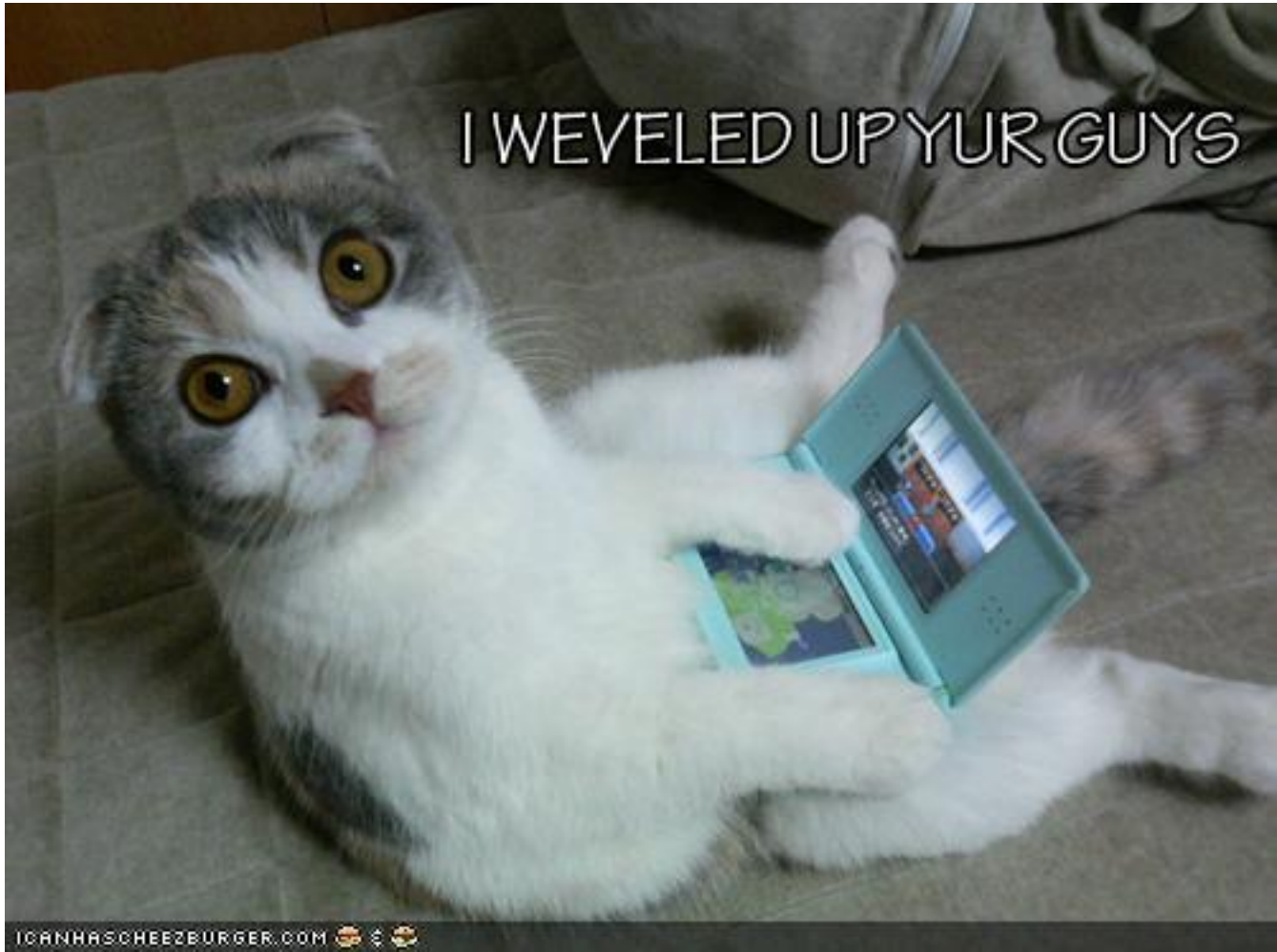
- Signup sheet
- New homework up: Due Friday



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:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[R_t - V(s_t) \right]$$

target: the actual return after time t

The simplest TD method, TD(0):

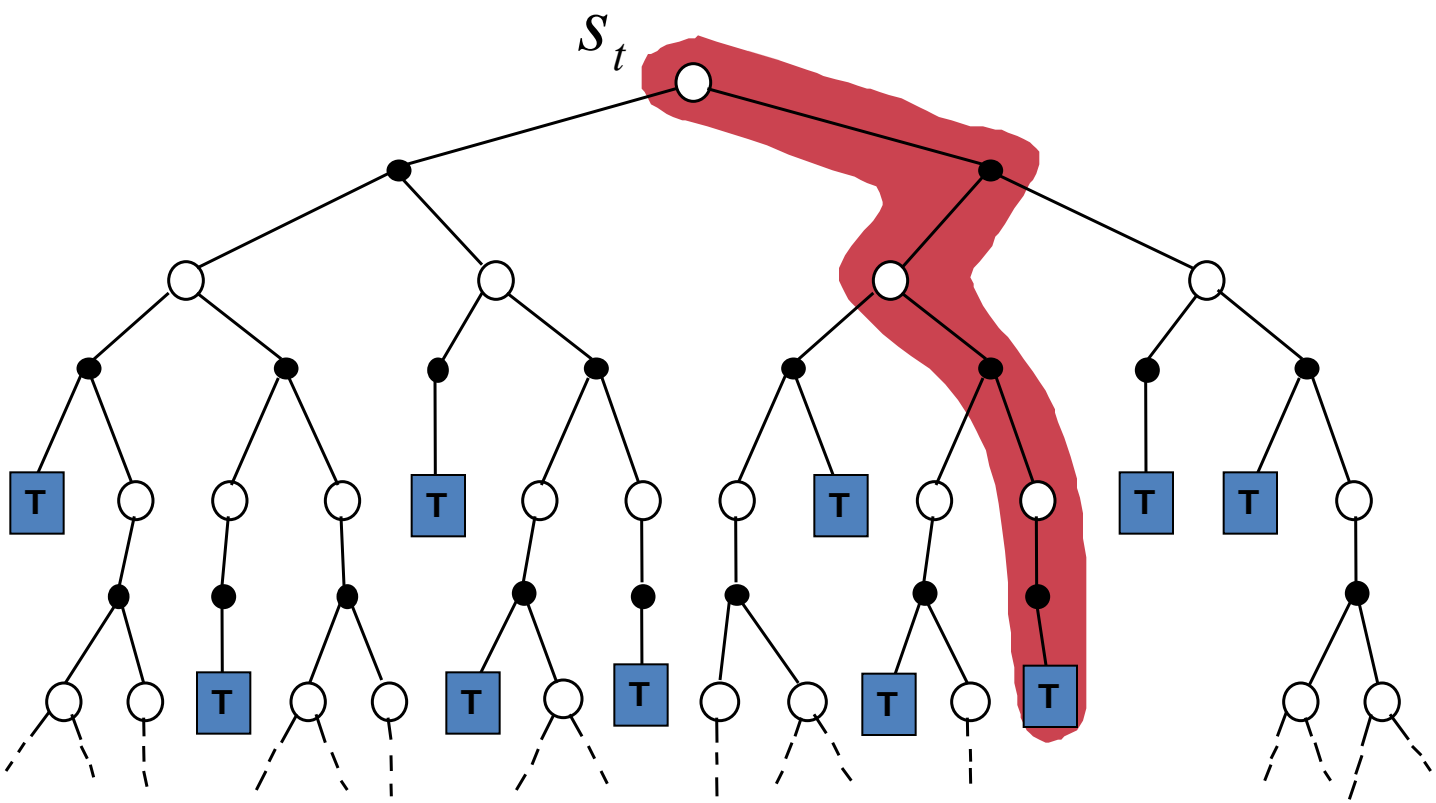
$$V(s_t) \leftarrow V(s_t) + \alpha \left[\underbrace{r_{t+1} + \gamma V(s_{t+1})} - V(s_t) \right]$$

target: an estimate of the return

Simple Monte Carlo

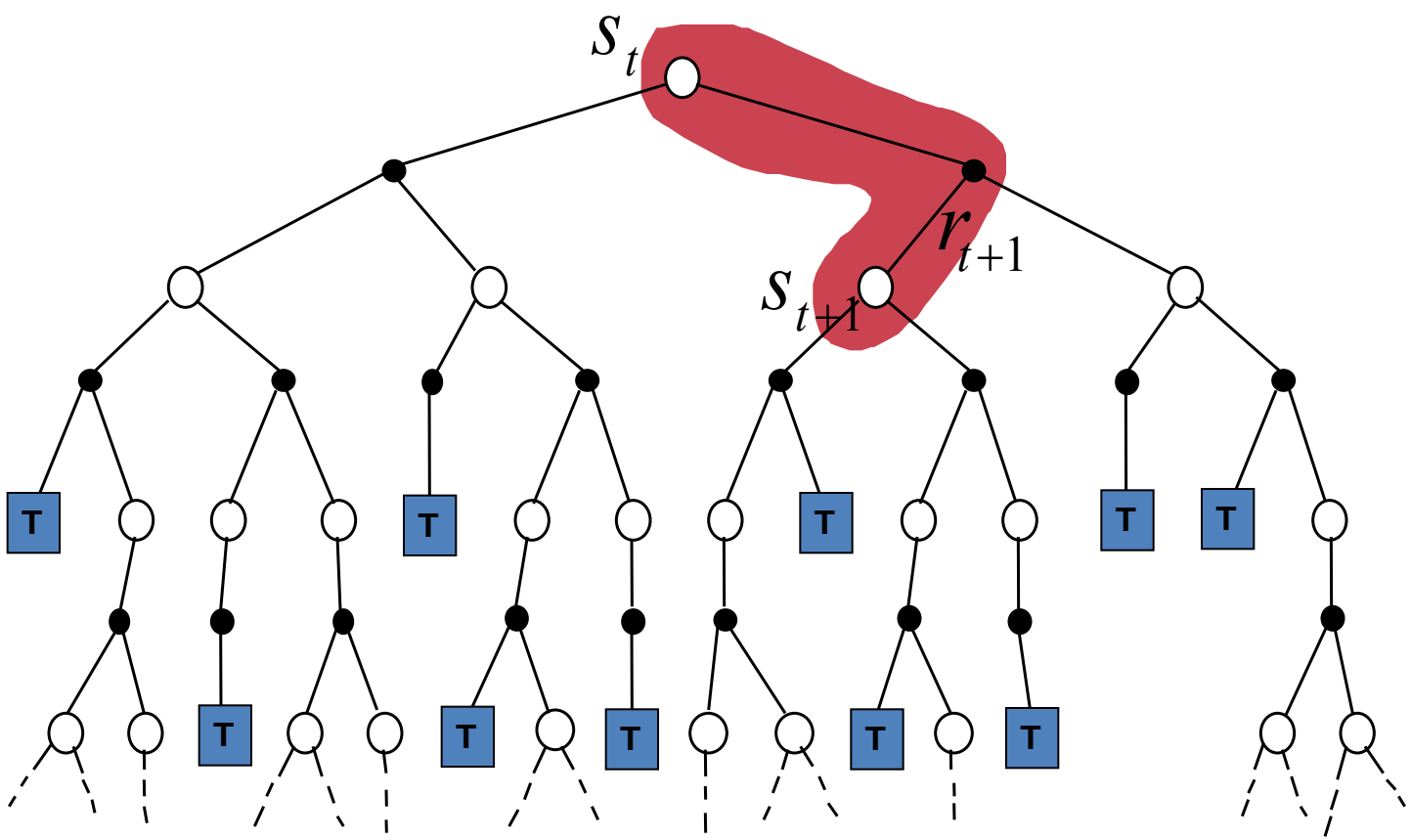
$$V(s_t) \leftarrow V(s_t) + \alpha\left[R_t - V(s_t)\right]$$

where R_t is the actual return following state s_t.



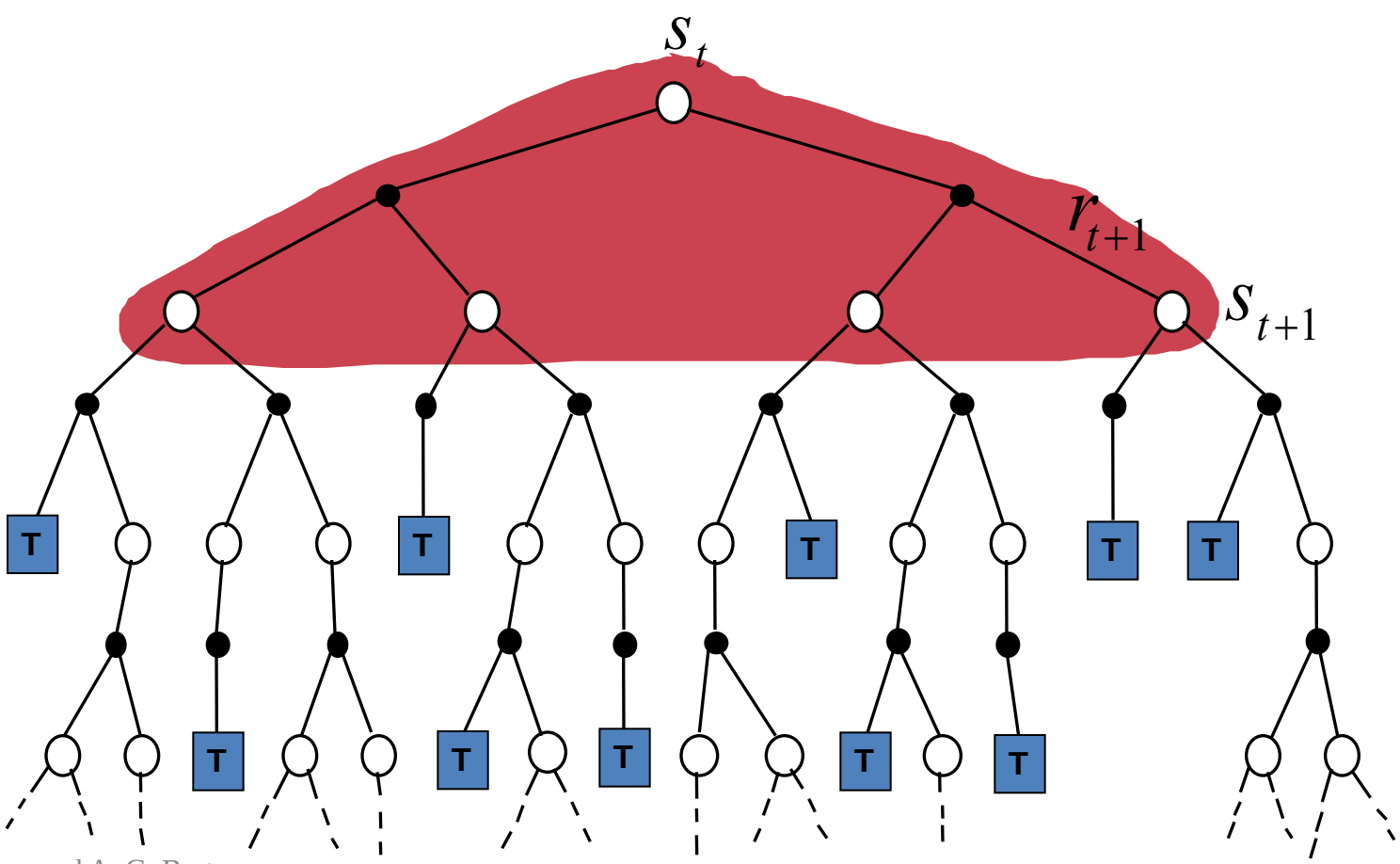
Simplest TD Method

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$



cf. Dynamic Programming

$$V(s_t) \leftarrow E_\pi \{ r_{t+1} + \gamma V(s_t) \}$$



Optimality of TD(0)

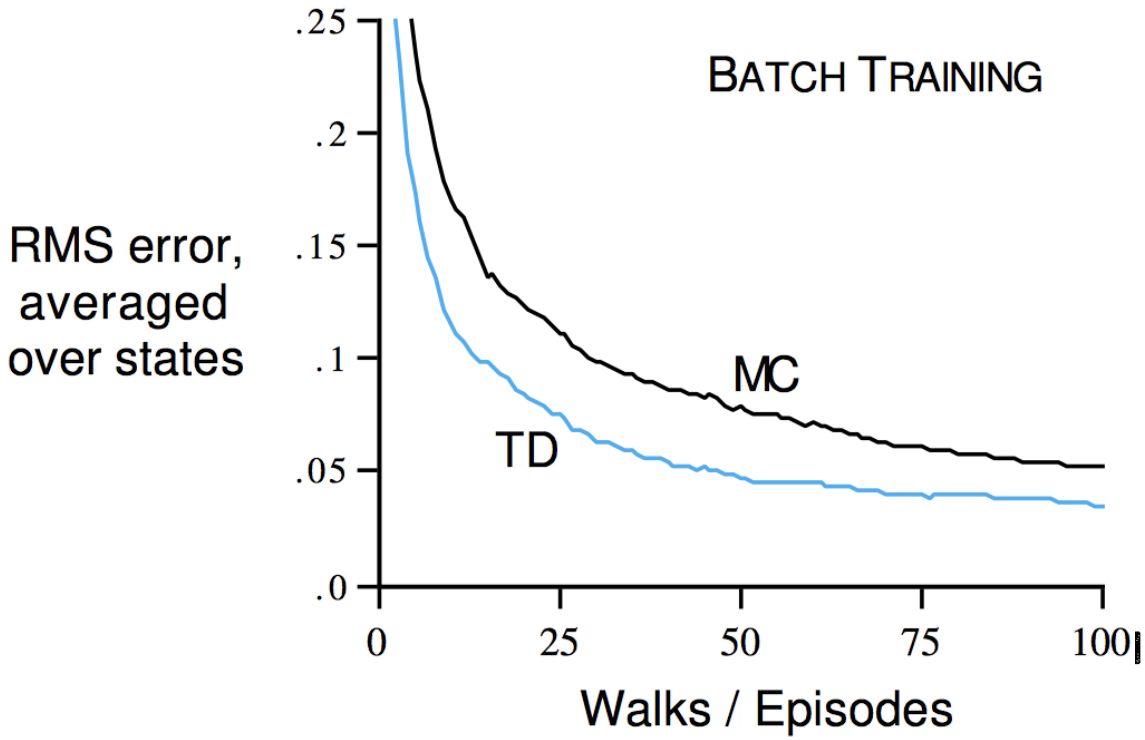
Batch Updating: train completely on a finite amount of data, e.g., train repeatedly on 10 episodes until convergence.

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

$V(\mathrm{A})?$

$V(\mathrm{B})?$

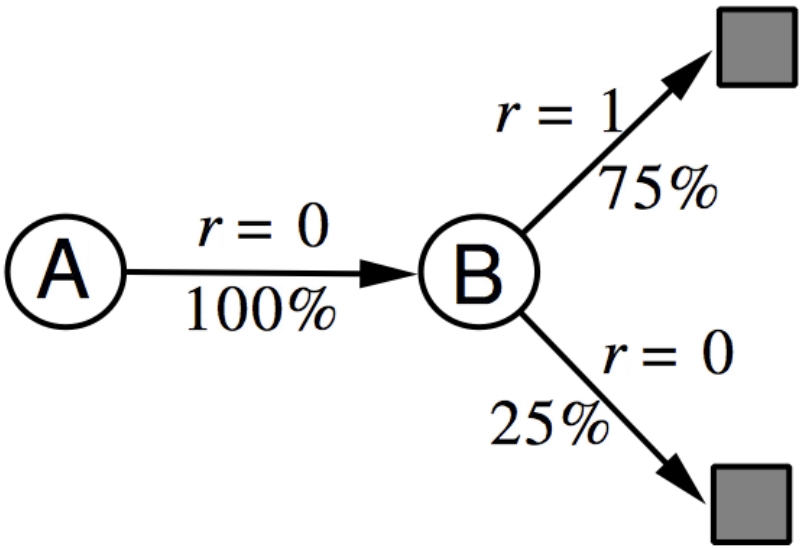
You are the Predictor

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

$V(\mathrm{A})?$

$V(\mathrm{B})?$



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
  - This is called the certainty-equivalence estimate
  - This is what TD(0) gets

What’s the right tool for the job?

Control vs. Evaluating Policies

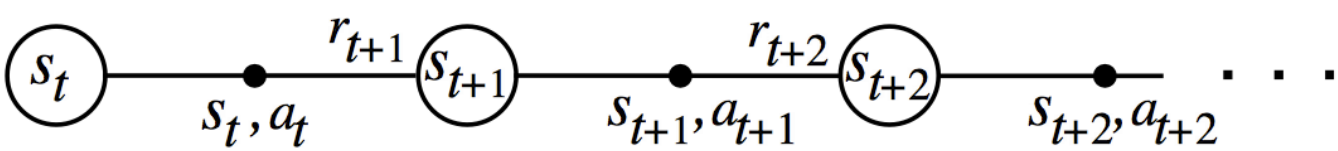
The simplest TD method, TD(0):

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

What if we wanted to learn Q? What would the updated be?

Learning An Action-Value Function

Estimate Q^{π} for the current behavior policy π.



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

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

If s_{t+1} is terminal, then $Q(s_{t+1}, a_{t+1}) = 0$.

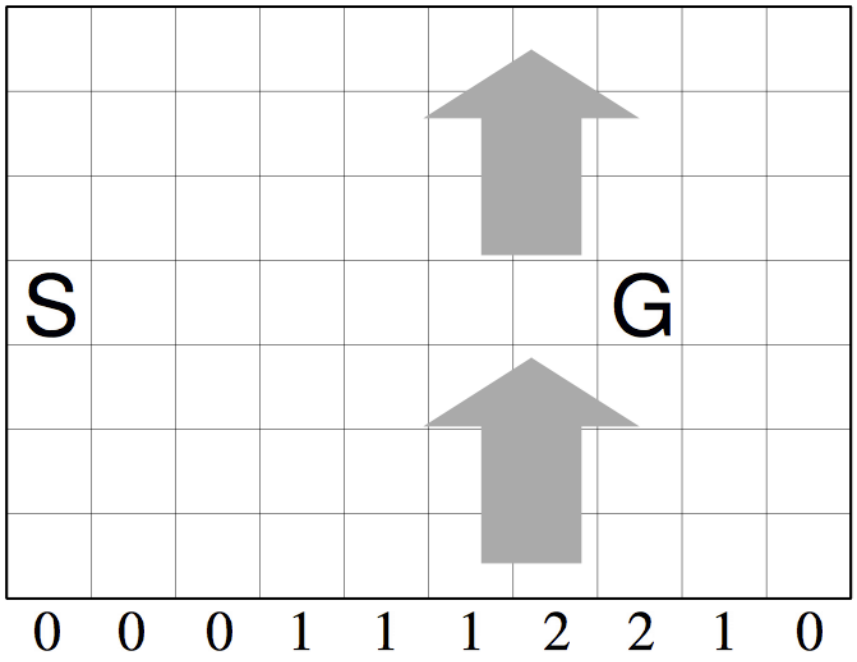
What would an algorithm look like?

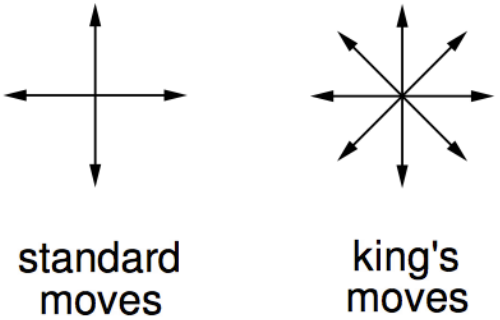
Sarsa: On-Policy TD Control

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

```
Initialize Q(s, a) arbitrarily
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) ← Q(s, a) + α[r + γQ(s', a') − Q(s, a)]
        s ← s'; a ← a';
    until s is terminal
```

Windy Gridworld

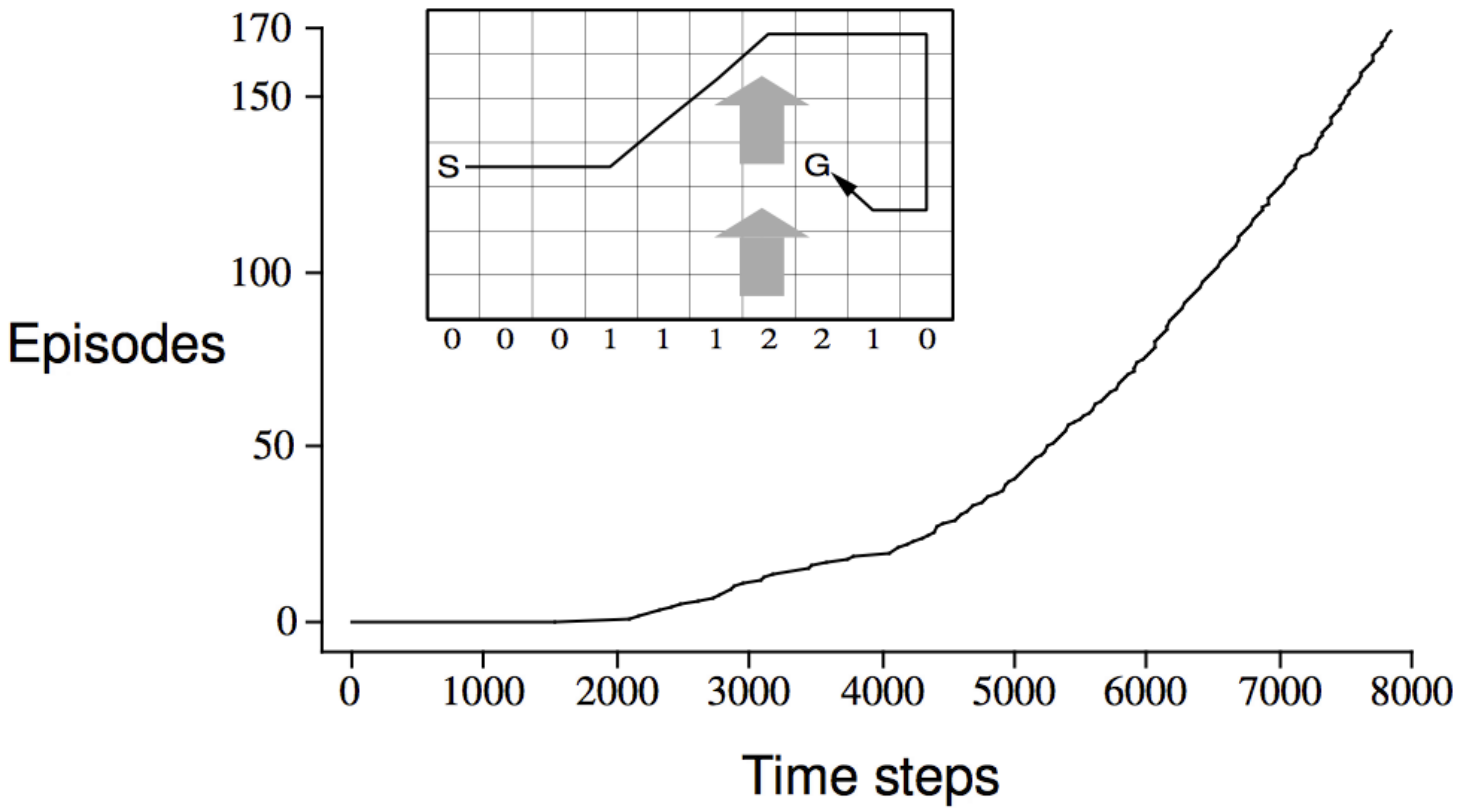




Undiscounted, episodic, reward = –1 until goal

What's the potential problem with MC, exploring starts?

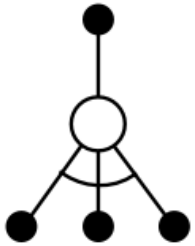
Results of Sarsa on the Windy Gridworld



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) arbitrarily
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) ← Q(s,a) + α[r + γ max_a' Q(s',a') − Q(s,a)]
        s ← s';
    until s is terminal
```