:

- Introduce Temporal Difference (TD) learning
- Focus first on policy evaluation, or prediction, methods
- Compare efficiency of TD learning with MC learning
- Then extend to control methods

# TD methods bootstrap and sample

- ▶ Bootstrapping: update involves an estimate of the value function
  - TD and DP methods bootstrap
  - MC methods **do not** bootstrap
- ▶ Sampling: update **does not** involve an expected value
  - TD and MC method sample
  - Classical DP **does not** sample

# TD Prediction

---

**Policy Evaluation (the prediction problem):**

for a given policy  $\pi$ , compute the state-value function  $v_\pi$

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):

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

**target**: an estimate of the return

# TD target for prediction

- ▶ The TD target:  $R_{t+1} + \gamma v_\pi(S_{t+1})$ 
  - it is an *estimate* like MC target because it **samples** the expected value
  - it is an *estimate* like DP target because it uses the current estimate of V instead of  $v_\pi$  — **bootstrapping**

## Tabular TD(0) for estimating $v_\pi$

Input: the policy  $\pi$  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  $\pi$  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

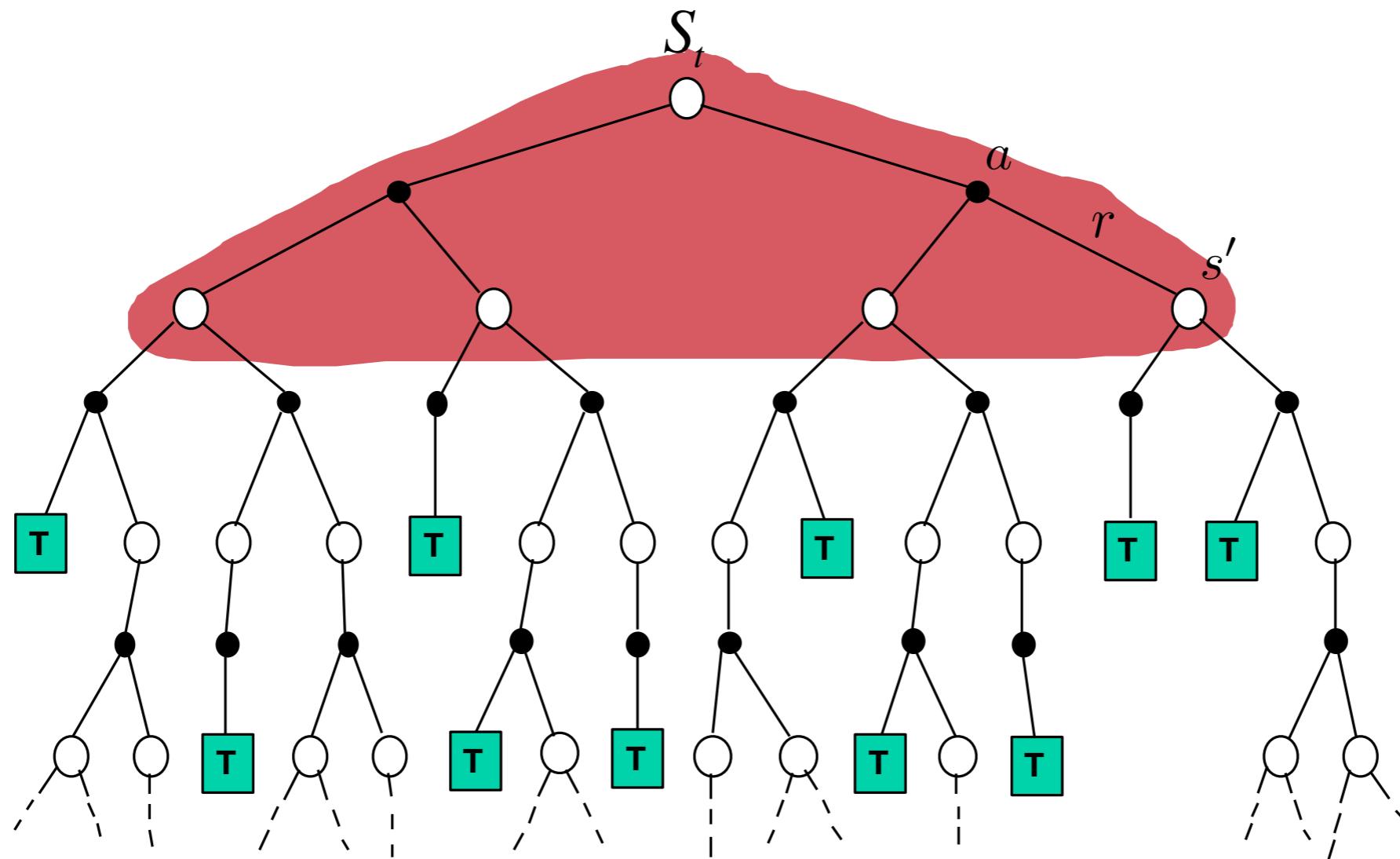
**Agent program**

**Environment program**

**Experiment program**

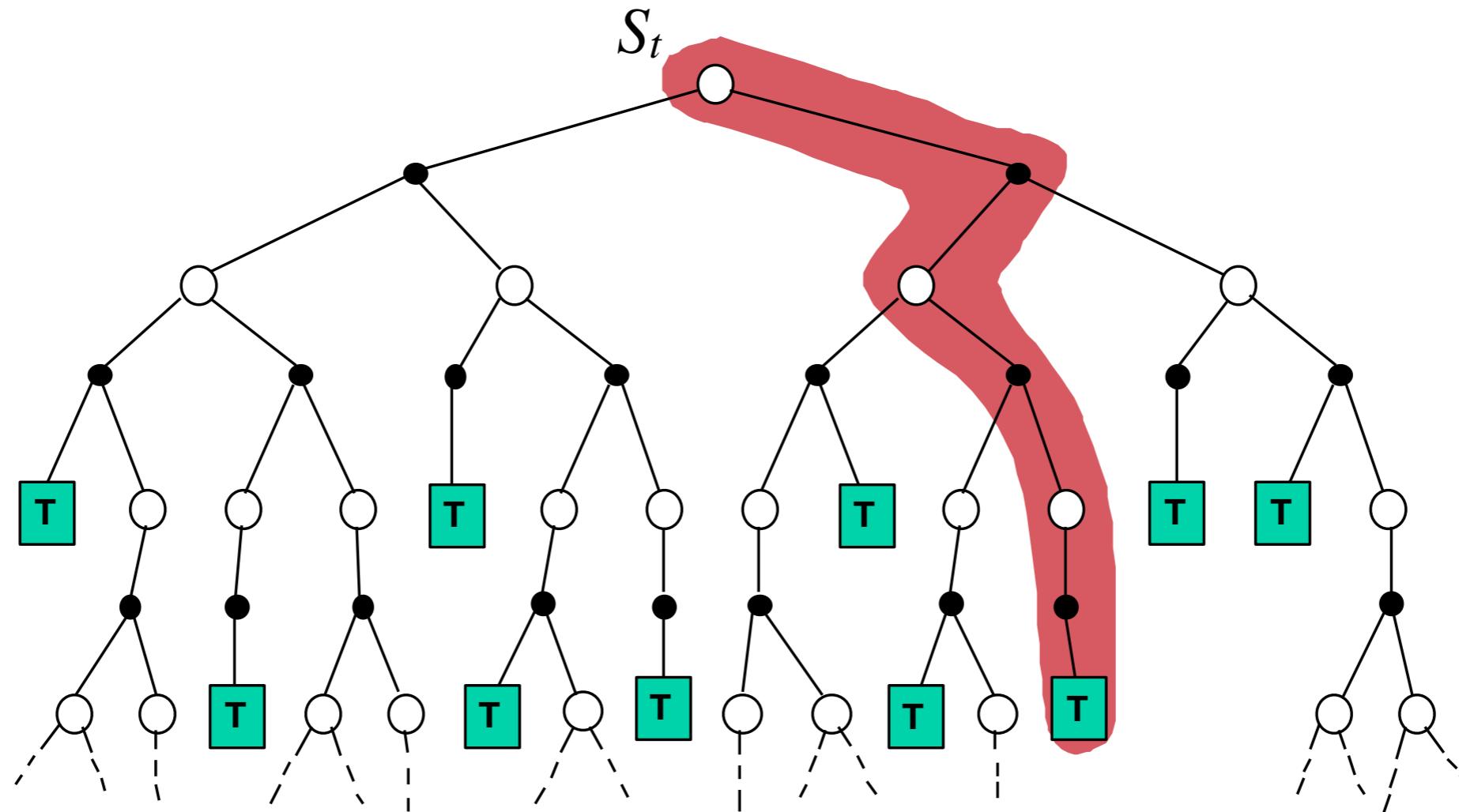
# Dynamic programming

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



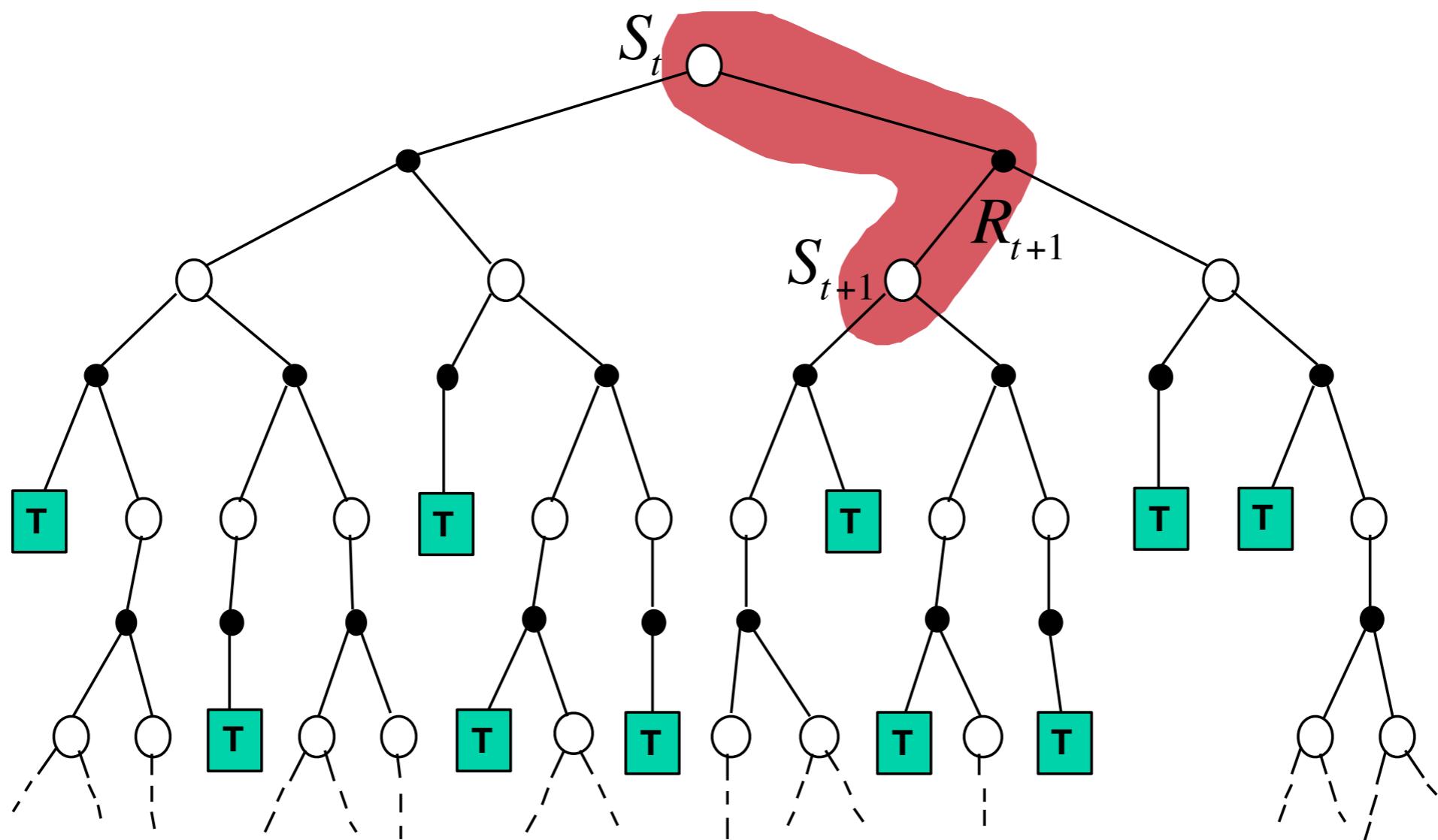
# Simple Monte Carlo

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



# Simplest TD method

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



# Example: Driving Home

- ▶ Consider driving home:
  - each day you drive home
  - your goal is to try and predict how long it will take at particular stages
  - when you leave office you note the time, day, & other relevant info
- ▶ Consider the policy evaluation or prediction task

# Driving Home

<i>State</i>	<i>Elapsed Time</i> <i>(minutes)</i>	<i>Predicted</i> <i>Time to Go</i>	<i>Predicted</i> <i>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

- ▶ Imagine continually predicting drive time (starting @ desk)

# Driving home as an RL problem

- ▶ Rewards = 1 per step (if we were minimizing travel time what would reward be?)
- ▶  $\gamma = 1$
- ▶  $G_t$  = time to go from state  $S_t$
- ▶  $V(S_t)$  = expected time to get home from  $S_t$

# Updating our predictions

- ▶ Goal: update the prediction of total time from leaving office, while driving home
- ▶ With MC we would need to wait for a termination—until we get home—then calculate  $G_t$  for each step of episode, then apply our updates

# Driving home

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

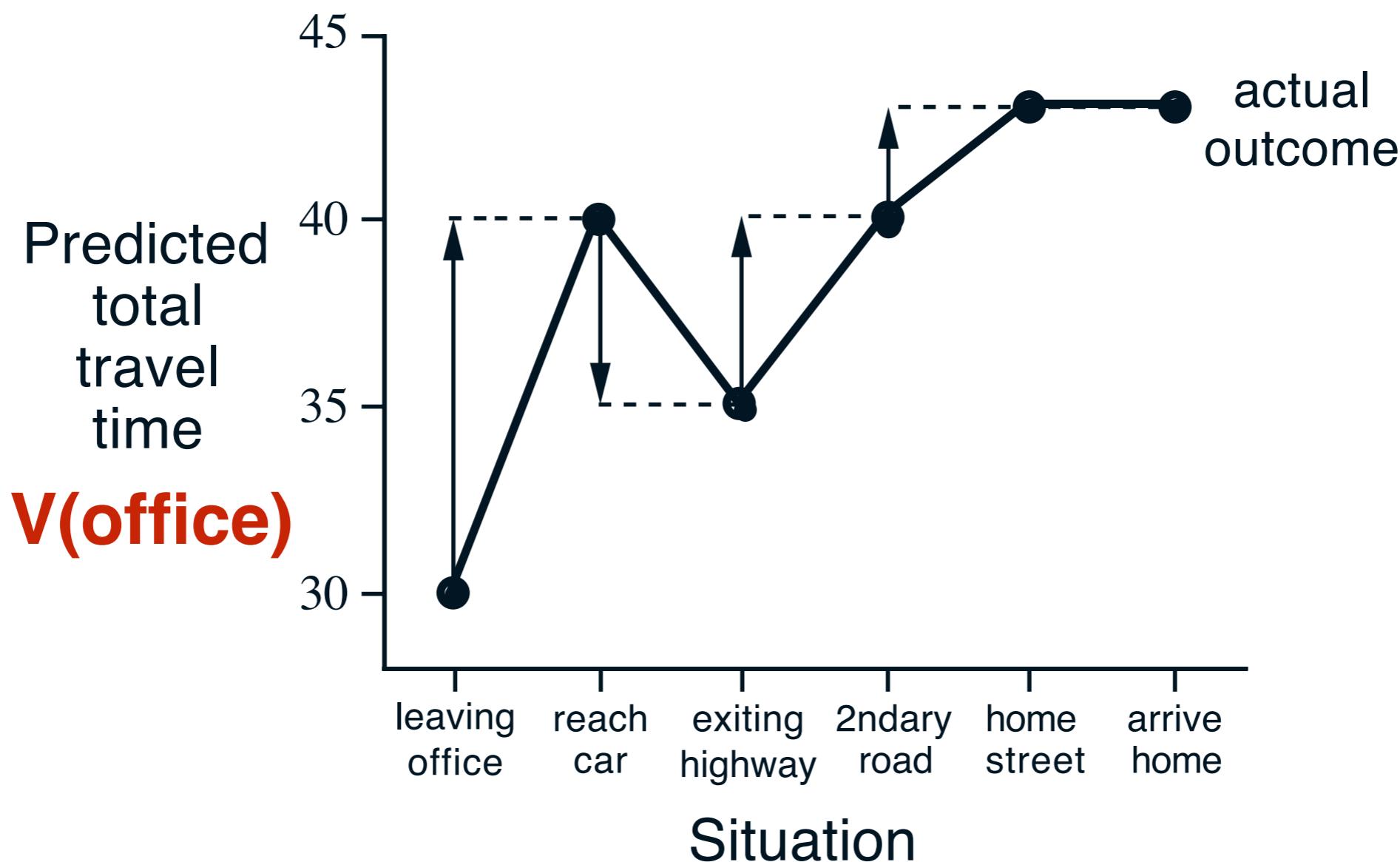
- ▶ Task: update **V** as we go based on observed elapsed time—Reward column

# Driving home

State	Elapsed Time (minutes)	V(s)	V(office)	Predicted Total Time
		R	Predicted Time to Go	
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

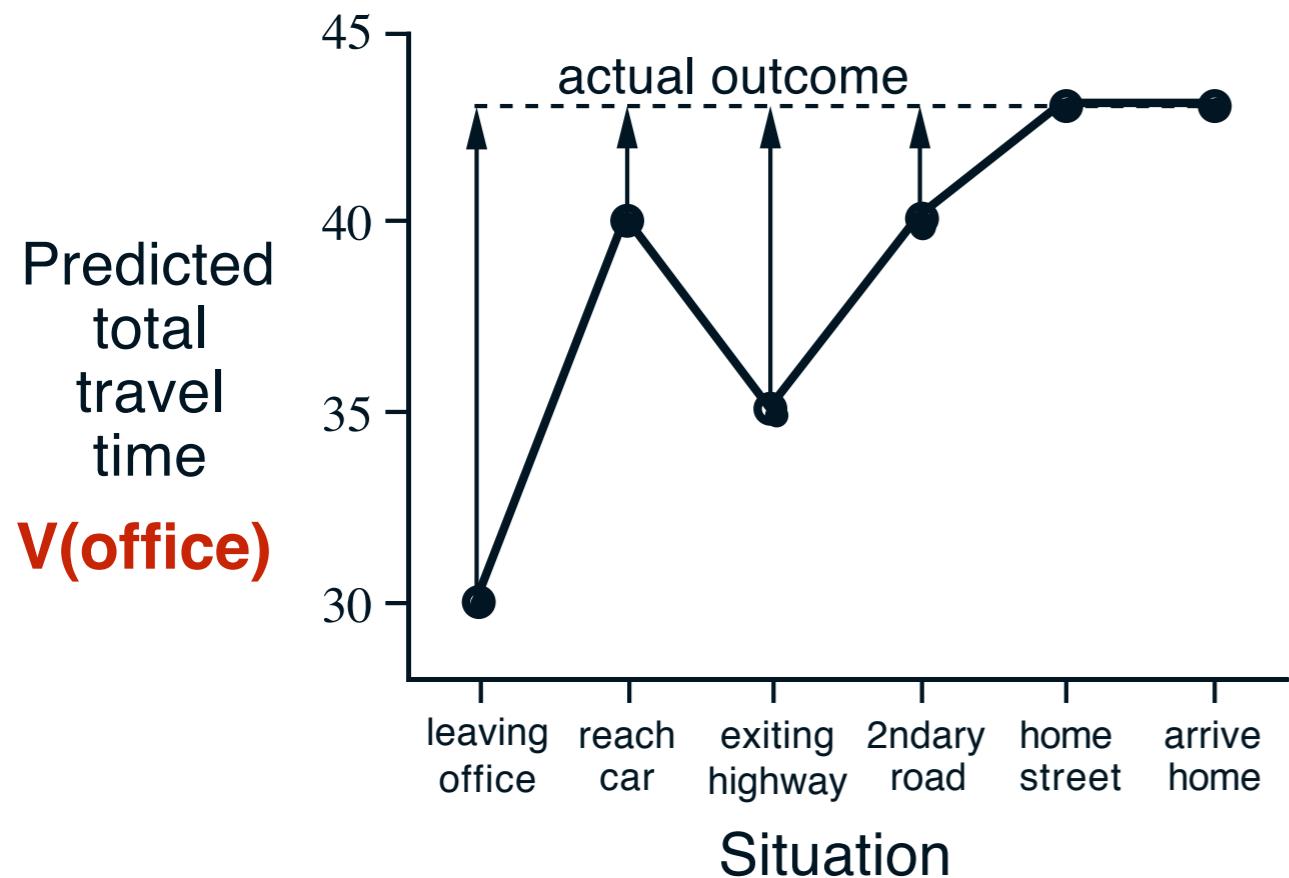
- ▶ update  $V(\text{office})$  with  $\alpha = 1$ ?
  - $V(s) = V(s) + \alpha[R_{t+1} + \gamma V(s') - V(s)]$
  - $V(\text{office}) = V(\text{office}) + \alpha[R_{t+1} + \gamma V(\text{car}) - V(\text{office})]$
  - new  $V(\text{office}) = 40$ ;  $\Delta = +10$
- ▶ update  $V(\text{car})$ ?
  - $V(\text{car}) = 30$ ;  $\Delta = -5$
- ▶ update  $V(\text{exit})$ ?
  - $V(\text{exit}) = 20$ ;  $\Delta = +5$

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

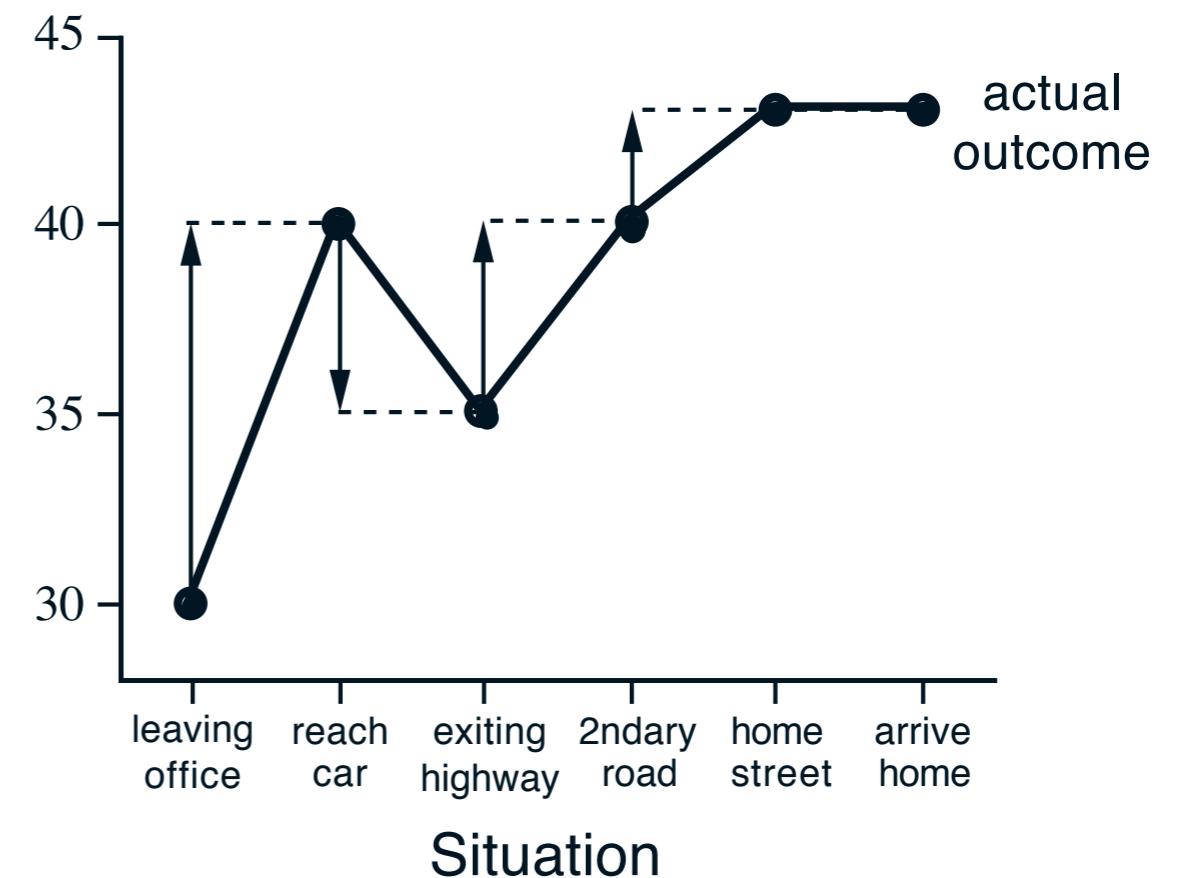


# Driving Home

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



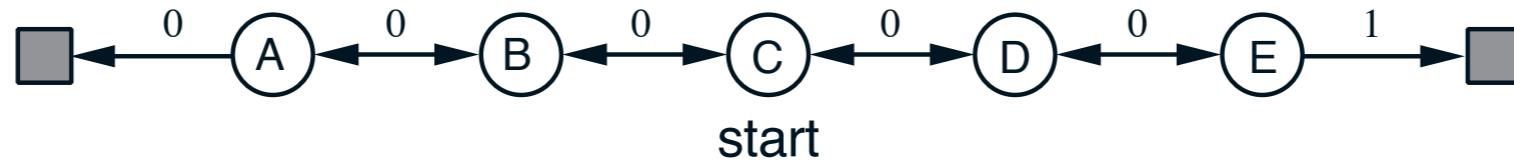
Changes recommended  
by TD methods ( $\alpha=1$ )



# Advantages of TD learning

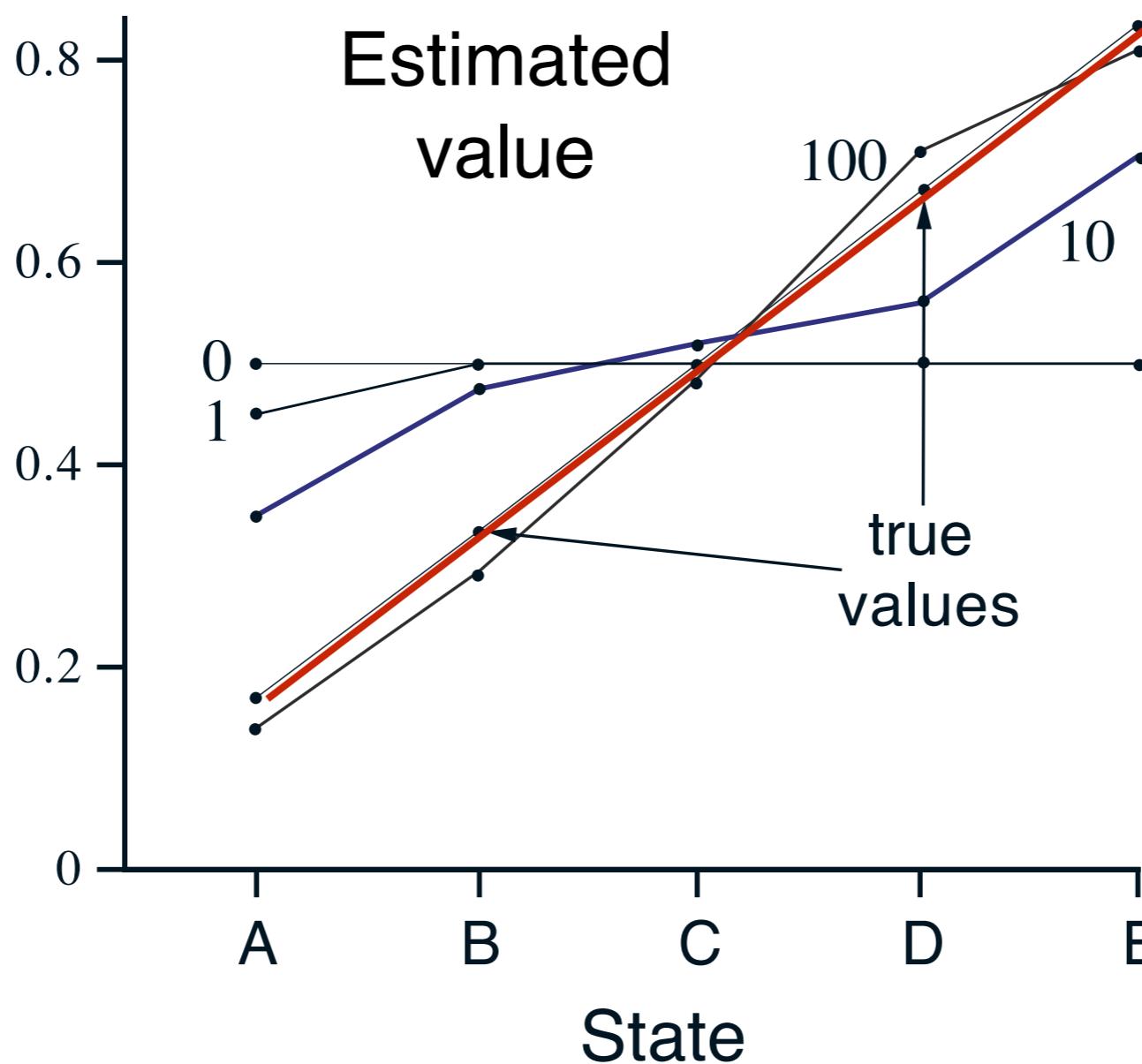
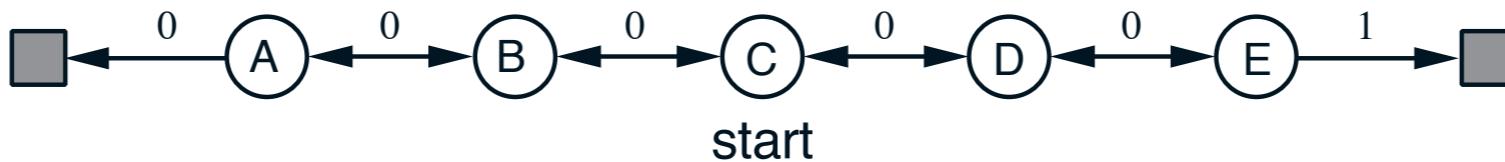
- ▶ TD methods do not require a model of the environment, only experience
- ▶ TD methods can be fully incremental
  - ▶ Make updates **before** knowing the final outcome
  - ▶ Requires less memory
  - ▶ Requires less peak computation
- ▶ You can learn **without** the final outcome, from incomplete sequences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

# Random walk



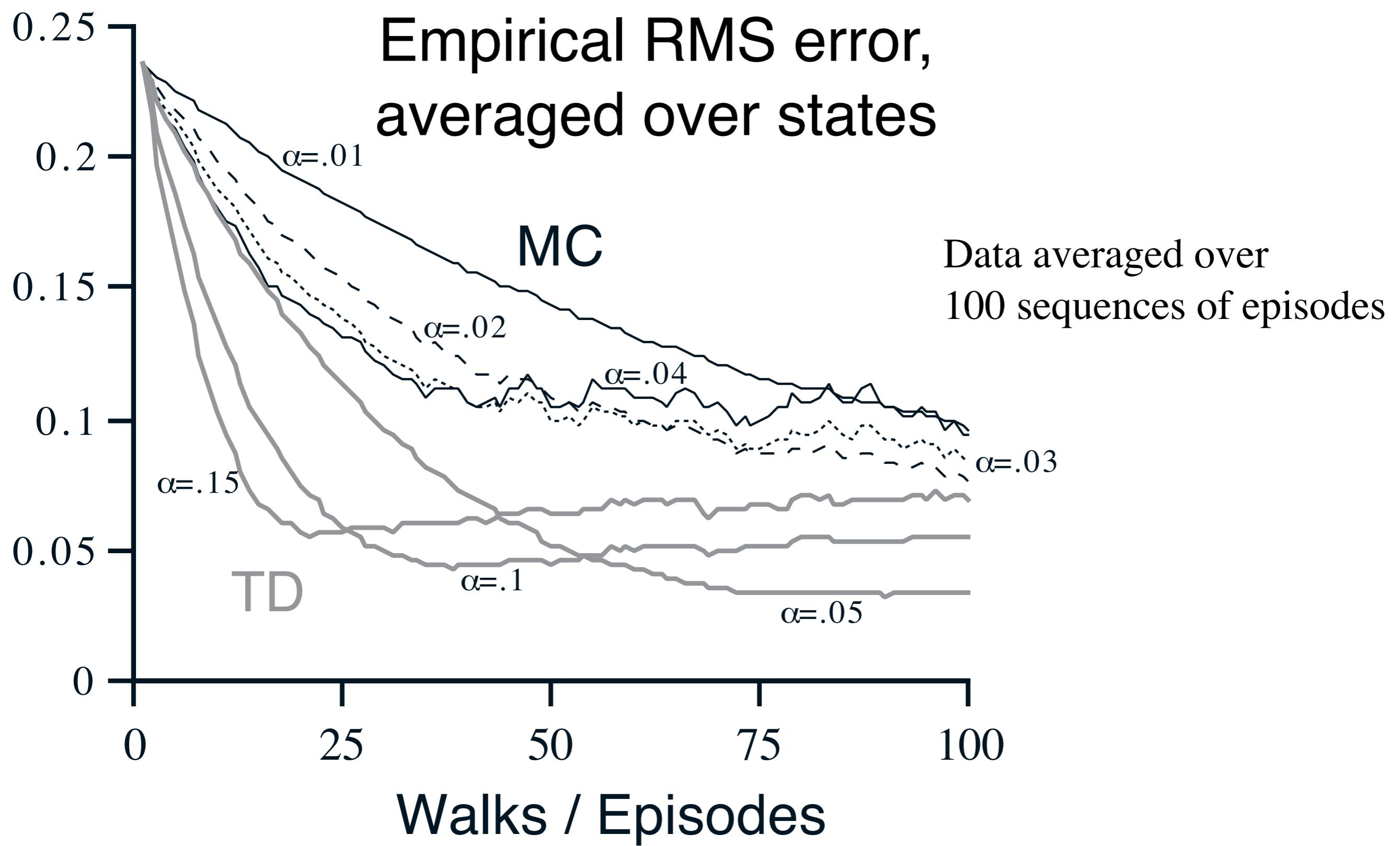
- ▶ C is start state, episodic, undiscounted  $\gamma = 1$
- ▶  $\pi$  is left or right with equal probability in all states
- ▶ termination at either end
- ▶ rewards +1 on **right** termination, 0 otherwise
- ▶ what does  $v_\pi(s)$  tell us?
  - probability of termination on right side from each state, under random policy
  - what is  $v_\pi = [A \ B \ C \ D \ E]$ ?
    - $v_\pi = [1/6 \ 2/6 \ 3/6 \ 4/6 \ 5/6]$
- ▶ Initialize  $V(s) = 0.5 \ \forall s \in \mathcal{S}$

# Values learned by TD from one run, after various numbers of episodes



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

# TD and MC on the Random Walk



# Batch Updating in TD and MC methods

---

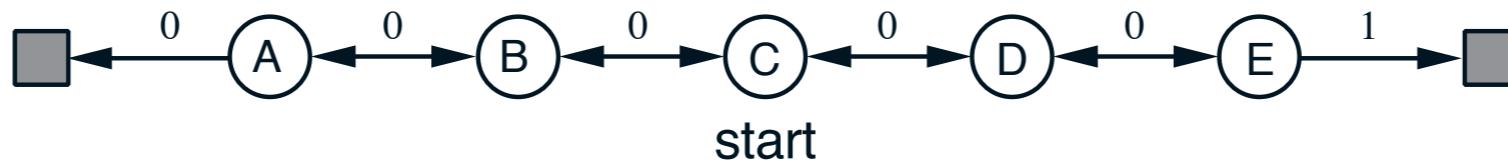
**Batch Updating:** train completely on a finite amount of data,  
e.g., train repeatedly on 10 episodes until convergence.

Compute updates according to TD or MC, but only update  
estimates after each complete pass through the data.

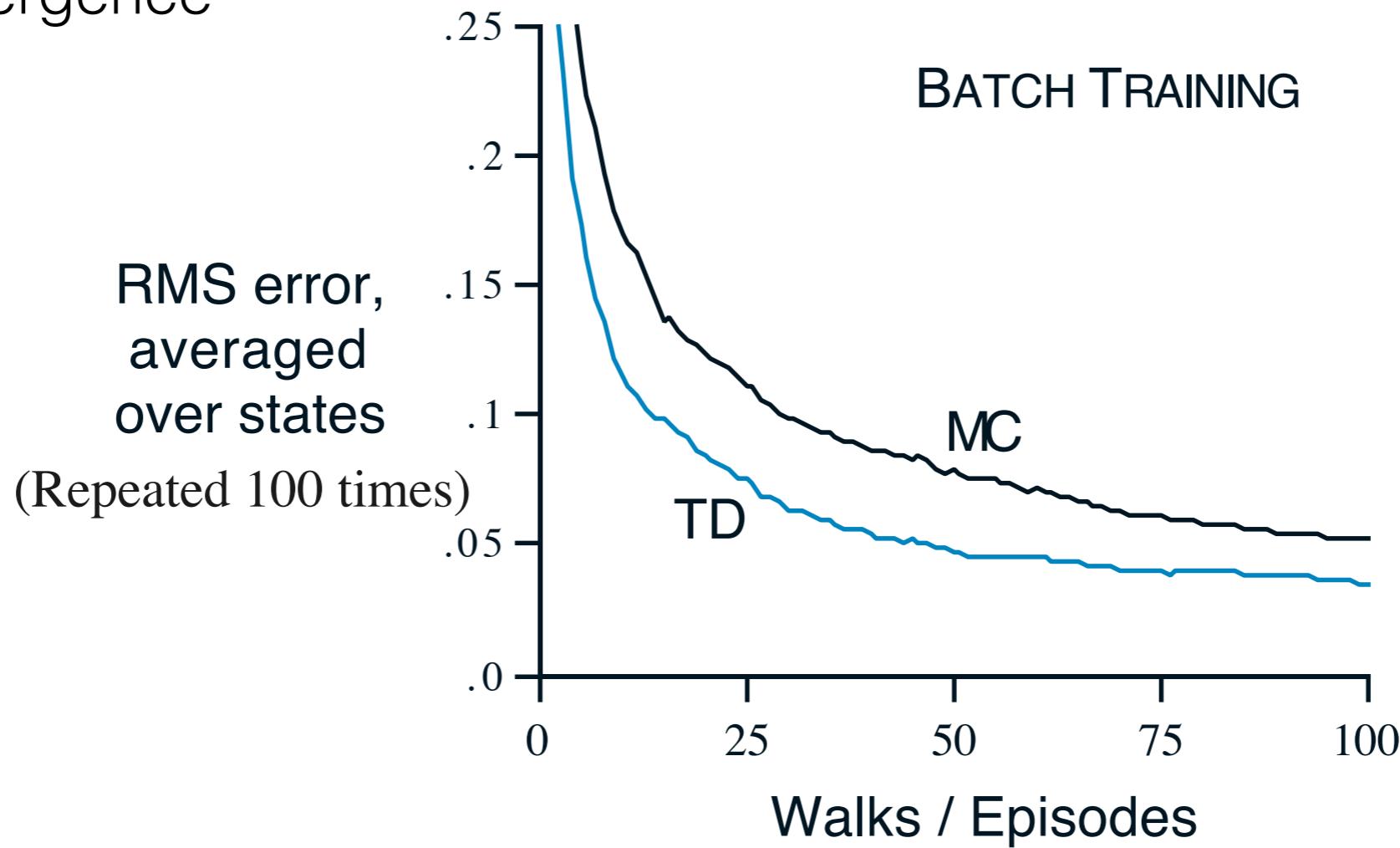
For any finite Markov prediction task, under batch updating,  
TD converges for sufficiently small  $\alpha$ .

Constant- $\alpha$  MC also converges under these conditions, **but to  
a different answer!**

# Random Walk under Batch Updating



- After each new episode, all episodes seen so far are treated as a batch
- This growing batch is repeatedly processed by TD and MC until convergence



# You are the Predictor

---

Suppose you observe the following 8 episodes:

A, 0, B, 0

B, 1

B, 1

$V(B)? \quad 0.75$

B, 1

$V(A)? \quad 0?$

B, 1

B, 1

B, 0

Assume Markov states, no discounting ( $\gamma = 1$ )

# Consider the following data

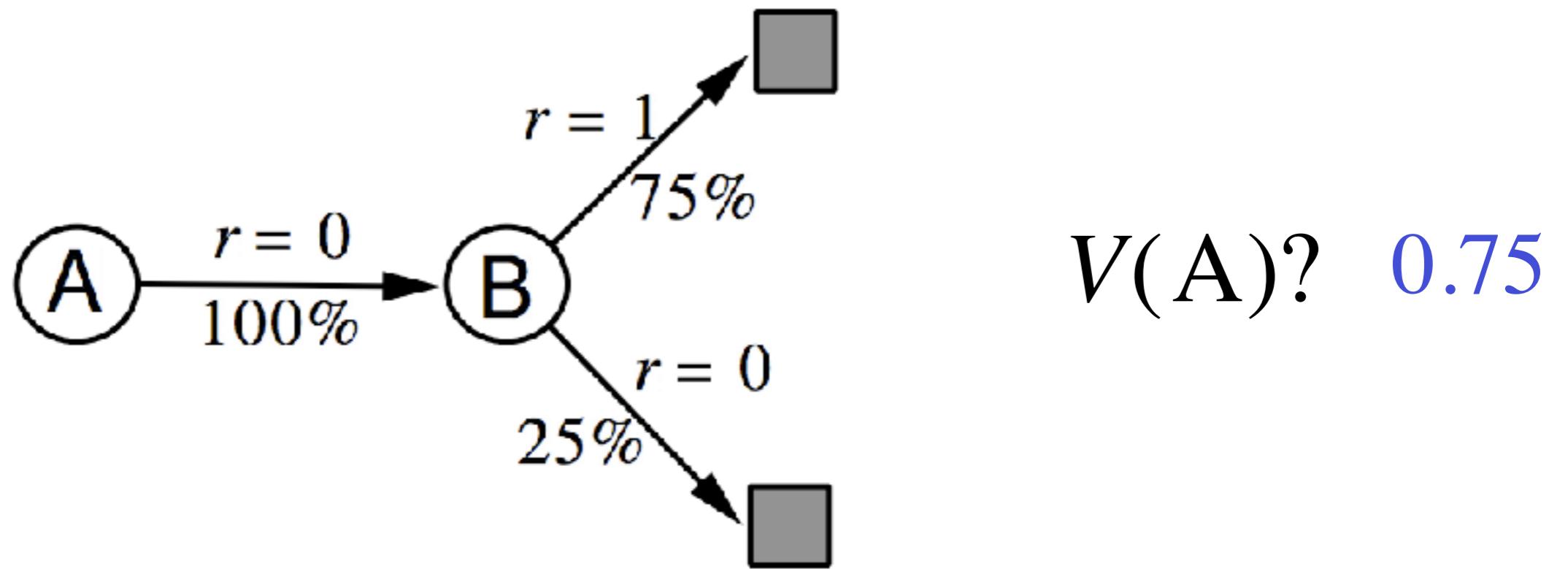
A, 0	B, 0	B, 1
B, 1		B, 1
B, 1		B, 1
B, 1		B, 0

- ▶  $V(B)$ 
  - 6 out of 8 times we saw a 1;  $V(B) = 3/4$
- ▶ The batch MC prediction for  $V(A)$ :
  - 100% of returns from A equal zero;  $V(A) = 0$

# You are the Predictor

---

- Modeling the Markov process based on the observed training data



# Optimality of TD(0)

- ▶ The prediction that best matches the training data is  $V(A)=0$ :
  - This minimizes the **mean-square-error** between  $V(s)$  and the sample returns in the training set. (**zero** MSE in our example)
  - Under batch training, this is what constant- $\alpha$  MC gets
- ▶ TD(0) achieves a different type of optimality, where  $V(A)=0.75$ 
  - This is correct for the maximum likelihood estimate of the Markov model generating the data
  - i.e., if we do a best fit Markov model, and assume it is exactly correct, and then compute the predictions
  - This is called the **certainty-equivalence estimate**
  - This is what TD gets

# Advantages of TD

- ▶ If the process is Markov, we expect the TD estimate to produce lower error on future data
- ▶ This helps explain why TD methods converge more quickly than MC in the batch setting
- ▶ TD(0) makes progress towards the certainty-equivalence estimate without explicitly building the model!

# Summary so far

---

- Introduced *one-step tabular model-free TD methods*
- These methods bootstrap and sample, combining aspects of DP and MC methods
- TD methods are *computationally congenial*
- 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

# Unified View

