Artificial Intelligence Foundation – JC3001

Lecture 44: Reinforcement Learning - I

Prof. Aladdin Ayesh (aladdin.ayesh@abdn.ac.uk)

Dr. Binod Bhattarai (binod.bhattarai@abdn.ac.uk)

Dr. Gideon Ogunniye, (g.ogunniye@abdn.ac.uk)

October 2025

Material adapted from:

Russell and Norvig (AIMA Book): Chapter 22

Sutton and Barto (Reinforcement Learning: An Introduction 2nd ed.)

David Silver (UCL)

Michael Littman (Brown University) and Charles Isbell (GA Tech)

Course Progression

- Part 1: Introduction
  1. Introduction to AI ✓
  2. Agents ✓
- Part 2: Problem-solving
  1. Search 1: Uninformed Search ✓
  2. Search 2: Heuristic Search ✓
  3. Search 3: Local Search ✓
  4. Search 4: Adversarial Search ✓
- Part 3: Reasoning and Uncertainty
  1. Reasoning 1: Constraint Satisfaction ✓
  2. Reasoning 2: Logic and Inference ✓
  3. Probabilistic Reasoning 1: BNs ✓
  4. Probabilistic Reasoning 2: HMMs ✓
- Part 4: Planning
  1. Planning 1: Intro and Formalism ✓
  2. Planning 2: Algorithms & Heuristics ✓
  3. Planning 3: Hierarchical Planning ✓
  4. Planning 4: Stochastic Planning ✓
- Part 5: Learning
  1. Learning 1: Intro to ML ✓
  2. Learning 2: Regression ✓
  3. Learning 3: Neural Networks ✓
  4. **Learning 4: Reinforcement Learning**
- Part 6: Conclusion
  1. Ethical Issues in AI
  2. Conclusions and Discussion

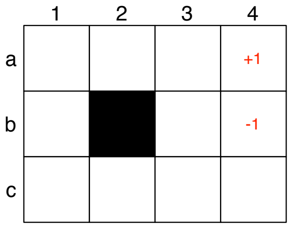
Objectives

- Bandit Problems
- Reinforcement Learning based Agents
- Tabular Reinforcement Learning
- Function Generalization

Outline

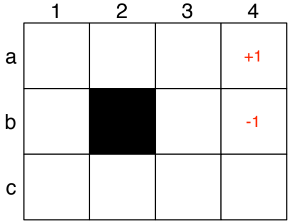
- Recap and Motivation
- Multi-armed Bandits
- Reinforcement Learning based Agents

Motivation



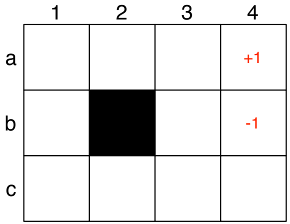
- Recall the grid world example we saw for MDPs

Motivation



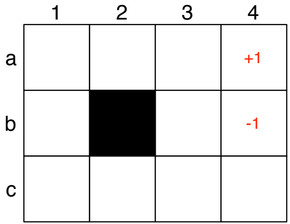
- Recall the grid world example we saw for MDPs
- We learned how to calculate the optimal policy using transition **probabilities** and **rewards**

Motivation



- Recall the grid world example we saw for MDPs
- We learned how to calculate the optimal policy using transition **probabilities** and **rewards**
- But what if we do not know the rewards, probabilities or both?

Motivation



- Recall the grid world example we saw for MDPs
- We learned how to calculate the optimal policy using transition **probabilities** and **rewards**
- But what if we do not know the rewards, probabilities or both?
- We should be able to explore the environment and discover the best policy

Main methods of learning:

- Supervised learning:
  From $(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots (\vec{x}^{(m)}, y^{(m)})$, derive a function $f(X^{(m)}) = Y^{(m)}$

Main methods of learning:

- Supervised learning:
  From $(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots (\vec{x}^{(m)}, y^{(m)})$, derive a function $f(X^{(m)}) = Y^{(m)}$
- Unsupervised learning:
  From $\vec{x}^{(1)}, \vec{x}^{(2)}, \vec{x}^{(m)}, \ldots, \vec{x}^{(m)}$, identify classes x_c and
  derive a function to calculate $P(X = x_c)$

Main methods of learning:

- Supervised learning:
  From $(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots (\vec{x}^{(m)}, y^{(m)})$, derive a function $f(X^{(m)}) = Y^{(m)}$
- Unsupervised learning:
  From $\vec{x}^{(1)}, \vec{x}^{(2)}, \vec{x}^{(m)}, \ldots, \vec{x}^{(m)}$, identify classes x_c and
  derive a function to calculate $P(X = x_c)$
- Reinforcement learning:
  From a sequence of state transitions (i.e. states and actions)
  and occasional rewards $[s^{\rightarrow r}, a, s^{\rightarrow r}, a, \ldots]$,
  discover the optimal policy $\pi^*(s)$ for this environment.

Machine Learning Problems

Which of the previous methods of learning would be more suitable to solve the following problems:

Sup.	Unsup.	Reinf.	
			Spam Filtering
			Speech Recognition
			Starlight classification
			Playing SFΞ against an unknown player
			Navigating an unknown environment
			Elevator controller

Machine Learning Problems

Which of the previous methods of learning would be more suitable to solve the following problems:

Sup.	Unsup.	Reinf.	
X			Spam Filtering
			Speech Recognition
			Starlight classification
			Playing SFΞ against an unknown player
			Navigating an unknown environment
			Elevator controller

Machine Learning Problems

Which of the previous methods of learning would be more suitable to solve the following problems:

Sup.	Unsup.	Reinf.	
X			Spam Filtering
X			Speech Recognition
			Starlight classification
			Playing SFΞ against an unknown player
			Navigating an unknown environment
			Elevator controller

Which of the previous methods of learning would be more suitable to solve the following problems:

Sup.	Unsup.	Reinf.	
X			Spam Filtering
X			Speech Recognition
	X		Starlight classification
			Playing SFΞ against an unknown player
			Navigating an unknown environment
			Elevator controller

Machine Learning Problems

Which of the previous methods of learning would be more suitable to solve the following problems:

Sup.	Unsup.	Reinf.	
X			Spam Filtering
X			Speech Recognition
	X		Starlight classification
		X	Playing SFΞ against an unknown player
			Navigating an unknown environment
			Elevator controller

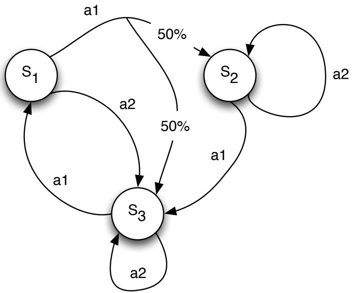
Which of the previous methods of learning would be more suitable to solve the following problems:

Sup.	Unsup.	Reinf.	
X			Spam Filtering
X			Speech Recognition
	X		Starlight classification
		X	Playing SFΞ against an unknown player
		X	Navigating an unknown environment
			Elevator controller

Which of the previous methods of learning would be more suitable to solve the following problems:

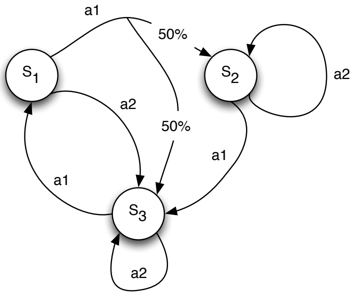
Sup.	Unsup.	Reinf.	
X			Spam Filtering
X			Speech Recognition
	X		Starlight classification
		X	Playing SFΞ against an unknown player
		X	Navigating an unknown environment
		X	Elevator controller

Markov Decision Process (MDP)



- An MDP is defined in terms of
  1. An initial state S_0
  2. A transition model $T(s, a, s') - P(s'|a, s)$ (Markovian)
  3. A reward function $R(s, a, s')$
- A solution to a MDP must specify what the agent should do for any state. Such a solution is called a policy

Markov Decision Process (MDP)



- An MDP is defined in terms of
  1. An initial state S_0
  2. A transition model $T(s, a, s') - P(s'|a, s)$ (Markovian)
  3. A reward function $R(s, a, s')$ — sometimes expressed as $R(s')$
- A solution to a MDP must specify what the agent should do for any state. Such a solution is called a policy

Finding the Optimal Policy

$$V(s) \leftarrow \left[\max_a \gamma \sum_{s'} P(s'|s,a) * V(s')\right] + R(s)$$

Finding the Optimal Policy

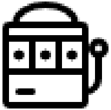
$$V(s) \leftarrow \left[\max_a \gamma \sum_{s'} P(s'|s,a) * V(s')\right] + R(s)$$

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a) * V(s')$$

- Recap and Motivation
- Multi-armed Bandits
- Reinforcement Learning based Agents

k-armed Bandit Problem

- Repeated choices among k different options or actions
- Each choice has a reward drawn from a stationary probability distribution
- Goal is to maximize rewards over a period of time



Thus, the value is the expected reward:

$$\begin{aligned} q_*(a) &\doteq \mathbb{E}\left[R_t \mid A_t = a\right] \forall a \in \{1, \ldots, k\} \\ &= \sum_r p(r \mid a) r \end{aligned}$$

And to maximize it over time

$$\arg\max_a q_*(a)$$

Computing the expected reward

Consider a doctor with 3 possible treatments to offer a patient.

- Each treatment can either cure the patient ($r = 1$), do nothing ($r = 0$), or make the patient sicker ($r = -1$), with probabilities ($p(r \mid drug)$) as follows:

  d_1 $p(r = 1 \mid d = d_1) = 0.1$, $p(r = 0 \mid d = d_1) = 0.8$, and $p(r = -1 \mid d = d_1) = 0.1$
  d_2 $p(r = 1 \mid d = d_2) = 0.3$, $p(r = 0 \mid d = d_2) = 0.1$, and $p(r = -1 \mid d = d_2) = 0.6$
  d_3 $p(r = 1 \mid d = d_3) = 0.3$, $p(r = 0 \mid d = d_3) = 0.5$, and $p(r = -1 \mid d = d_3) = 0.2$

Computing the expected reward

Consider a doctor with 3 possible treatments to offer a patient.

- Each treatment can either cure the patient ($r = 1$), do nothing ($r = 0$), or make the patient sicker ($r = -1$), with probabilities ($p(r \mid drug)$) as follows:

  d_1 $p(r = 1 \mid d = d_1) = 0.1$, $p(r = 0 \mid d = d_1) = 0.8$, and $p(r = -1 \mid d = d_1) = 0.1$

  d_2 $p(r = 1 \mid d = d_2) = 0.3$, $p(r = 0 \mid d = d_2) = 0.1$, and $p(r = -1 \mid d = d_2) = 0.6$

  d_3 $p(r = 1 \mid d = d_3) = 0.3$, $p(r = 0 \mid d = d_3) = 0.5$, and $p(r = -1 \mid d = d_3) = 0.2$

Compute q_* for all drug options and help the doctor choose:

d_1 $q(d = d_1) = (0.1 * 1) + (0.8 * 0) + (0.1 * -1) = 0$

d_2 $q(d = d_2) = (0.3 * 1) + (0.1 * 0) + (0.6 * -1) = -0.3$

d_3 $q(d = d_4) = (0.3 * 1) + (0.5 * 0) + (0.2 * -1) = 0.1$

Sample-Average Method

$$Q_t(a) \doteq \frac{\text{sum of rewards for } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

$$\frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

$Q_0(\;) = 0$

$Q_0(\;) = 0$

$Q_0(\;) = 0$

$$\frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

1

$Q_1($$) = 1$

0

$Q_1($$) = 0$

0

$Q_1($$) = 0$

$$\frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

1

0

$Q_2($$) = 0.5$

0

0

$Q_2($$) = 0$

0

0

$Q_2($$) = 0$

$$\frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

		
1	0	0
0	0	0
0	-1	0
$Q_3($$) = 0.5$	$Q_3($$) = -1$	$Q_3($$) = 0$

$$\frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

		
1	0	0
0	0	0
0	-1	0
0	0	1
$Q_4($$) = 0.5$	$Q_4($$) = -1$	$Q_4($$) = 1$

$$\frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

			
1	0		0
0	0		0
0	-1		0
0	0		1
0	0		0
$Q_5($$) = 0.5$	$Q_5($$)$ -0.5	$=$	$Q_5($$) = 1$

Incremental Update Rule

$$Q_{n+1} = \frac{1}{n} \sum_{i=1}^{n} R_i$$

Incremental Update Rule

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i)
\end{aligned}
$$

Incremental Update Rule

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i)
\end{aligned}
$$

Incremental Update Rule

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i)
\end{aligned}
$$

$$= \frac{1}{n}(R_n + (n-1)Q_n)$$

Recall: $Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i$

Incremental Update Rule

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{n}(R_n + (n-1)Q_n) \\
&= \frac{1}{n}(R_n + nQ_n - Q_n)
\end{aligned}
$$

Recall: $Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i$

Incremental Update Rule

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i)
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{n}(R_n + (n-1)Q_n) \\
&= \frac{1}{n}(R_n + nQ_n - Q_n) \\
&= Q_n + \frac{1}{n}(R_n - Q_n)
\end{aligned}
$$

Recall: $Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i$

Incremental Update Rule

We will see this form of update throughout this lecture:

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize}\,[\text{Target} - \text{OldEstimate}]$$

$$Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$$

$$Q_{n+1} = Q_n + \alpha_n(R_n - Q_n)$$

Outline

Agents execute **trials** in the environment, using some policy π.
Typical trials look like:

$$(1,1)_{-.04} \rightsquigarrow (1,2)_{-.04} \rightsquigarrow (1,3)_{-.04} \rightsquigarrow (1,2)_{-.04} \rightsquigarrow (1,3)_{-.04} \rightsquigarrow (2,3)_{-.04} \rightsquigarrow (3,3)_{-.04} \rightsquigarrow (4,3)_{+1}$$

$$(1,1)_{-.04} \rightsquigarrow (1,2)_{-.04} \rightsquigarrow (1,3)_{-.04} \rightsquigarrow (2,3)_{-.04} \rightsquigarrow (3,3)_{-.04} \rightsquigarrow (3,2)_{-.04} \rightsquigarrow (3,3)_{-.04} \rightsquigarrow (4,3)_{+1}$$

$$(1,1)_{-.04} \rightsquigarrow (2,1)_{-.04} \rightsquigarrow (3,1)_{-.04} \rightsquigarrow (3,2)_{-.04} \rightsquigarrow (4,2)_{-1}$$

What to do if we do not know R

What to do if we do not know R, or P

What to do if we do not know R, or P, or both?

- We can try to learn R and P

What to do if we do not know R, or P, or both?

- We can try to learn R and P
- Or something else that obviates the need to learn them

Agent Types for RL

What to do if we do not know R, or P, or both?

- We can try to learn R and P
- Or something else that obviates the need to learn them

What to do if we do not know R, or P, or both?

- We can try to learn R and P
- Or something else that obviates the need to learn them

Agent	We Know	What to Learn	What to use

What to do if we do not know R, or P, or both?

- We can try to learn R and P
- Or something else that obviates the need to learn them

Agent	We Know	What to Learn	What to use
Utility-based agent	P	$R \to U$	U

What to do if we do not know R, or P, or both?

- We can try to learn R and P
- Or something else that obviates the need to learn them

Agent	We Know	What to Learn	What to use
Utility-based agent	P	$R \to U$	U
Q-learning agent		$Q(s,a)$	Q

Agent Types for RL

What to do if we do not know R, or P, or both?

- We can try to learn R and P
- Or something else that obviates the need to learn them

Agent	We Know	What to Learn	What to use
Utility-based agent	P	$R \to U$	U
Q-learning agent		$Q(s, a)$	Q
Reflex agent		$\pi(s)$	π

Passive vs. Active

Agent	We Know	What to Learn	What to use
Utility-based agent	P	$R \rightarrow U$	U
Q-learning agent		$Q(s,a)$	Q
Reflex agent		$\pi(s)$	π

- Besides these designs, we decide how “adventurous” our agent will be

Passive vs. Active

Agent	We Know	What to Learn	What to use
Utility-based agent	P	$R \to U$	U
Q-learning agent		$Q(s,a)$	Q
Reflex agent		$\pi(s)$	π

- Besides these designs, we decide how “adventurous” our agent will be

Passive RL The agent simply follows a fixed policy, and learns the components above as he goes
The goal is to learn how good that policy is

Passive vs. Active

Agent	We Know	What to Learn	What to use
Utility-based agent	P	$R \to U$	U
Q-learning agent		$Q(s, a)$	Q
Reflex agent		$\pi(s)$	π

- Besides these designs, we decide how "adventurous" our agent will be

Passive RL The agent simply follows a fixed policy, and learns the components above as he goes
The goal is to learn how good that policy is

Active RL After learning the components for a while,
the agent changes policy to "cash in" on the learned components

To continue in the next session.