COSC 689: Deep Reinforcement Learning

Spring 2019

Prof. Grace Hui Yang
Department of Computer Science
Georgetown University

Logistics

- Midterm grades are out (grading; comments)
- Final Project Due: May 5th.
  - Be Active & Seek advice from me and the TA
  - Be resourceful
  - Optimize your own work routines
  - Read well, design well, code well, experiment sufficiently, write well
  - How to come out with a strong project?

Outline

- What is Model?
- Model-Based vs. Model-free
- What is Planning? (Control and Planning)
- Approaches
  - Optimal Control & Planning
    - MCTS
  - Model-Based RL Only
    - Global Models (Gaussian Processes)
    - Local Models
  - Model Based RL + Policy Learning
    - Dyna
    - Backpropagation through policy (PILCO)
    - Dagger-like (PLATO)
    - Controlled optimization (GPS)
  - Learning in Latent Space
    - POMDPs
- Summary

Today

- What is Model?
- Model-Based vs. Model-free
- What is Planning? (Control and Planning)
- Approaches
  - Optimal Control & Planning
    - MCTS
  - Model-Based RL Only
    - Global Models (Gaussian Processes)
    - Local Models
  - Model Based RL + Policy Learning
    - Dyna
    - Backpropagation through policy (PILCO)
    - Dagger-like (PLATO)
    - Controlled optimization (GPS)
  - Learning in Latent Space
    - POMDPs
- Summary

Next Week

- What is Model?
- Model-Based vs. Model-free
- What is Planning? (Control and Planning)
- Approaches
  - Optimal Control & Planning
    - MCTS
  - Model-Based RL Only
    - Global Models (Gaussian Processes)
    - Local Models
  - Model Based RL + Policy Learning
    - Dyna
    - Backpropagation through policy (PILCO)
    - Dagger-like (PLATO)
    - Controlled optimization (GPS)
  - Learning in Latent Space
    - POMDPs
- Summary

What is “Model”?

- We use the word “model” very often in SML/RL
- The word “model” in “Model-based RL” specially means
  - the “transition” model — from one state to the next state:

    deterministic case: $s' = f(s, a)$

    stochastic case: $p(s' \mid s, a)$
- That an agent can use it to predict how the environment would respond to the agent’s actions.
- Note: the textbook also mentions that Model-based RL would learn the “reward” function (reward the agent will receive given state and action), too. However in contemporary DRL, reward function is learned via inverse RL (a topic we won’t cover in this term).

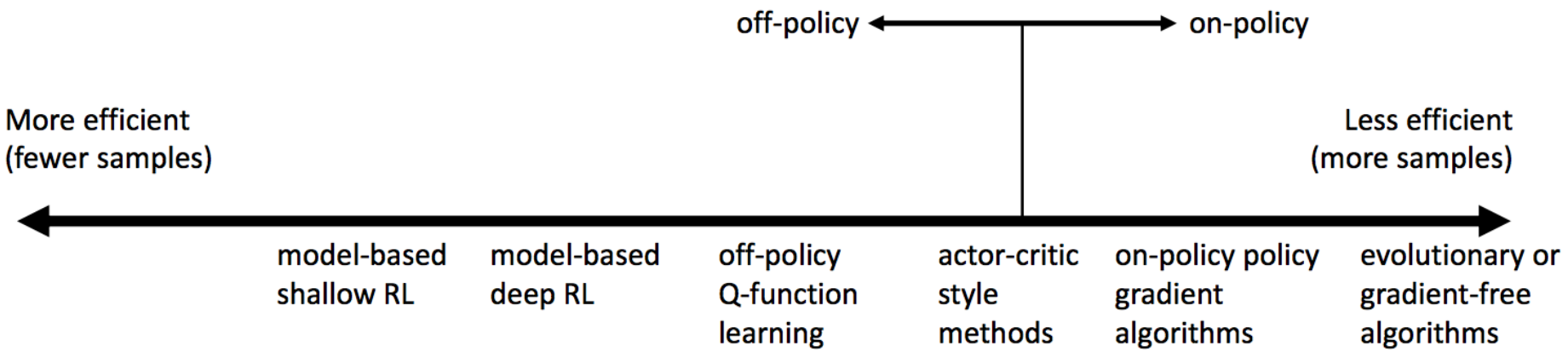
Model-Based vs. Model-Free

- **Model-Based:**
  - Dynamic Programming
  - Heuristic Search
- Do a lot of planning (by controlling the system dynamics via manipulating the transition rules/probabilities)
- We can manipulate the model
  - Inject prior knowledge or heuristics

- **Model-Free:**
  - Monte-Carlo Methods
  - TD Methods
  - Gradient-Based Methods
- Mostly learning from data

Efficiency (Speed)

Comparison: sample efficiency



From Sergey Levine's CS CS 294-112

Which RL Algorithm to Use?

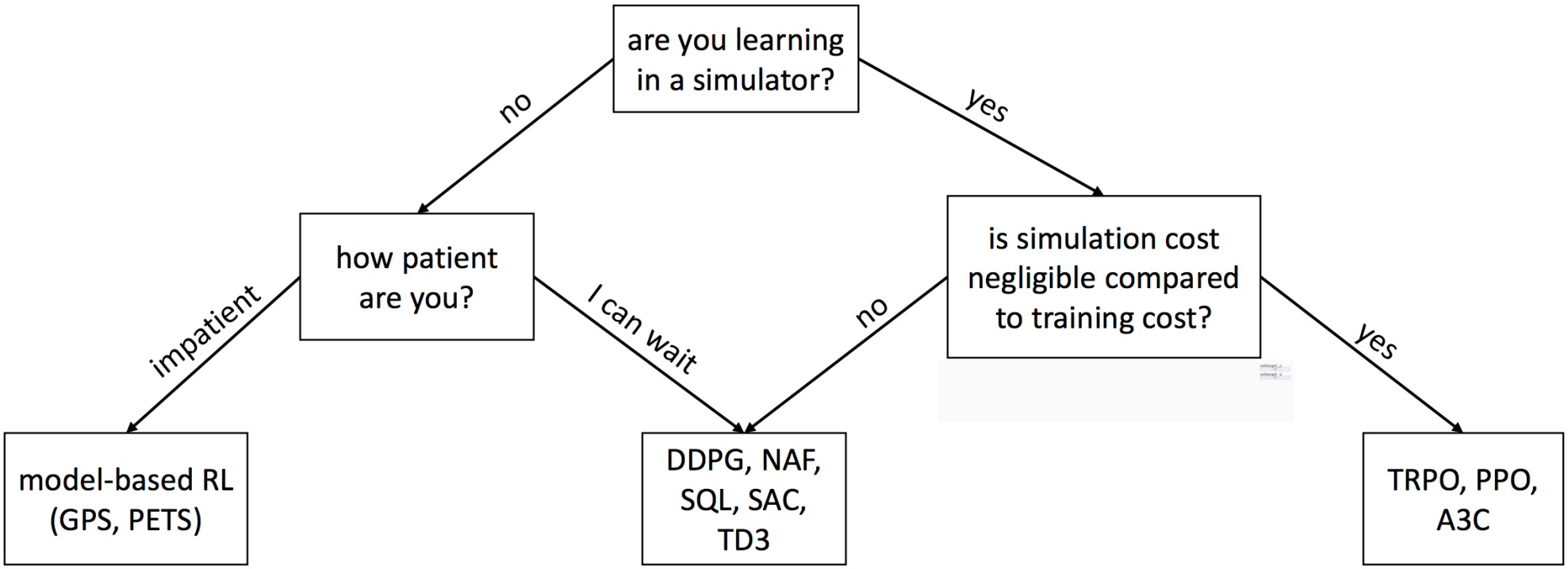
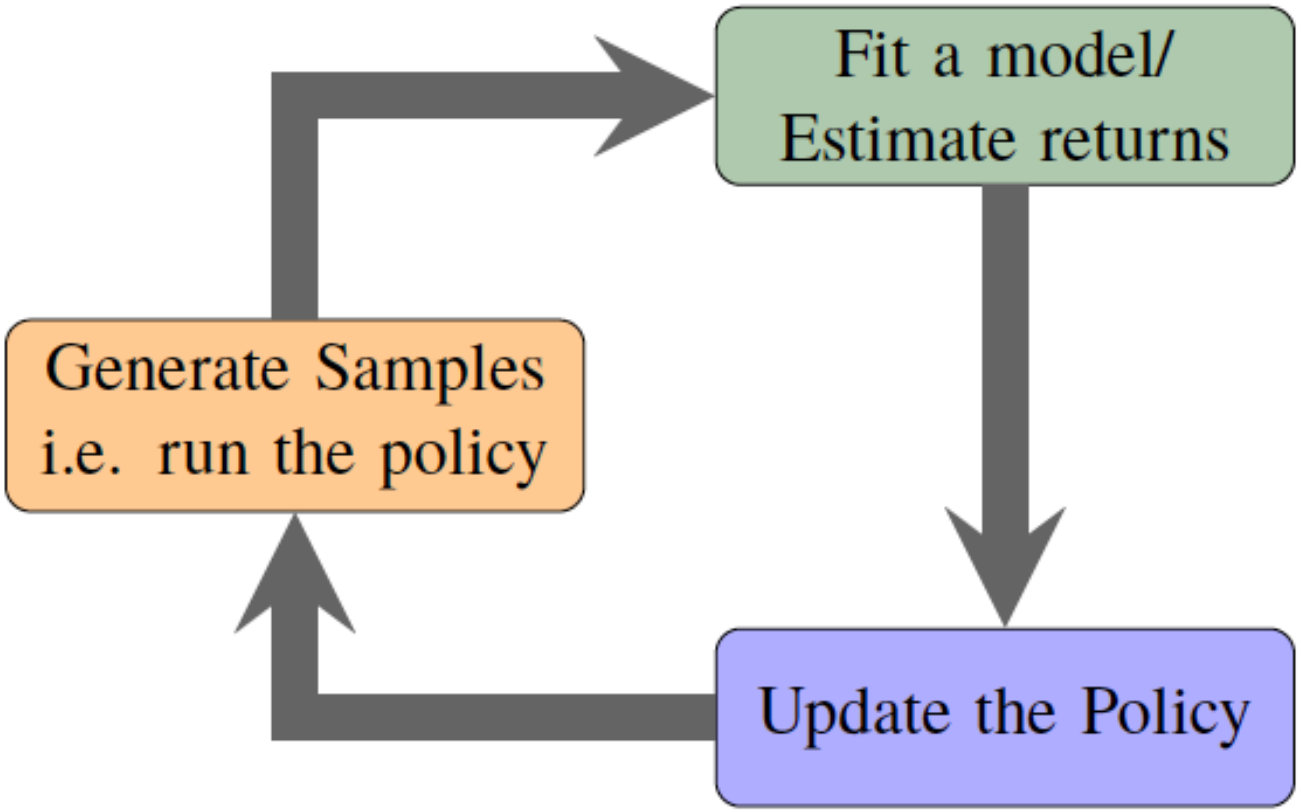


Diagram from Sergey Levine's CS CS 294-112

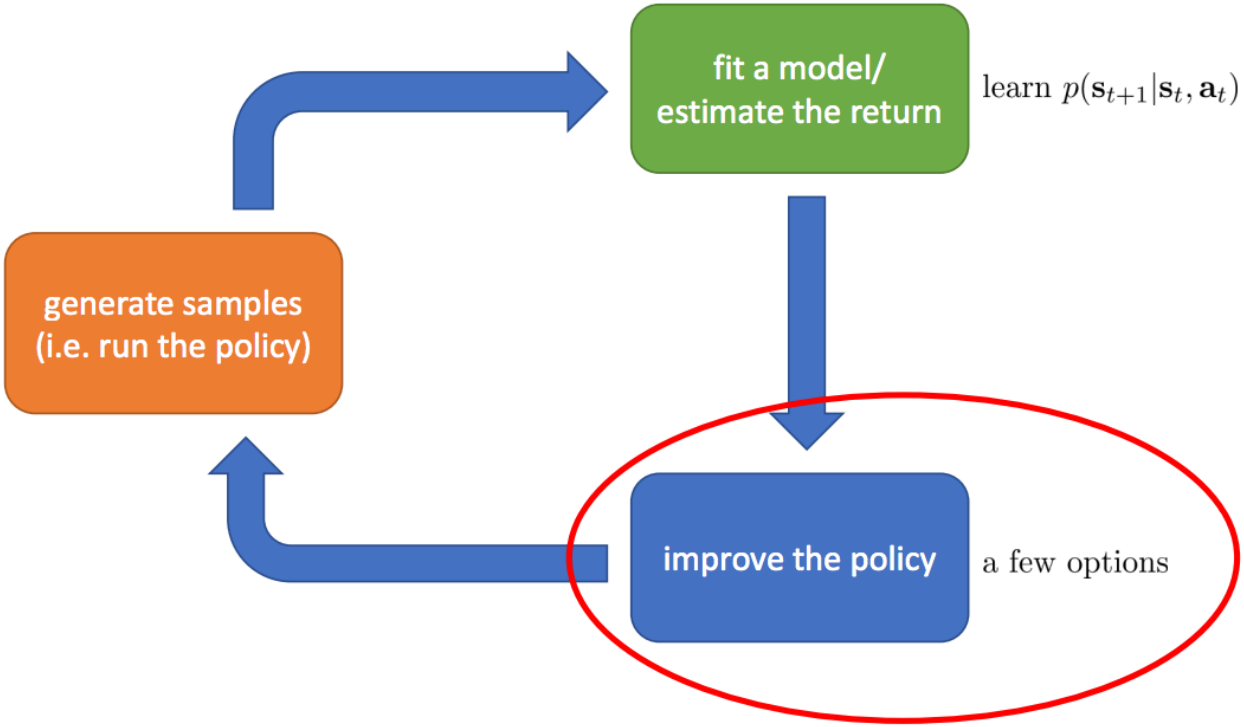
What's the use of Model?

- To directly decide which action to use (No policy)
  - planning by search in discrete action space
  - optimal control by action/trajectory optimization (mainly in continuous spaces)
- To simulate experiences
  - and feed them as training data to (model-free) policy learners or to value-function learners
- Can help policy learner by back-propagating through the gradients

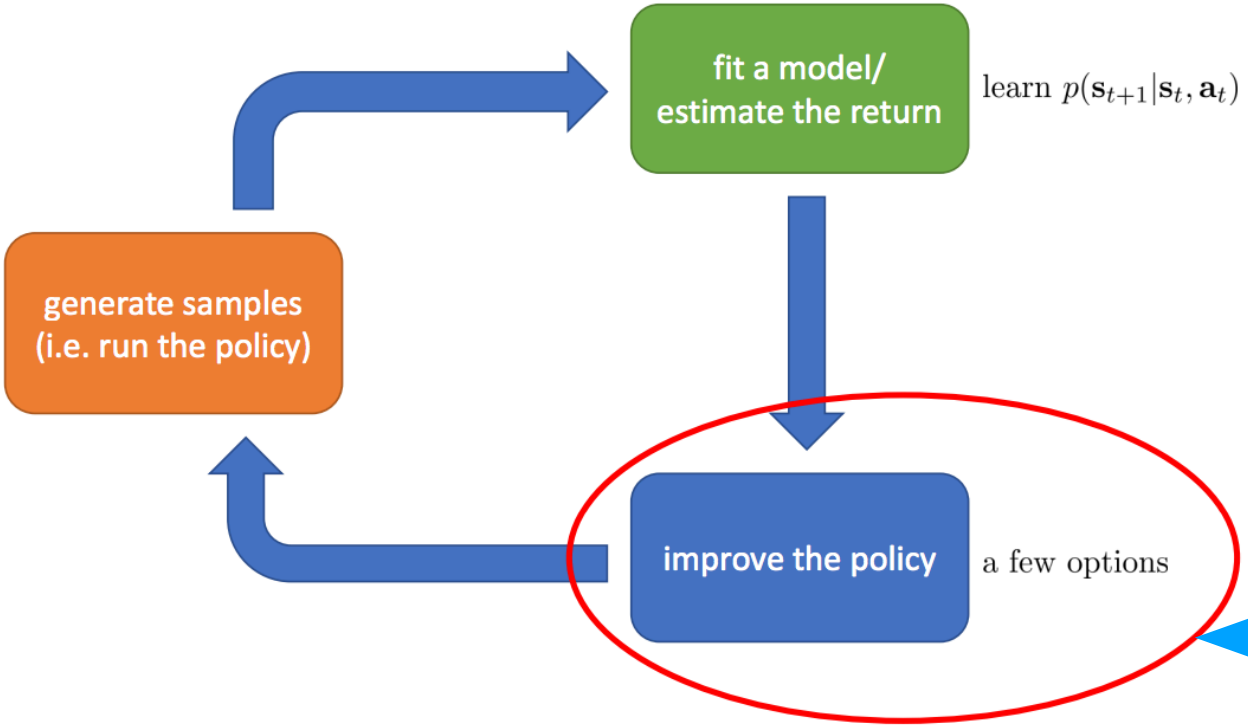
Anatomy of RL



Model-Based RL



Model-Based RL



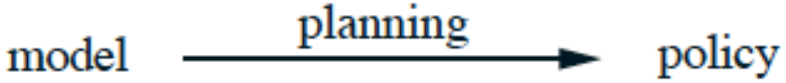
- Planning
- Simulate more experiences
- Backpropagate gradients & help w/ policy

How we can make use of experiences?

- Within a planning agent, there are at least two roles for real experience:
  - It can be used to improve the model (to make it more accurately match the real environment) — model learning
  - It can be used to directly improve the value function and policy — those RL methods that we have seen so far
  - It can be used to improve value functions and policies indirectly via the model —planning

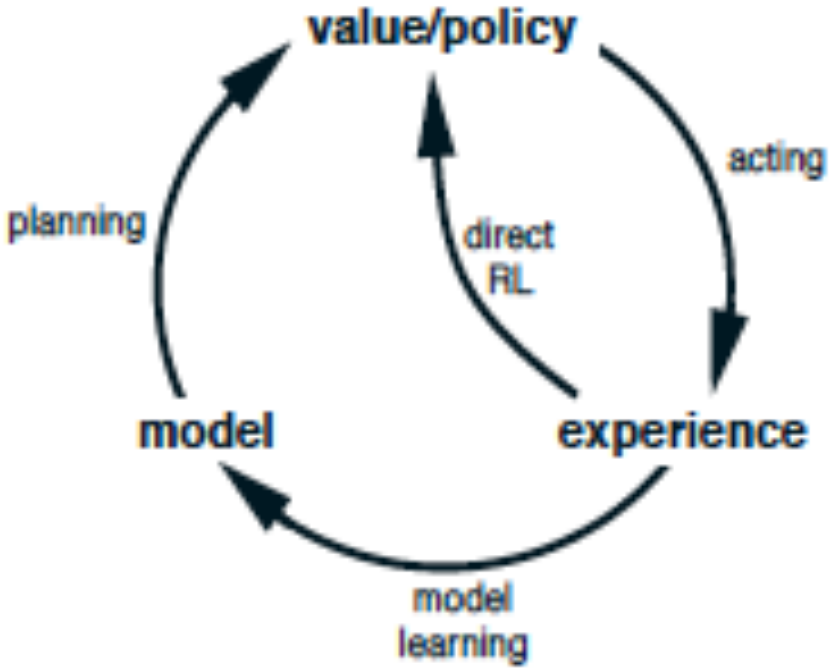
What is Planning?

- Something to make the policy better, based on a model

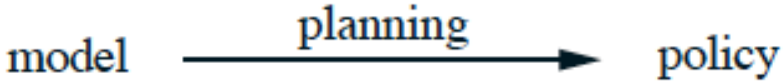


- It would change the system by influence either V or policy (action) before the system moves on to next time step
- It won't directly generate policy; instead, affect policy learning indirectly
- It is indirect RL.

Planning vs. Learning



What is Planning?



- State-space planning
  - A search through the state space for an optimal policy or an optimal path to a goal
  - Majority of RL planning is about it
- Plan-space planning
  - A search through the space of plans (Remember GEP-PG?)
  - Quite difficult to be applied to stochastic sequential decision making (the main focus of RL)
  - Not popular

Q-Planning

Random-sample one-step tabular Q-planning

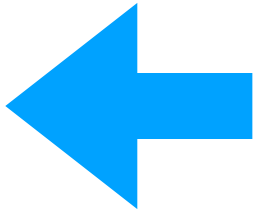
Loop forever:

1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(S)$, at random
2. Send S, A to a sample model, and obtain
   a sample next reward, R, and a sample next state, S'
3. Apply one-step tabular Q-learning to S, A, R, S':
   $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

What is Planning?

- Background Model planning
  - To gradually improve a policy or value function on the basis of simulated experience obtained from a model (either a sample or a distribution model).
  - Selecting actions is then a matter of comparing the current state's action values obtained from a table
    - Or, by evaluating a mathematical expression in the approximate methods
  - Planning is not focused on the current state
- Decision-time Planning
  - To begin and complete it after encountering each new state s, as a computation whose output is the selection of a single action a; on the next step planning begins with s’ to produce a’, and so on.
  - They include heuristic search, and rollout algorithms like MCTS

Approaches Overview



- Optimal Control & Planning
  - MCTS
- Model-Based RL Only
  - Global Models (Gaussian Processes)
  - Local Models
- Model Based RL + Policy Learning
  - Dyna
  - Backpropagation through policy (PILCO)
  - Dagger-like (PLATO)
  - Controlled optimization (GPS)
- Learning in Latent Space
  - POMDPs

Model-Based RL

- It would learn the transition dynamics, and then figure out how to make use of it to choose actions

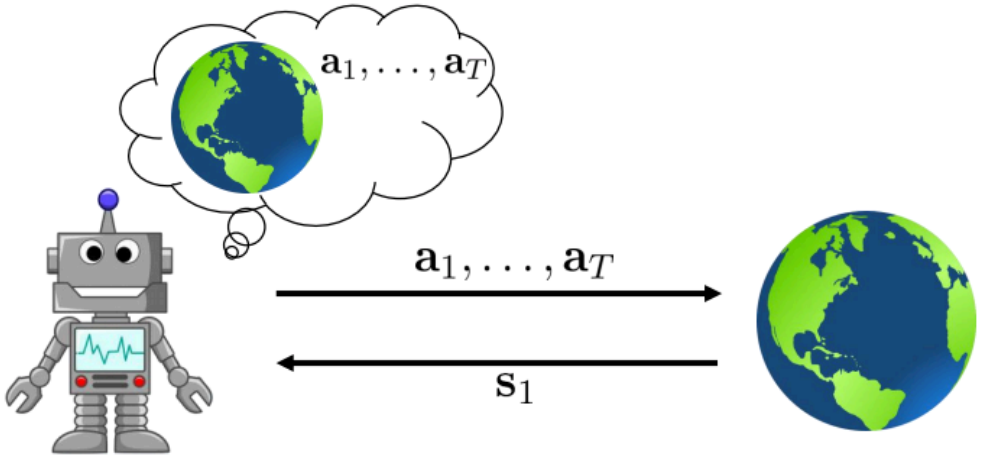
Planning with Known Dynamics

- What if we've already known dynamics?
  - e.g. the game of Go
  - e.g. in a simulated environments (like video games)
  - e.g. very simple models (car navigation)
- Then, we should go ahead make use them
- Let's see how to do planning with known dynamics

Task of Planning

- When the dynamics are known, the task of planning is reduced to
  - How to choose a series of actions, given all the known stuff

Deterministic Case



- After being at s1 and get reposes from the environment, how should the agent to act in its most optimal way?

$$a_1, a_2, \ldots, a_T = \arg \max_{a_1, a_2, \ldots, a_T} \sum_{t=1}^{T} r(s_t, a_t)$$

s.t. $a_{t+1} = f(s_t, a_t)$

Diagram from Sergey Levine's CS CS 294-112

Heuristic Search

- For each state encountered, a large tree of possible continuations is considered
- The approximate value function is applied to the leaf nodes and then backed up toward the current state at the root.
- Once the backed-up values of these nodes are computed, the best of them is chosen as the current action, and then all backed-up values are discarded.
- Usually look ahead for 1-step

Heuristic Search

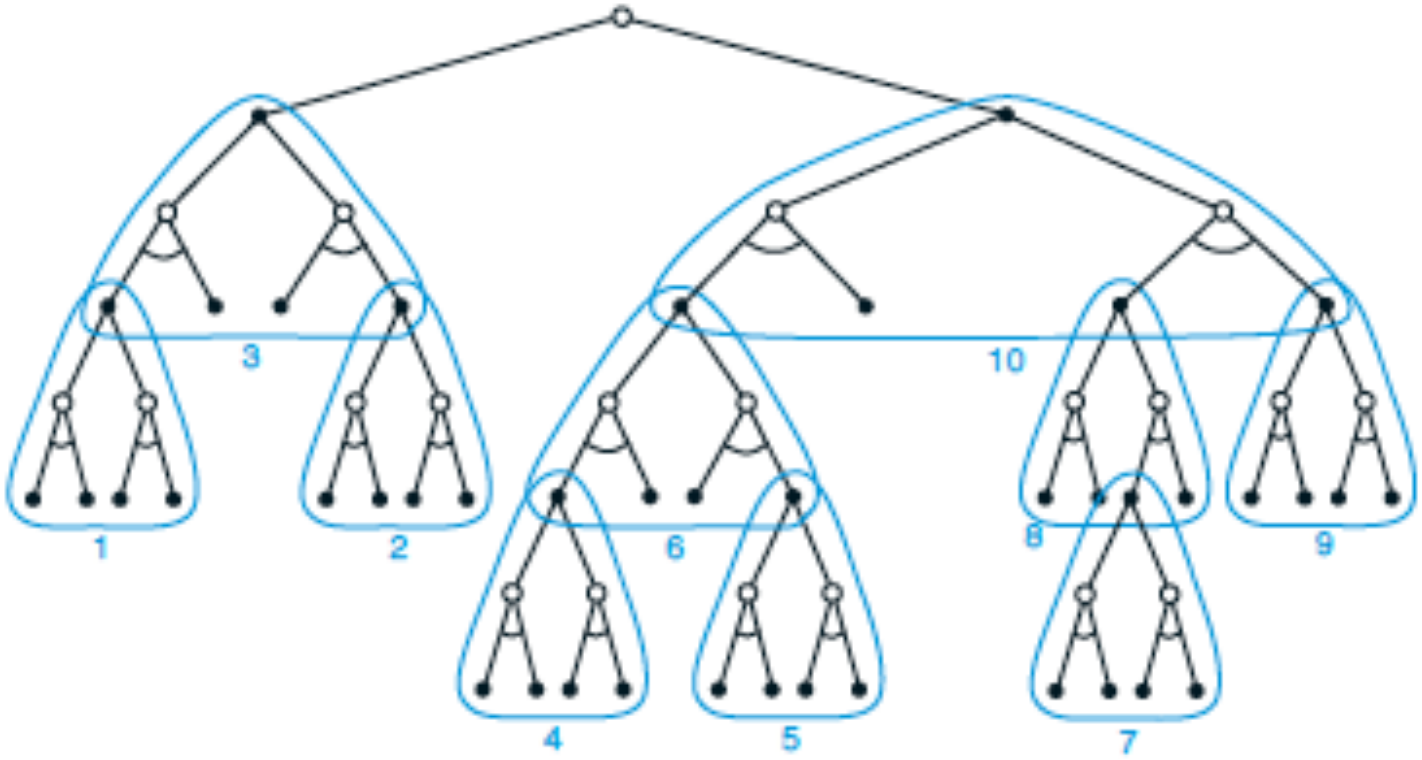


Figure 8.9: Heuristic search can be implemented as a sequence of one-step updates (shown here outlined in blue) backing up values from the leaf nodes toward the root. The ordering shown here is for a selective depth-first search.

Rollout Algorithms

- Rollout from the current state:
  - simulated trajectories that all begin at the current environment state
- MCTS is a rollout algorithm
  - Used in AlphaGo

Monte Carlo Tree Search (MCTS)

- Discrete planning as a search
- Four steps are applied per search iteration:

1. Selection: Starting at the root node, a child selection policy is recursively applied to descend through the tree until the most urgent expandable node is reached. A node is expandable if it represents a nonterminal state and has unvisited (i.e. unexpanded) children

2. Expansion: One (or more) child nodes are added to expand the tree, according to the available actions.

3. Simulation: A simulation is run from the new node(s) according to the default policy to produce an outcome.

4. Backpropagation: The simulation result is “backed up” (i.e. backpropagated) through the selected nodes to update their statistics.

MCTS

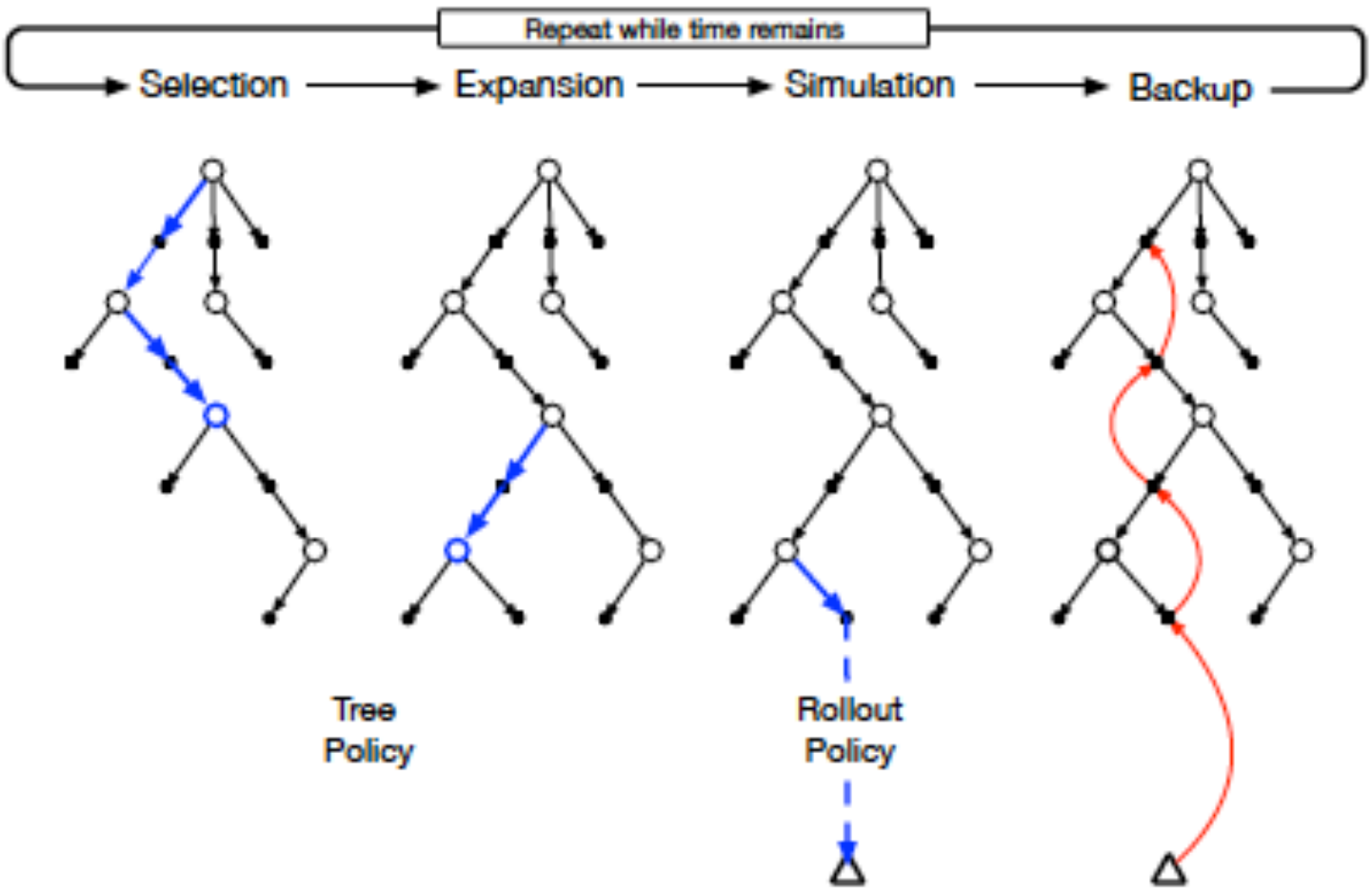


Figure 8.10: Monte Carlo Tree Search. When the environment changes to a new state, MCTS executes as many iterations as possible before an action needs to be selected, incrementally building a tree whose root node represents the current state. Each iteration consists of the four operations **Selection**, **Expansion** (though possibly skipped on some iterations), **Simulation**, and **Backup**, as explained in the text and illustrated by the bold arrows in the trees. Adapted from Chaslot, Bakkes, Szita, and Spronck (2008).

UCB is used in MCTS

- Intuition:
  - Cannot search all paths; then where to start first?
  - choose nodes that are less visited before, but with good rewards
- It's used as a TreePolicy during selection (the first step): choose a child with the best

$$\text{Score}(s_t) = \frac{Q(s_t)}{N(s_t)} + 2C\sqrt{\frac{2\ln N(s_{t-1})}{N(s_t)}}$$

Stochastic Case

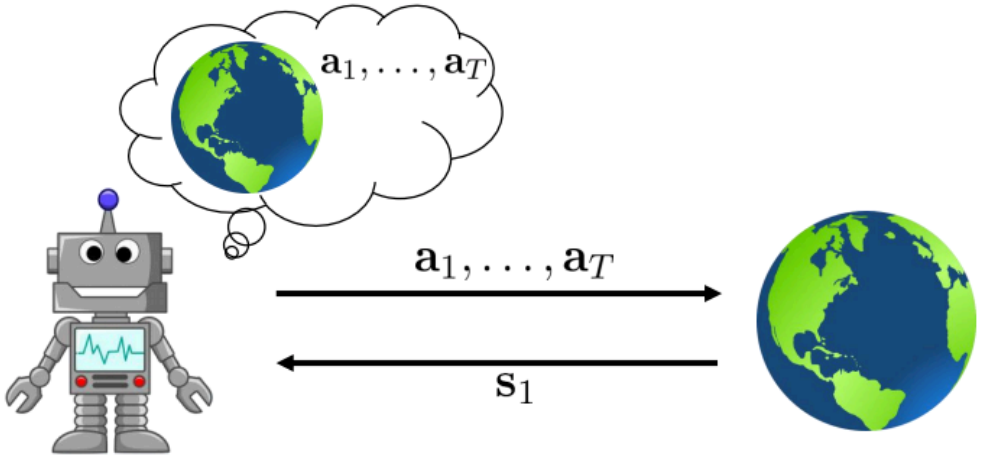


Diagram from Sergey Levine's CS CS 294-112

- After being at s1 and get reposes from the environment, how should the agent to act in its most optimal way?

$$a_1, a_2, \ldots, a_T = \arg \max_{a_1, a_2, \ldots, a_T} E[\sum_{t=1}^{T} r(s_t, a_t) \,|\, a_1, a_2, \ldots, a_T]$$

where $p_\theta(s_1, s_2, \ldots, s_T \,|\, a_1, a_2, \ldots, a_T) = p(s_1) \prod_{t=1}^{T} p(s_{t+1} \,|\, s_t, a_t)$

Recall: the complete RL system is: $p_\theta(s_1, s_2, \ldots, s_T \,|\, a_1, a_2, \ldots, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t \,|\, s_t) p(s_{t+1} \,|\, s_t, a_t)$

Random Shooting

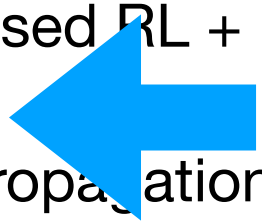
- It is the simplest method
- Basically, it says guess and check
- Steps:
  - Sample actions a1, a2, … an from some distribution J (e.g. uniform, gaussian…)
  - choose a_i based on $a_i = \arg\max_i J(a_i)$

Pros and Cons

- Pros:
  - Very fast (especially paralleled)
  - Very simple
- Cons:
  - Would be easily suffered from the curse of dimensionality
  - Limited applications

Approaches Overview

- Optimal Control & Planning
  - MCTS
- Model-Based RL Only
  - Global Models (Gaussian Processes)
  - Local Models
- Model Based RL + Policy Learning
  - Dyna
  - Backpropagation through policy (PILCO)
  - Dagger-like (PLATO)
  - Controlled optimization (GPS)
- Learning in Latent Space
  - POMDPs



Dyna

- Integrate Planning and Learning
- Online Q-Learning (which is model-free) with a model added now

Dyna

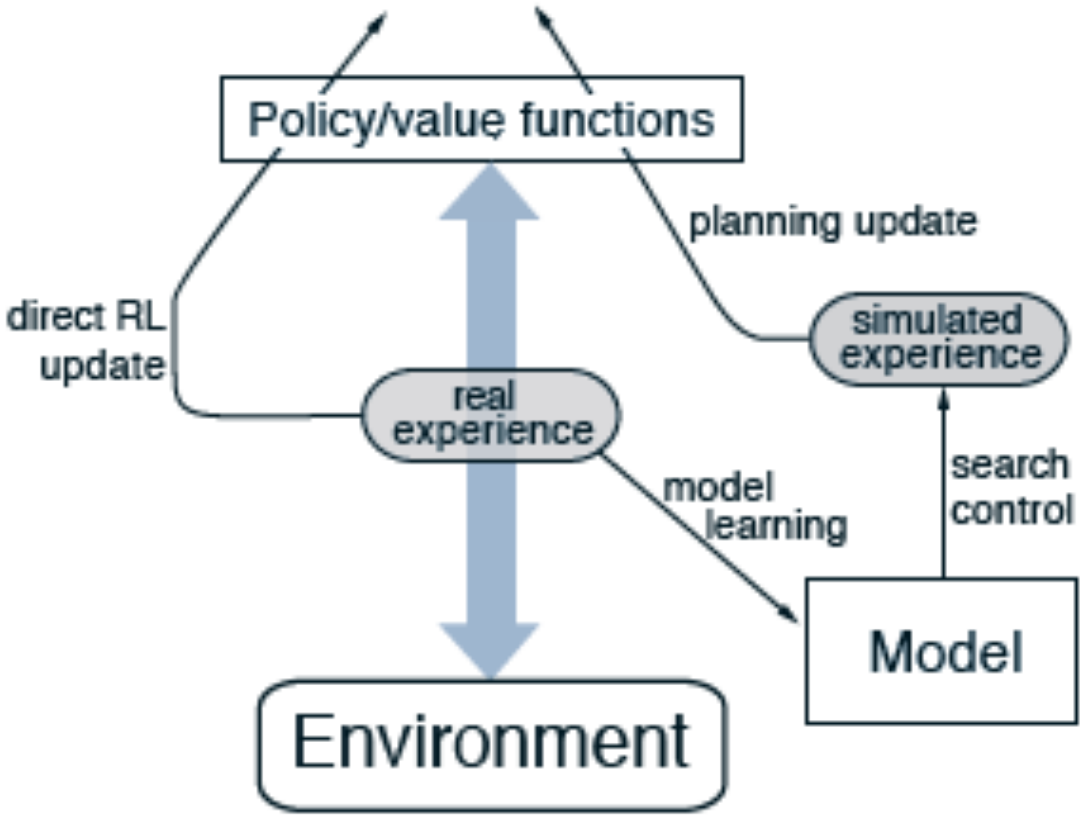


Figure 8.1: The general Dyna Architecture. Real experience, passing back and forth between the environment and the policy, affects policy and value functions in much the same way as does simulated experience generated by the model of the environment.

Dyna-Q

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:

(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Loop repeat n times:

$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

General Dyna-style RL

1. Collect some data, including transition dynamics: (s,a,s',r)
2. Learn the transition model (and optionally the reward model $r(s,a)$ too): $p(s'|s,a)$
3. Repeat K times:
   - 3.1. sample $s \sim B$ from a buffer
   - 3.2. choose action a (from B, from π , or random)
   - 3.3. simulate $s' \sim p(s'|s,a)$ (optionally reward model $r(s,a)$)
   - 3.4. train the policy π (or V, or Q) on (s,a,s',r)
   - 3.5. take N more model-based steps

Prioritized Sweeping

- Planning can be much more efficient if simulated transitions and updates are focused on particular state-action pairs
- It is natural to prioritize the updates according to a measure of their urgency, and perform them in order of priority
  - e.g. the measure could be the size of TD error

Prioritized Sweeping

Prioritized sweeping for a deterministic environment

Initialize $Q(s,a)$, $Model(s,a)$, for all s,a, and $PQueue$ to empty
Loop forever:
- (a) $S \leftarrow$ current (nonterminal) state
- (b) $A \leftarrow policy(S,Q)$
- (c) Take action A; observe resultant reward, R, and state, S'
- (d) $Model(S,A) \leftarrow R, S'$
- (e) $P \leftarrow |R + \gamma \max_a Q(S',a) - Q(S,A)|$.
- (f) if $P > \theta$, then insert S,A into $PQueue$ with priority P
- (g) Loop repeat n times, while $PQueue$ is not empty:
  - $S,A \leftarrow first(PQueue)$
  - $R,S' \leftarrow Model(S,A)$
  - $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
  - Loop for all $\bar{S},\bar{A}$ predicted to lead to S:
    - $\bar{R} \leftarrow$ predicted reward for $\bar{S},\bar{A},S$
    - $P \leftarrow |\bar{R} + \gamma \max_a Q(S,a) - Q(\bar{S},\bar{A})|$.
    - if $P > \theta$ then insert $\bar{S},\bar{A}$ into $PQueue$ with priority P

Summary

- Planning is indirect RL, which relies on a model
- When the dynamics (transition functions) are known, we can do optimal control and planning directly
    - MCTS is discrete, online planning with known dynamics
- When the dynamics are unknown, we need to learn them and then use them
    - Dyna integrates model learning, planning, and policy learning together
        - It can be used in combination with a variety of RL methods (e.g. Dyna-Q)
- Next class we will see more on learning the unknown dynamics or learning dynamics and policy together

References

- RL Textbook Chapter 8.
- MCTS: http://mcts.ai/pubs/mcts-survey-master.pdf
- AlphaGo: https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf
- Model-Based RL with Global Model: https://rse-lab.cs.washington.edu/papers/robot-rl-rss-11.pdf
- UC Berkeley CS294-112 Deep Reinforcement Learning