CS 440/ECE448 Lecture 23: Model-Based Reinforcement Learning

Mark Hasegawa-Johnson, 4/2021

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

Reinforcement learning

- **Solving a known MDP**
  - Given:
    - Transition model $P(s' | s, a)$
    - Reward function $R(s)$
  - Find:
    - Policy $\pi(s)$

- **Reinforcement learning**
  - Transition model and reward function initially unknown
  - Still need to find the right policy
  - "Learn by doing"

Reinforcement learning: Basic scheme

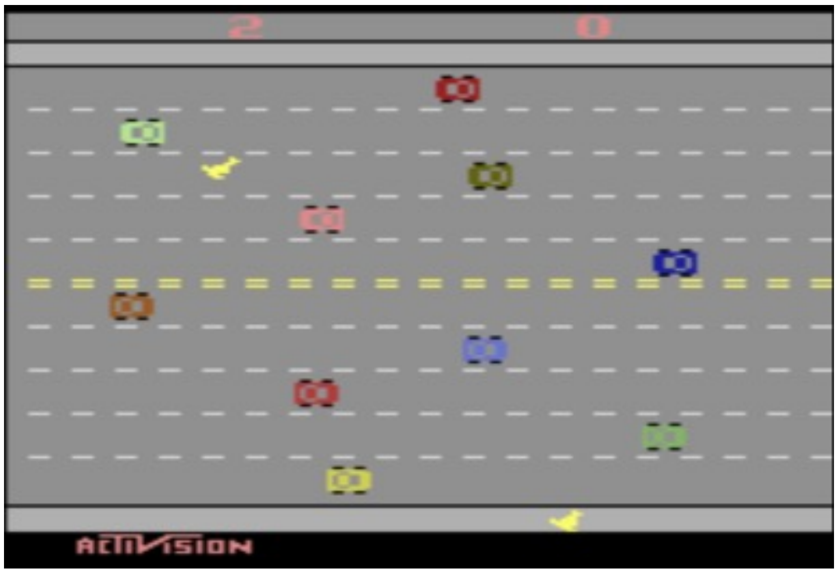
In each time step:

- 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

In 1950, Claude Shannon built a robot mouse named Theseus. As he explored his maze, Theseus learned:

- $s =$ Position in the maze
- $a =$ Forward, Left, Right, Back
- $P(s'|s,a) = 1$ if the movement from s to s' succeeds, otherwise 0
- $R(s) = 1$ when Theseus reaches the end of the maze, 0 otherwise

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

For more information about Theseus, and for a great introduction to the goals of reinforcement learning in general (and the problem of exploration versus exploitation), I recommend this video.

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
  - How it can fail: an example
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least ϵ 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
  - How it can fail: an example
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least ϵ times.
  - Epsilon-greedy learning: explore w/prob. ϵ, exploit w/prob $1 - \epsilon$.

The observation-model-policy loop

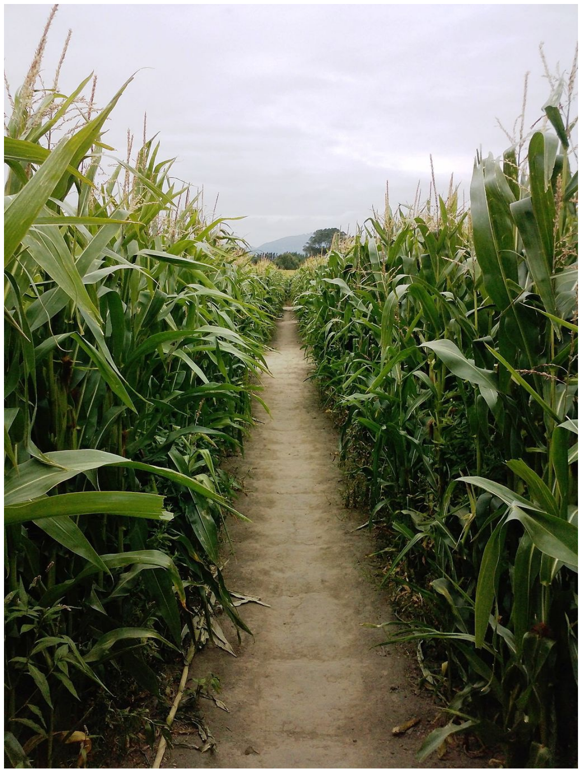
Basic idea:

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, and repeat.

1. Observation: Follow some initial policy, and keep a record of what happens

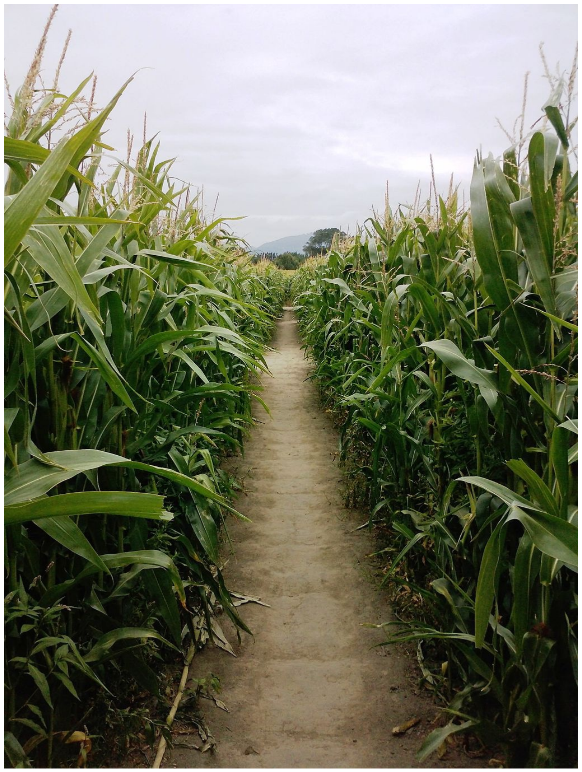
Enter the maze…

A view from inside a corn maze near Christchurch, New Zealand



2. Model: Try to learn P(s'|s,a) and R(s)

Enter the maze…



A view from inside a corn maze near Christchurch, New Zealand

By Hugho226 - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=30724285

…update your map as you go…

By Philip Mitchell - http://www.dwarvenforge.com/dwarvenforums/viewtopic.php?pid=15595#p15595, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1913429

3. Policy: Solve the MDP to find a new policy

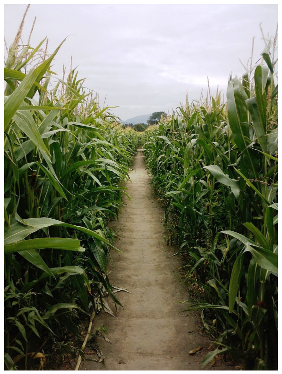
…and figure out what to do.

By Edward Burne-Jones - lgFxdQtUgyzs7Q at Google Cultural Institute, zoom level maximum, Public Domain, https://commons.wikimedia.org/w/index.php?curid=29661124

…update your map as you go…

By Philip Mitchell - http://www.dwarvenforge.com/dwarvenforums/viewtopic.php?pid=15595#p15595, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1913429

1. Follow some initial policy, to guide your actions



By Hugho226 - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=30724285

For $t = 1$ to n (for some sufficiently large value of n):

- Observe: find out what is your current state (s).
- Act: use your current policy to choose an action (a).
- Observe: see what state you move to (s').
- Observe: see what reward you receive (R).

If you finish the game within this many steps, start over, until you reach your desired n.

Keep a record of your (s, a, s', R) tuples. These are now your training database:

$$\mathcal{D} = \{(s_1, a_1, s_1', R_1), (s_2, a_2, s_2', R_2), \dots, (s_n, a_n, s_n', R_n)\}$$

2. Try to learn P(s’|s,a) and R(s)

By Philip Mitchell - http://www.dwarvenforge.com/dwarvenforums/viewtopic.php?pid=15595#p15595, CC BY-SA 3.0

Just like Bayesian networks! Use maximum likelihood parameter learning, possibly also with Laplace smoothing.

$$P(s'|s,a) = \frac{\#\text{ times that action } a \text{ in state } s \text{ led to state } s'}{\#\text{ times action } a \text{ was performed in state } s}$$

$$R(s) = R \text{ that was received when you were in state } s$$

If s or a are continuous-valued, you’ll have to estimate these using a neural network or some other parametric model.

3. Update your policy

By Edward Burne-Jones - lgFxdQtUgyzs7Q at Google Cultural Institute, zoom level maximum, Public Domain, https://commons.wikimedia.org/w/index.php?curid=29661124

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$$

As you know from last lecture, you'll have to use value iteration or policy iteration to solve for $\pi(s)$ given $P(s'|s,a)$ and $R(s)$.

Model-based reinforcement learning

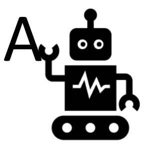
Basic idea:

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, and repeat.

Why does this fail?

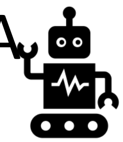
Observe-Model-Policy loop

Suppose we know the states (A, B, C, …) and the actions (Right, Left, Up, Down), but we don't know rewards or transition probabilities.

A 	B	C
D	E	F
G	H	I
J	K	L

Observe

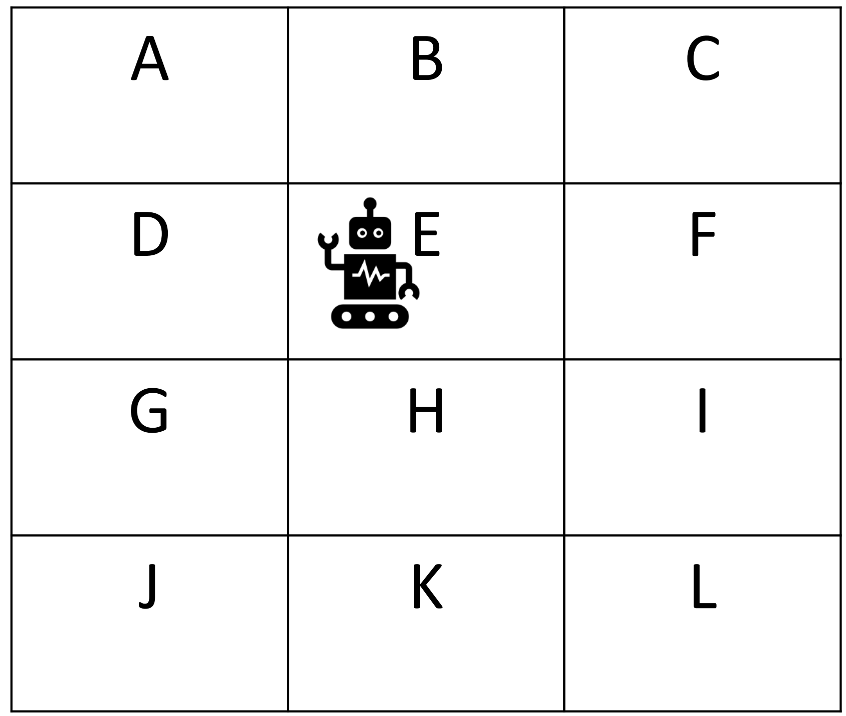
Start with some initial policy, e.g., “choose a direction at random.”

A 	B	C
D	E	F
G	H	I
J	K	L

Observe

Record the sequence of actions and consequences.

s	a	s’	R
A	Right	B	-0.04
B	Down	E	-0.04
E	Left	D	-0.04
D	Right	E	1

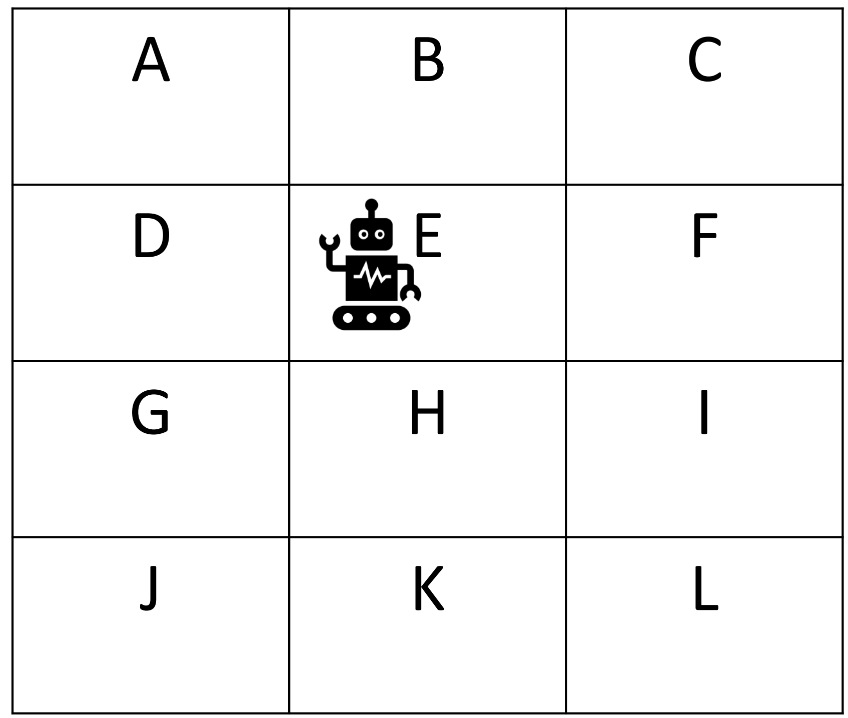


Model

Estimate the transitions & rewards.

s	R(s)
A	-0.04
B	-0.04
D	1
E	-0.04
Else	?

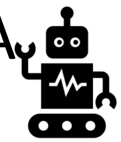
s,a	s'	P(s'|s,a)
A,Right	B	1.0
A,Right	*	0
B,Down	E	1.0
B,Down	*	0
D,Left	E	1.0
D,Left	*	0
E,Right	D	1.0
E,Right	*	0
Else	*	?



Policy

Use value iteration or policy iteration to find the new optimal policy.

s	$\pi(s)$
A	Right
B	Down
D	Left
E	Right
Else	?

A 	B	C
D	E	F
G	H	I
J	K	L

What went wrong?

- If you always act according to your current estimated optimal policy, then you will never learn the consequences of any other action.
- On the other hand, if you always act randomly, then you'll never maximize reward.
- How can we balance these things?

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
  - How it can fail: an example
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least ϵ times.
  - Epsilon-greedy learning: explore w/prob. ϵ, exploit w/prob $1-\epsilon$.

How did Theseus solve this problem?

- 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.

Extending Theseus to stochastic environments: the epsilon-first strategy

- If you have tried (state,action) combination less than ϵ times, then try it.
- Choose ϵ so that your estimate of $P(s'|s,a)$ will have some desired precision, e.g., $\epsilon = 10$ gives precision of 0.1.
- 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.

Advantages and disadvantages of the epsilon-first strategy

Advantages:

- After you've finished estimating the model, then you get to concentrate on maximizing reward.

Disadvantage:

- Your understanding of disfavored actions will never improve, because you'll stop using them.

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

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 ϵ trials**: If the least-explored action has been tested fewer than ϵ times, then perform that action (ϵ 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,

- **Explore with probability $\mathbf{0 < \epsilon < 1}$**: choose any action, uniformly at random.
- **Exploit with probability $(\mathbf{1 - \epsilon})$**: 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 $\rightarrow \infty$.

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
  - How it can fail: an example
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least ϵ times.
  - Epsilon-greedy learning: explore w/prob. ϵ, exploit w/prob $1-\epsilon$.