CS 440/ECE448 Lecture 30: Reinforcement Learning

Mark Hasegawa-Johnson, 4/2020

Including slides by Svetlana Lazebnik, 11/2016

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*)

Theseus the Mouse

- The study of reinforcement learning by machines goes back at least to 1950, when Claude Shannon built a robot mouse named “Theseus.”
- Like his classical namesake, Theseus had to learn how to navigate a maze.
- He learned by trial and error.
- His reinforcement learning strategy permitted him to adapt to changes in the maze.

Found at Bell Labs website, The photo was part of a press release, widely circulated in the public domain through news articles appearing in national newspapers and books. Its use in Wikipedia is therefore claimed under the Fair use guidelines., https://en.wikipedia.org/w/index.php?curid=4289542

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

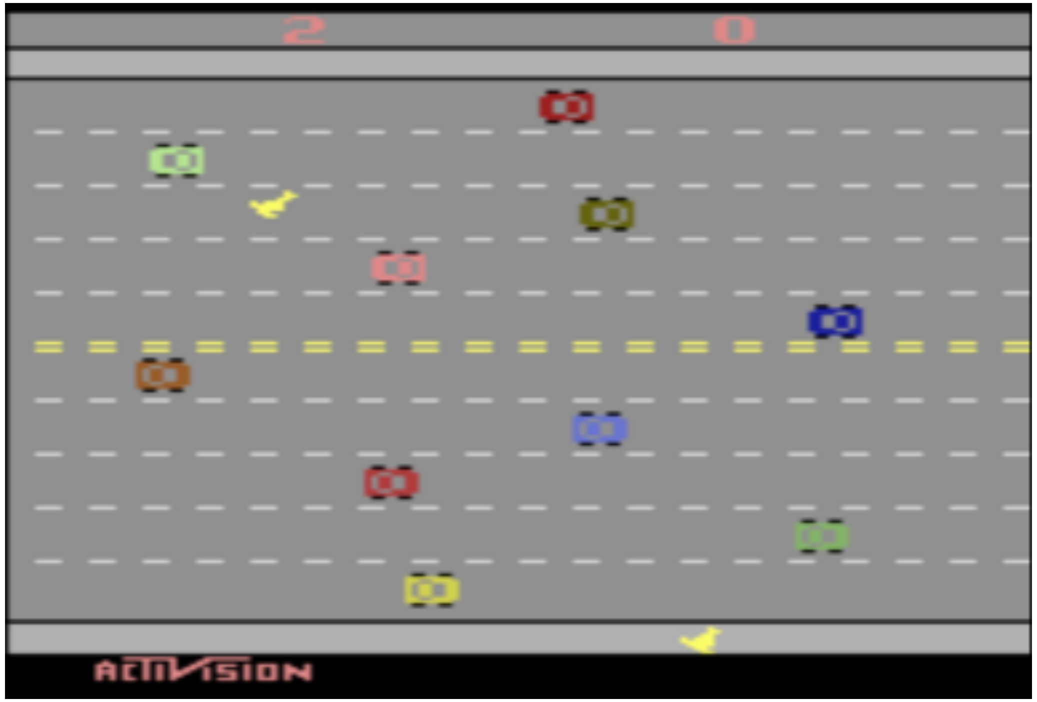
- Types of reinforcement learning
  - Model-free: keep track of the quality of each action in each state.
  - Model-based: try to learn P(s'|s,a) explicitly.
- Model-based reinforcement learning
  - The observation -> model -> policy loop
- Exploration versus Exploitation
  - Epsilon-greedy learning versus Epsilon-first learning

Model-based reinforcement learning

Model-based reinforcement learning uses what's sometimes called the observation -> model -> policy loop.

- Test a few actions, and **observe** the results
- Based on those results, estimate a **model**: a lookup table (or neural network estimate) of the transition probabilities $P(s'|s,a)$, and of the reward function $R(s)$.
- Based on the model, use value iteration or policy iteration to find an optimal **policy**.
- … and repeat this loop, as often as you can.

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

Model-free reinforcement learning

- In model-free reinforcement learning, we never try to explicitly learn what the world is like ($P(s'|s,a)$ and $R(s)$).
- Instead, we keep track of a simple lookup table:
  - In state s, if I perform action a, what will be my expected utility?
  - This is called the "quality" of action a in state s, $Q(s,a)$.
- If the states and actions are discrete, $Q(s,a)$ can be a lookup table. If not, $Q(s,a)$ can be a function learned by a neural network.

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

Screenshot of the video game "Breakout," copyright Activision. Reproduced here under the terms of fair use enumerated at https://en.wikipedia.org/w/index.php?curid=52132637

Playing Atari with Deep Reinforcement Learning (Mnih, Kavukcuoglu, Silver, Graves, Antonoglou, Wierstra, and Riedmiller)

- Video: https://www.youtube.com/watch?v=cjpEIotvwFY&feature=youtu.be

- Article: https://arxiv.org/abs/1312.5602

Reinforcement learning strategies

- **Model-based**
  - Learn the model of the MDP (transition probabilities and rewards) and try to solve the MDP concurrently
- **Model-free**
  - Learn how to act without explicitly learning the transition probabilities $P(s' \mid s, a)$
  - **Q-learning:** learn an action-utility function $Q(s,a)$ that tells us the value of doing action a in state s

Outline

- Types of reinforcement learning
  - Model-free: keep track of the quality of each action in each state.
  - Model-based: try to learn P(s’|s,a) explicitly.
- Model-based reinforcement learning
  - The observation -> model -> policy loop
- Exploration versus Exploitation
  - Epsilon-greedy learning versus Epsilon-first learning

Model-based reinforcement learning

Basic idea:

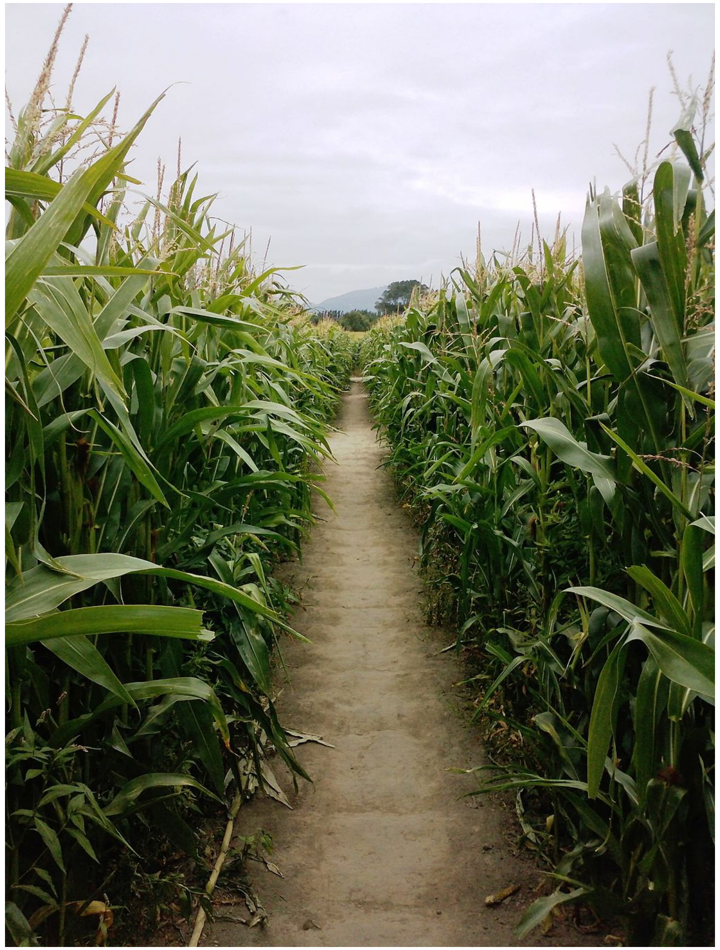
1. Follow some initial policy, to guide your actions.
2. Try to learn $P(s'|s,a)$ and $R(s)$.
3. Use your estimated $P(s'|s,a)$ and $R(s)$ to decide on a new policy, and repeat.

1. Follow some initial policy, to guide your actions

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

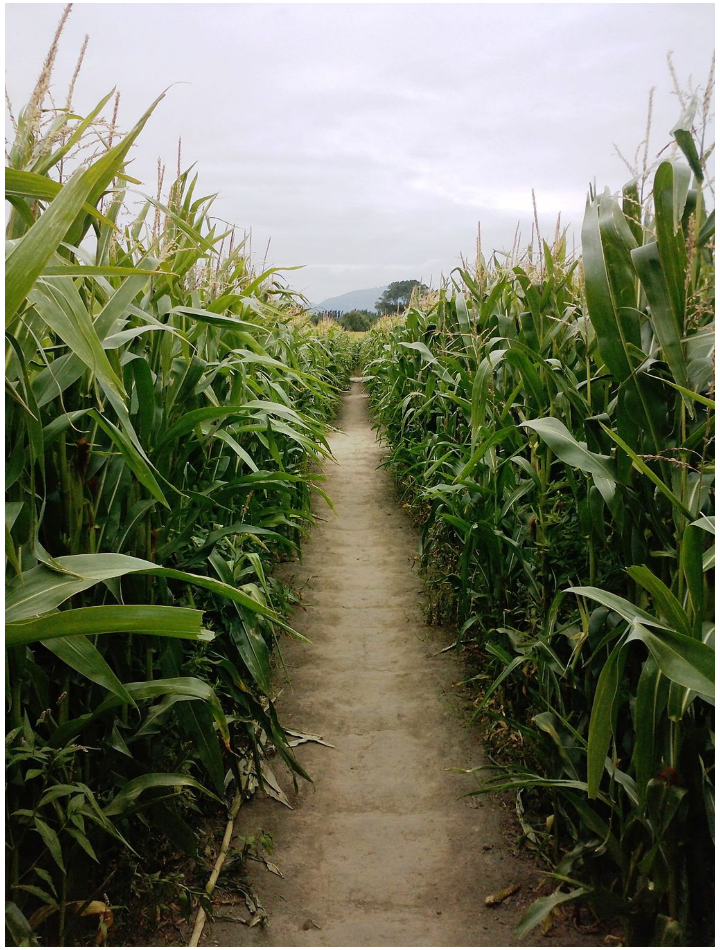


2. Try to learn $P(s'|s,a)$ and $R(s)$

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

Enter the maze…



…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. Update your policy

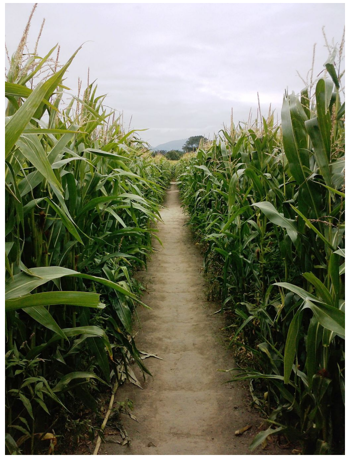
…and be ready to act.

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:

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

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. Follow some initial policy, to guide your actions.
2. Try to learn $P(s'|s,a)$ and $R(s)$.
3. Use your estimated $P(s'|s,a)$ and $R(s)$ to decide on a new policy, and repeat.

Why does this fail?

Model-based reinforcement learning

Basic idea:

1. Follow some initial policy, to guide your actions.
2. Try to learn P(s'|s,a) and R(s).
3. Use your estimated P(s'|s,a) and R(s) to decide on a new policy, and repeat.

Why does this fail?

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

1. If your current policy is $\pi(s) = a_1$, then you will never perform action a_2 in state s.
2. Therefore, your estimate of $P(s'|s,a_2)$ will be completely uninformed. You'll probably think that $P(s'|s,a_2)$ is uniform (every s' is equally likely).
3. If a_1 leads to a good state more than half the time, then you will conclude that a_1 is better than a_2. So when you revise your policy in step 3, you will still choose $\pi(s) = a_1$. …and the trap snaps shut behind you…

Outline

- Types of reinforcement learning
  - Model-free: keep track of the quality of each action in each state.
  - Model-based: try to learn P(s’|s,a) explicitly.
- Model-based reinforcement learning
  - The observation -> model -> policy loop
- Exploration versus Exploitation
  - Epsilon-greedy learning versus Epsilon-first learning

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 ϵN trials**: If the least-explored action has been tested fewer than ϵN times, then perform that action.
- **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 ϵ**: choose any action, uniformly at random.
- **Exploit with probability $(1-\epsilon)$**: choose the action with the highest expected utility, according to your current estimates.

How to trade off exploration vs. exploitation

Epsilon-first strategy:

- Advantages:
  - ϵN can be chosen to guarantee that your model is correct w/pre-specified level of confidence.
  - After the first ϵN trials, you are always getting best possible reward.
- Disadvantages:
  - After the first ϵN trials, your model stops improving.
  - If the world changes, you won't know.

Epsilon-greedy strategy:

- Advantages:
  - If the world is static, epsilon-greedy converges to the correct model.
  - If the world changes, you'll find out.
- Disadvantages:
  - Never, at any time, will you focus solely on maximizing your utility (exploiting). You are always "wasting" ϵ of your time exploring.

There are dozens of other ways you can balance exploration versus exploitation.