Advanced Topics in Reinforcement Learning

Lecture 10: Models and Planning I

Josiah Hanna
University of Wisconsin — Madison

Announcements

- Homework 2 due Thursday @ 9:29 AM
- Project proposals will be reviewed and feedback provided this week.
- Start reading chapter 9 for next week (Function Approximation).
- Office hours until 11:45 today.

This Week

- Model learning in RL.
- Planning and learning from simulated experience.
- Planning at decision-time.
- Next week:
    - We begin to discuss function approximation in RL.

Review

- Dynamic Programming Methods
  - Model is given.
- Monte Carlo methods.
  - Model-free but no bootstrapping.
- Temporal Difference Learning.
  - Model-free and bootstrapping.

Plan-Space Search

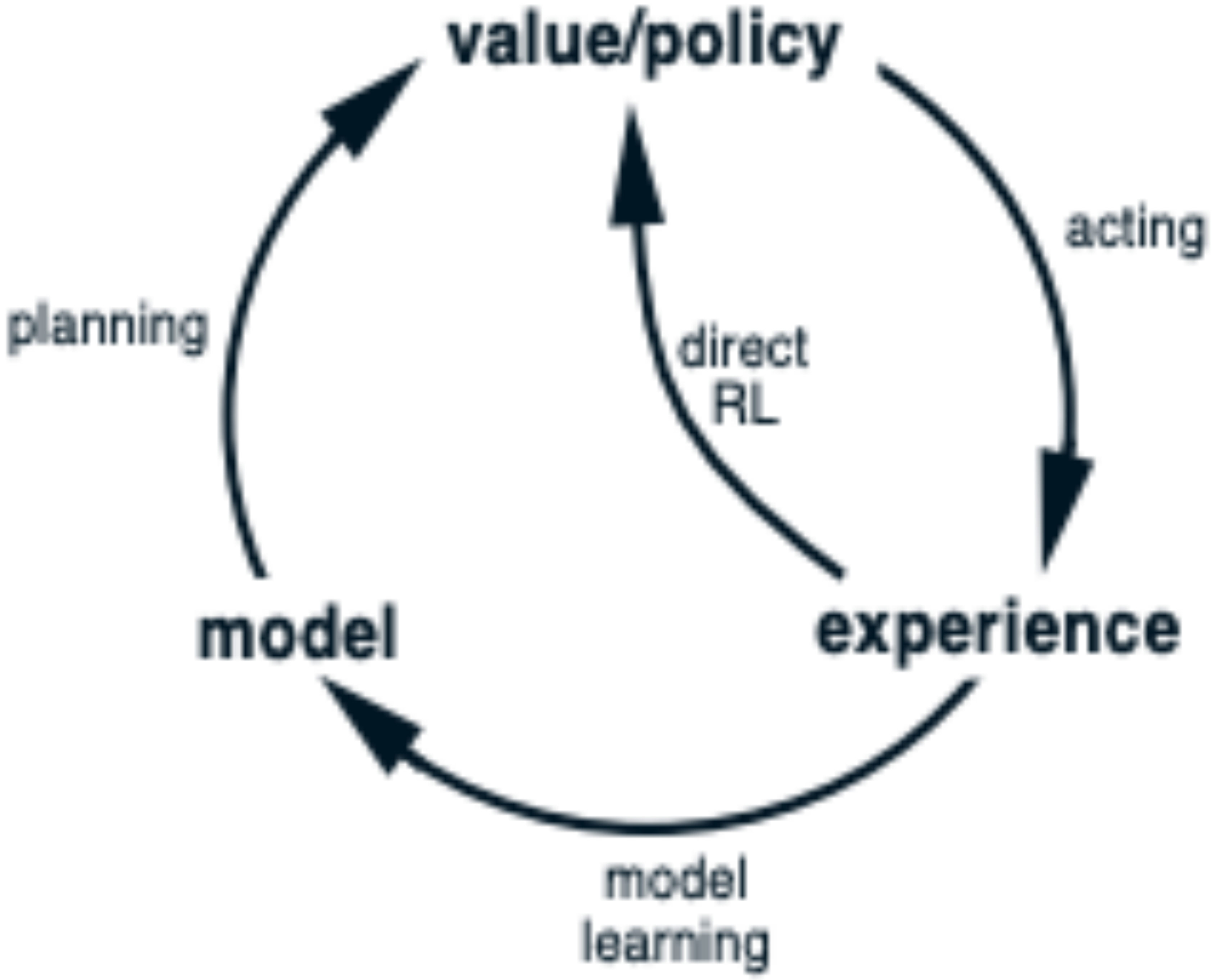
- The RL methods we will learn about in this class all follow the generalized policy iteration scheme.
  - $\pi_k \rightarrow q_k \rightarrow \pi_{k+1} \rightarrow q_{k+1} \rightarrow \cdots$
- Alternatively: search directly for a good policy without computing a value function.
  - Genetic algorithms, evolutionary strategies, random search, optimization, etc.
  - Define $f : \pi \rightarrow \mathbb{R}$ and then find π with maximal $f(\pi)$.
- (+) Robust to violations of MDP formalism.
- (+) Can be applied to almost any type of policy.
- (-) Size of policy space is exponential in the number of states and actions.
- (-) If interaction time is long and γ large, then learning methods may be preferred to static policies.

Plan-Space Search

RL with a Learned Model

- Model-based learning and planning.
  - Use experience to model p. Then use planning methods to back-up values.
  - How to model p is a challenging question in practice!
  - Often more data-efficient (better policy with less interaction). Why?
  - Less computationally-efficient per time-step. Why?
- Access to a model provides much flexibility in how we back-up values.

RL with a Learned Model



Cameron’s Presentation

Slides

Dyna Agents

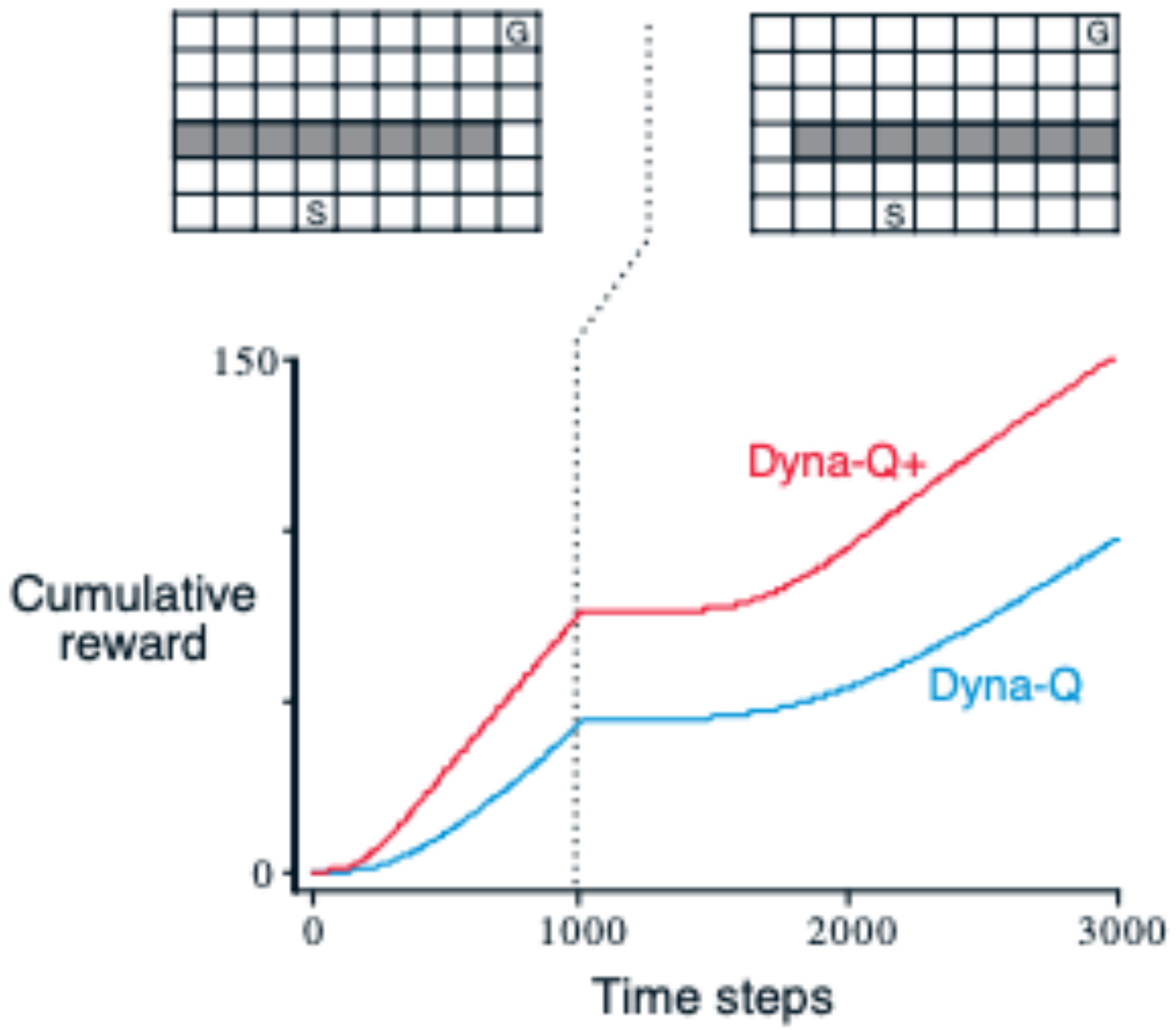
- In state S, take action A, observe R, and S'.
- Update model: Model(S, A) $\leftarrow R, S'$
- Repeat n times:
  - Sample random state-action pair, S, A.
  - $R, S' \leftarrow$ Model(S, A).
  - Apply q-learning update with S, A, R, S'.

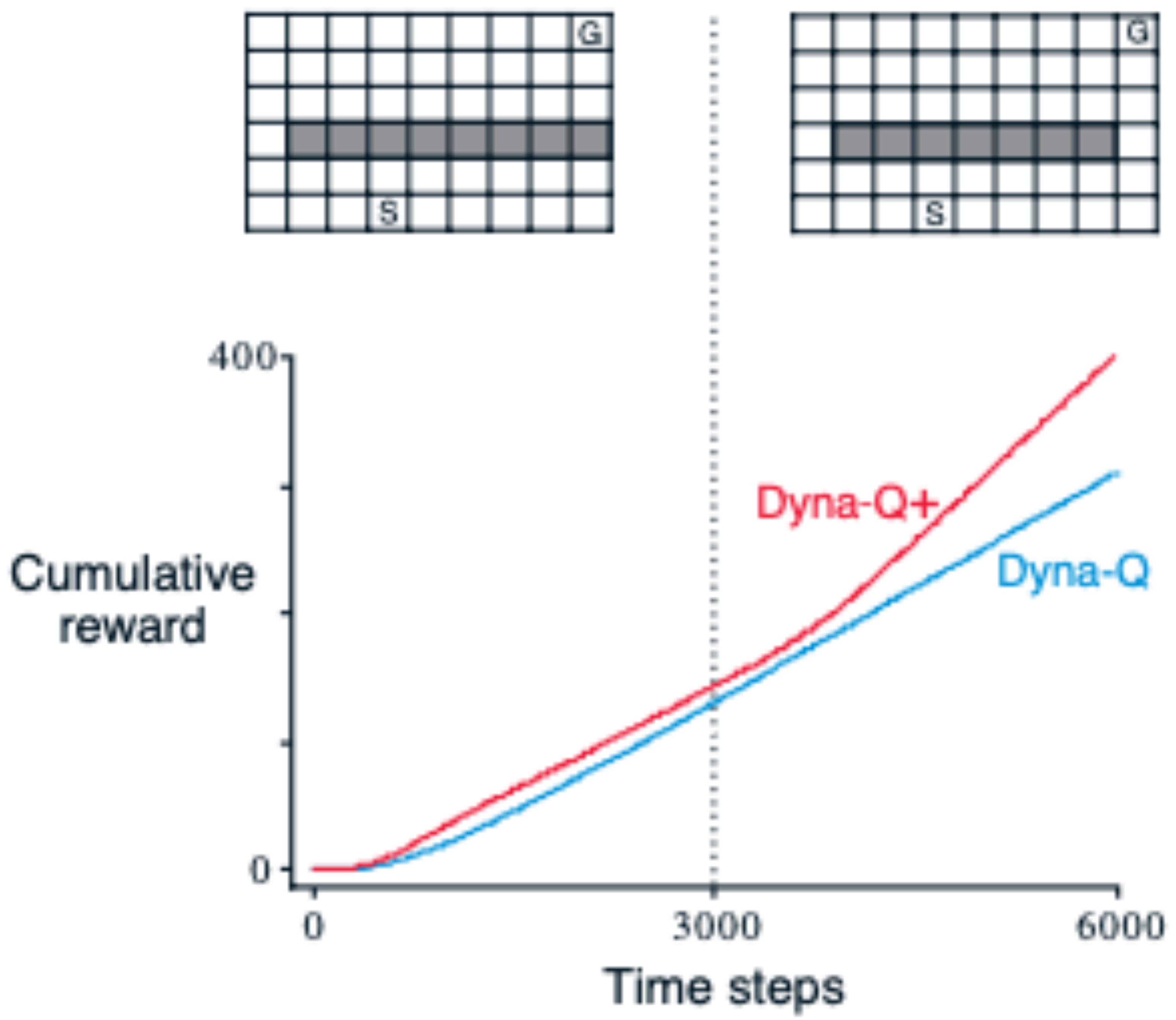
Real Experience

Synthetic / Simulated Experience

When the Model is Wrong

- “All models are wrong but some are useful.” - George Box.
- In practice, models can be inaccurate for many reasons.
  - Partial observability, non-stationarity, function approximation, missing data, etc.





Prioritizing Updates

- Dyna-Q updates a random sub-set of states.
- What is one case where this is inefficient?
  - When many states won't have a value change.

```
##################################################
# YOUR IMPLEMENTATION HERE                       #
# Choose state with the maximal value change potential  #
# Do a one-step lookahead to find the best action      #
# Update the value function. Ref: Sutton book eq. 4.10. #
##################################################
top_state = self.pq.pop()
action_values = self.one_step_lookahead(top_state)
self.V[top_state] = max(action_values)
# update the priority queue
for state in self.pred[top_state]:
    self.pq.update(
        state, -abs(self.V[state] - max(self.one_step_lookahead(state)))
    )
```

To Sample or Not?

- Dynamic programming typically uses *expected updates*.

$$Q(s,a) \leftarrow \sum_{s',r} \hat{p}(s',r|s,a)[r + \gamma \max_{a'} Q(s',a')]$$

- TD-method methods use *sample updates.*

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a')]$$

- Can use either with simulated experience. Which to choose?

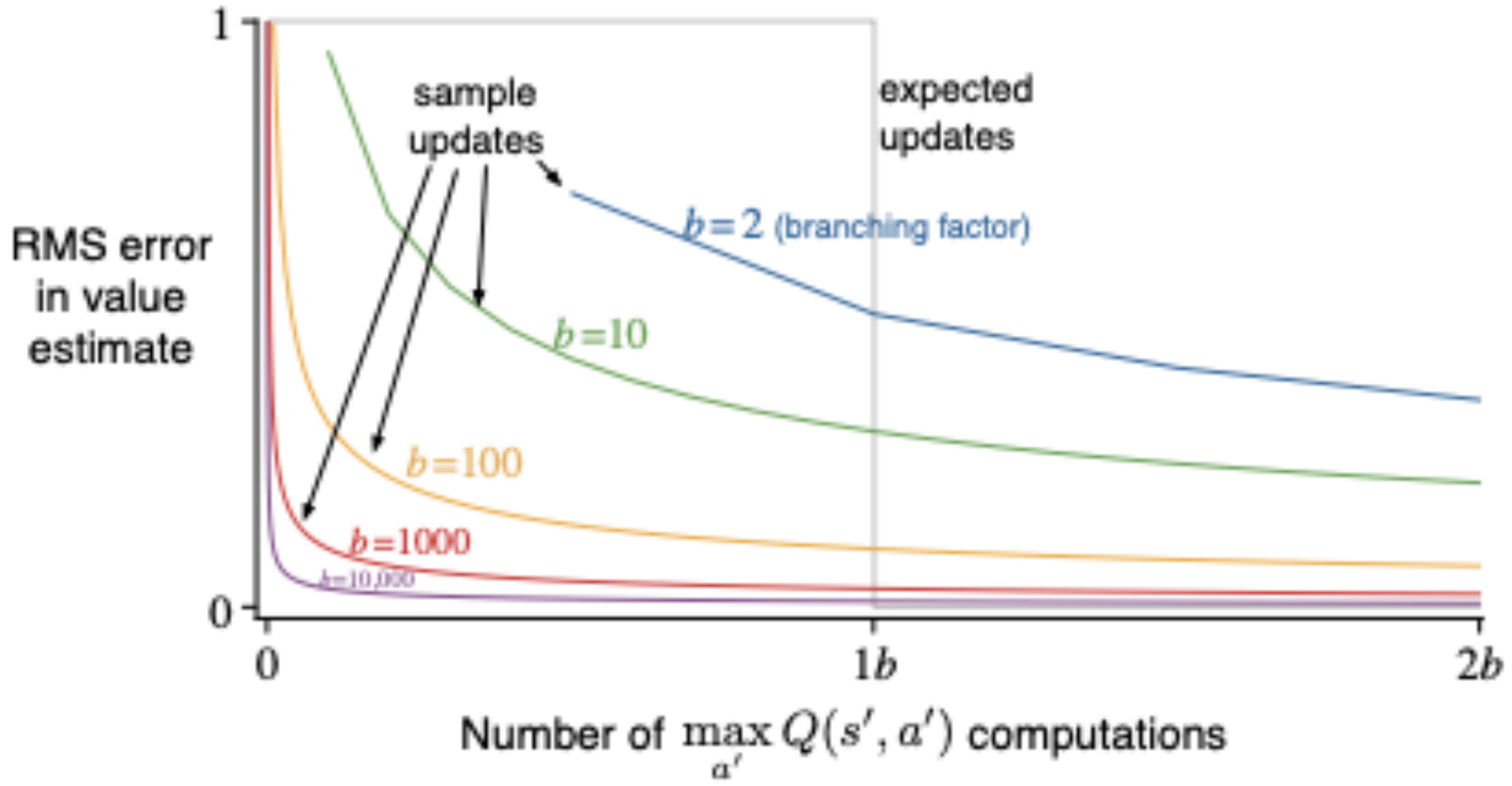


Figure 8.7: Comparison of efficiency of expected and sample updates.

Trajectory Sampling

- Uniform sampling of states can be inefficient.
- It may be more effective to focus value back-ups on states that the agent will visit often.
- How to know what states the agent will visit?
    - Simulate entire trajectories within the model. Back-up these states.
    - Initialize the agent in a start state and follow the current policy from there.

Real-time Dynamic Programming

- Key Idea: perform a value-iteration update on each state as it is visited.
- For n episodes:
    - Start in initial state, S_0.
    - Repeat $A_t \sim \pi(A = a \mid S_t), S', R \sim \texttt{Model}(S_t, A_t)$ where π is ϵ-greedy.
    - At each step, t, apply the value iteration update to $Q(S_t, A_t)$.

Summary

- Building a model from experience can improve the efficiency of RL.
- Models can be used for:
    - Planning, i.e., dynamic programming updates.
    - Learning with synthetic experience.
- When coupling planning and interaction, we need to make efficient use of limited computational resources.

Action Items

- Homework 2 due Thursday @ 9:29 am.
- Begin literature review
- Begin reading Chapter 9.