CS 440/ECE448 Lecture 33: Model-Based Reinforcement Learning

Mark Hasegawa-Johnson, 4/2022

Including slides by Svetlana Lazebnik, 11/2016

By Nicolas P. Rougier - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=29327040

Review: Markov Decision Process

- MDP defined by states, actions, transition model, reward function
- The "solution" to an MDP is the policy: what do you do when you're in any given state
- The Bellman equation tells the utility of any given state, and incidentally, also tells you the optimum policy. The Bellman equation is N nonlinear equations in N unknowns (the policy), therefore it can't be solved in closed form.
- Value iteration:
  - At the beginning of the (i+1)'st iteration, each state's value is based on looking ahead i steps in time
  - … so finding the best action = optimize based on (i+1)-step lookahead
- Policy iteration:
  - Find the utilities that result from the current policy,
  - Improve the current policy

Reinforcement learning: Basic scheme

But what if you don't know $P(s'|s, a)$ or $R(s)$?

Answer: "learning by doing" (a.k.a. reinforcement learning).

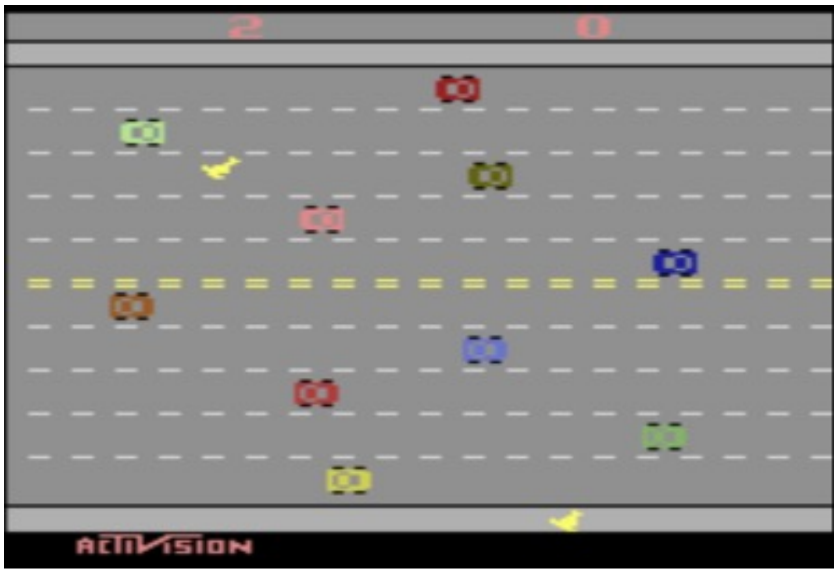
In each time step:

- Take some action
- Observe the outcome of the action: successor state and reward
- Update some internal representation of the environment and policy
- If you reach a terminal state, just start over (each pass through the environment is called a *trial*)

Model-Based and Model-Free RL

- Model-Based Reinforcement Learning:
  - Explore randomly.
  - At each state s, see what reward you get. Estimate $R(s)$ from these measurements.
  - At each state s, try some action a, and see what state s' you end up in. Estimate $P(s'|s, a)$ from these measurements.
  - Once you have learned $P(s'|s, a)$ and $R(s)$ well enough, then solve the MDP to find the optimal policy, $\pi(s)$.
- Model-Free Reinforcement Learning:
  - Learn a function $Q(s, a)$ = quality of action a in state s, or…
  - Learn the best policy, $\pi(s)$, directly.
  - Next lecture: more about how you might accomplish these things.

Example of model-based reinforcement learning: Playing classic Atari video games



Screenshot of the video game "Freeway," copyright Activision. Reproduced here under the terms of fair use enumerated at https://en.wikipedia.org/w/index.php?curid=56419703

Model-Based Reinforcement Learning for Atari (Kaiser, Babaeizadeh, Milos, Osinski, Campbell, Czechowski, Erhan, Finn, Kozakowski, Levine, Mohiuddin, Sepassi, Tucker, and Michalewski)

- Blog and videos: https://sites.google.com/view/modelbasedrlatari/home
- Article: https://arxiv.org/abs/1903.00374

Example of model-based reinforcement learning: Theseus the Mouse

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

Model-based reinforcement learning: Theseus' strategy

Learning phase:

- At each position in the maze (s),
  - For every possible action $a \in \{\text{Forward}, \text{Left}, \text{Right}, \text{Back}\}$:
    - If the action succeeded in changing the state ($s' \neq s$), then set $P(s'|s,a) = 1$
    - If not, set $P(s'|s,a) = 0$ for all $s' \neq s$

Once you've learned the maze, then compute the best policy ($\pi(s)$) using Value Iteration.

- If $P(s'|s,a) \in \{0,1\}$, Value Iteration = BFS

Outline of Today's Lecture

- Reinforcement learning
  - Model-based: learn $P(s'|s,a)$ and $R(s)$, then solve the MDP.
  - Model-free: learn $\pi(s)$ and/or $Q(s,a)$.
- The observation, model, policy loop
  - How it works: observe at random, estimate model, optimize policy
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least $1/\epsilon$ times.
  - Epsilon-greedy learning: explore w/prob. ϵ, exploit w/prob $1-\epsilon$.

Outline

- Reinforcement learning
  - Model-based: learn $P(s'|s,a)$ and $R(s)$, then solve the MDP.
  - Model-free: learn $\pi(s)$ and/or $Q(s,a)$.
- The observation, model, policy loop
  - How it works: observe at random, estimate model, optimize policy
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least $1/\epsilon$ times.
  - Epsilon-greedy learning: explore w/prob. ϵ, exploit w/prob $1-\epsilon$.

The observation-model-policy loop

Basic idea:

1. Observation: Follow some initial policy, to guide your actions.
2. Model: Try to learn $P(s'|s, a)$ and $R(s)$.
3. Policy: Use your estimated $P(s'|s, a)$ and $R(s)$ to decide on a new policy (using Value Iteration, for example).

Observation-Model-Policy Loop: Theseus

- If you're in state s, and there's an action, a, that you've never taken before while in this state, then take it.
- If you've already taken all possible actions from this state, then choose the best one.
- Continue re-estimating the model after every action. If transition probabilities change, compute a better policy.

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

What Theseus never had to deal with: Probabilities

- What happens if $P(s'|s,a)$ is not 0 or 1, but something in between?
- Trying it just once is not enough

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

How to deal with probabilities

- Suppose, for example, that you want to estimate $P(s'|s,a)$ with a precision of 0.1.
- In other words, if the true value is $P(s'|s,a)$, and your estimate is $\hat{P}\ (s'|s,a)$, you want it to be true that
  $|\hat{P}\ (s'|s,a) - P(s'|s,a)|\ <\ 0.1$
- How can you do that?

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

The epsilon-first strategy

The "epsilon-first" strategy tries every action $1/\epsilon$ times, where ϵ is the desired modeling precision. For example, if we want

$$|\hat{P}\,(s'|s,a) - P(s'|s,a)| \; < \; 0.1$$

… then we might set $\frac{1}{\epsilon} = 10.^{*}$

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

* We can never guarantee that $|\hat{P}\,(s'|s,a) - P(s'|s,a)| < \epsilon$ with 100% confidence, but using $1/\epsilon$ trials is enough to be pretty confident. If you've taken ECE 313 or CS 361, you should be able to work out the relationship more precisely.

The epsilon-first strategy

The epsilon-first strategy works like this:

- Keep two different tables:
  - $N(s,a)$ tells you how many times action a has been performed in state s
  - $M(s'|s,a)$ is the number of times that it resulted in state s'.
  - The current model estimate is

$$P(s'|s,a) \approx \frac{M(s'|s,a)}{N(s,a)}$$

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

The epsilon-first strategy

As you wander through the maze, you reach some state, s.

- If there is any action, a, for which $N(s,a) < 1/\epsilon$, then try that action.
- If not, then use value iteration (with the current estimates of $P(s'|s,a)$) to decide what is the best action to take.

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

The epsilon-first strategy

As you wander through the maze, you reach some state, s.

- If there is any action, a, for which $N(s,a) < 1/\epsilon$, then **explore** (= try the action, to see what it does).
- If not, then **exploit** your knowledge (choose the action that, according to your model, will lead to the highest utility).

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

Outline

- Reinforcement learning
  - Model-based: learn $P(s'|s,a)$ and $R(s)$, then solve the MDP.
  - Model-free: learn $\pi(s)$ and/or $Q(s,a)$.
- The observation, model, policy loop
  - How it works: observe at random, estimate model, optimize policy
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least $1/\epsilon$ times.
  - Epsilon-greedy learning: explore w/prob. ϵ, exploit w/prob $1-\epsilon$.

Exploration vs. Exploitation

- **Exploration:** take a new action with unknown consequences
  - Pros:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - Cons:
    - When you're exploring, you're not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - Pros:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff
  - Cons:
    - Might also prevent you from discovering the true optimal strategy

"Search represents a core feature of cognition:"
Exploration versus exploitation in space, mind, and society.

How to trade off exploration vs. exploitation

Epsilon-first strategy: when you reach state s, check how many times you've tested each of its available actions.

- **Explore for the first $\mathbf{1/\epsilon}$ trials**: If the least-explored action has been tested fewer than $1/\epsilon$ times, then perform that action ($1/\epsilon$ is an integer).
- **Exploit thereafter:** Once you've finished exploring, start exploiting (work to maximize your personal utility).

Epsilon-greedy strategy: in every state, every time, forever,

- **With probability $\boldsymbol{\epsilon}$, Explore**: choose any action, uniformly at random.
- **With probability $(\mathbf{1}-\boldsymbol{\epsilon})$, Exploit**: choose the action with the highest expected utility, according to your current estimates.
- Guarantee: $P(s'|s,a)$ converges to its true value as #trials $\rightarrow \infty$.

The epsilon-greedy strategy

The epsilon-greedy strategy works like this:

- Keep two different tables:
  - $N(s,a)$ tells you how many times action a has been performed in state s
  - $M(s'|s,a)$ is the number of times that it resulted in state s'.
  - The current model estimate is

$$P(s'|s,a) \approx \frac{M(s'|s,a)}{N(s,a)}$$

Claude Shannon and Theseus the Mouse. Public domain image, Bell Labs.

The epsilon-greedy strategy

As you wander through the maze, you reach some state, s. You generate a uniform random number, $z \in (0,1)$.

- If $z \leq \epsilon$, then **explore**. Choose an action, a, uniformly at random, and try it. See what s' results. Increment $N(s,a)$ and $M(s'|s,a)$.
  - This happens with probability ϵ.
- If $z > \epsilon$, then **exploit**. Use value iteration or policy iteration to figure out the best action in the current state, then do that action.
  - This happens with probability $1 - \epsilon$.

Compare: Epsilon-first and Epsilon-greedy

For both: $P(s'|s,a) \approx \frac{M(s'|s,a)}{N(s,a)}$

<u>Advantages of Epsilon-first:</u>

- In the beginning, when $P(s'|s,a)$ is still inaccurate, we just try things at random (explore).
- We can choose the level of precision that's "enough" for us. When $P(s'|s,a)$ reaches that point, we stop exploring, and instead, we focus on getting the biggest rewards possible (exploit).

<u>Advantages of Epsilon-greedy:</u>

- Gradually, over a series of many experiments, $N(s,a) \rightarrow \infty$
- Therefore, as the number of experiments gets large,

$$|\hat{P}(s'|s,a) - P(s'|s,a)| \rightarrow 0$$

Outline

- Reinforcement learning
  - Model-based: learn $P(s'|s,a)$ and $R(s)$, then solve the MDP.
  - Model-free: learn $\pi(s)$ and/or $Q(s,a)$ directly, without ever explicitly learning $P(s'|s,a)$ and $R(s)$.
- The observation, model, policy loop
  - How it works: observe at random, estimate model, optimize policy
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least $1/\epsilon$ times.
  - Epsilon-greedy learning: explore w/prob. ϵ, exploit w/prob $1-\epsilon$.