Introduction to Reinforcement Learning

Pratik Gajane

September 7, 2022

2AMM20 Research Topics in Data Mining
Eindhoven University of Technology

Preliminaries

Objectives

- To gain an understanding of various reinforcement learning problems and formulate them mathematically.
- To devise solution strategies for these problems.
- To prove performance guarantees for these solutions.

Prerequisites

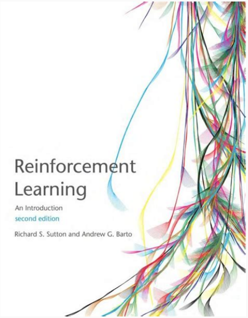
- Elementary statistics and probability theory.
- Comfort with applying mathematical tools.
- Bachelor's course worth of background knowledge in Data Mining and Machine Learning.

Class Information

- Course webpage : https://canvas.tue.nl/courses/21915/pages/reinforcement-learning-track-page
- Uploaded lecture slides may be updated as the course progresses.
- Contact me : p.gajane@tue.nl
- Please put [2AMM20] (with the square brackets) in the subject line of your email.

Resources

- Reinforcement Learning – An Introduction [Sutton and Barto, 2018][Chapter 1, 2 and 3]
- Bandit Algorithms [Lattimore and Szepesvari, 2020]
- Markov Decision Processes: Discrete Stochastic Dynamic Programming [Puterman, 1994][Chapter 4]
- Research Articles



Lecture 1 : Outline

- What is Reinforcement Learning?
- Elements of a Reinforcement Learning (RL) Problem
- Formulating RL with Multi-Armed Bandits
- Formulating RL with Markov Decision Processes

What is Reinforcement Learning?

What is Learning?

- Learning : Learning occurs when performance at given tasks improves with experience [Mitchell, 1997].

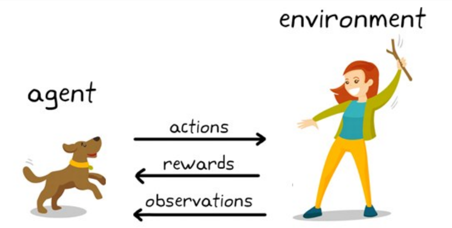
What is Learning?

- Learning : Learning occurs when performance at given tasks improves with experience [Mitchell, 1997].
- Definition not all encompassing : Breaking-in a new pair of shoes. Do the shoes *learn* to fit our feet better?

What is Learning?

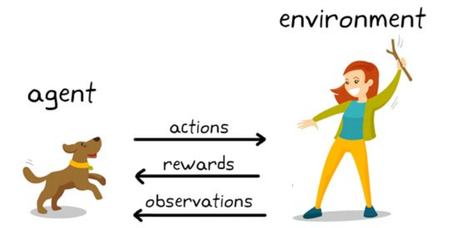
- Learning : Learning occurs when performance at given tasks improves with experience [Mitchell, 1997].
- Definition not all encompassing : Breaking-in a new pair of shoes. Do the shoes *learn* to fit our feet better?
- How do people and animals learn?

Learning by Reinforcement



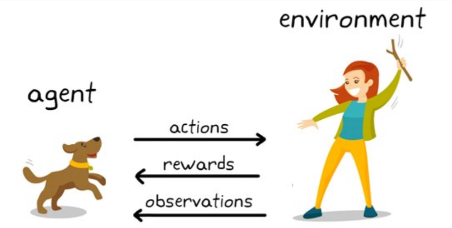
- Goal : To train the dog (*agent/model/learner*) to complete a task within an *environment*.

Learning by Reinforcement



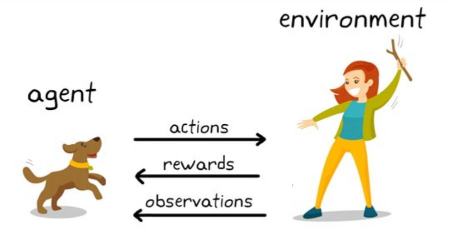
- Goal : To train the dog (*agent/model/learner*) to complete a task within an *environment*.
- Trainer issues a command/cue which the dog observes.

Learning by Reinforcement



- Goal : To train the dog (*agent/model/learner*) to complete a task within an *environment*.
- Trainer issues a command/cue which the dog observes.
- The dog performs an action.

Learning by Reinforcement



- Goal : To train the dog (*agent/model/learner*) to complete a task within an *environment*.
- Trainer issues a command/cue which the dog observes.
- The dog performs an action.
- If desired action,

  then reward,

  otherwise

  no (or negative) reward.

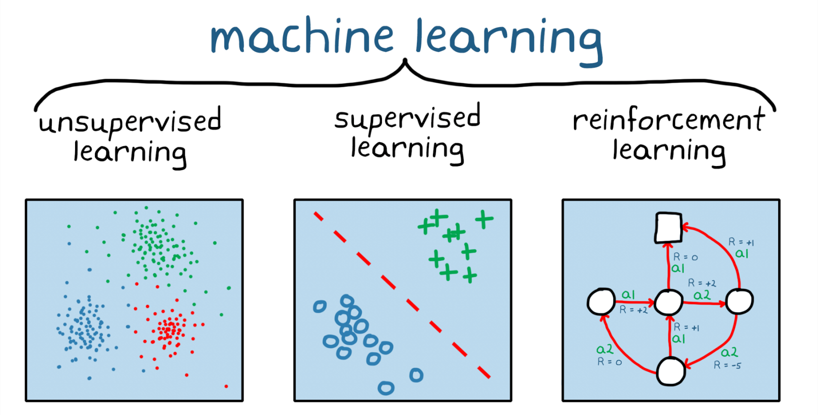
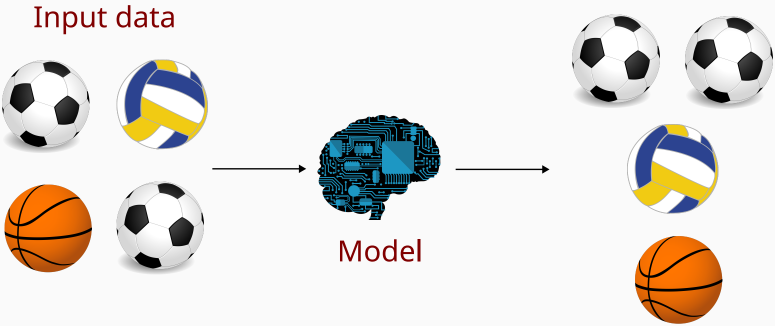


Image source: *mathworks*

Figure 1: Basic machine learning paradigms.

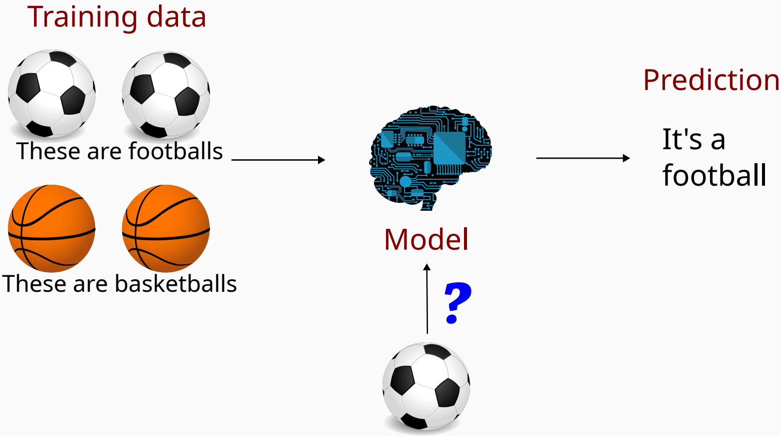
Unsupervised Learning

Aims to find structures/clusters in unlabeled data.



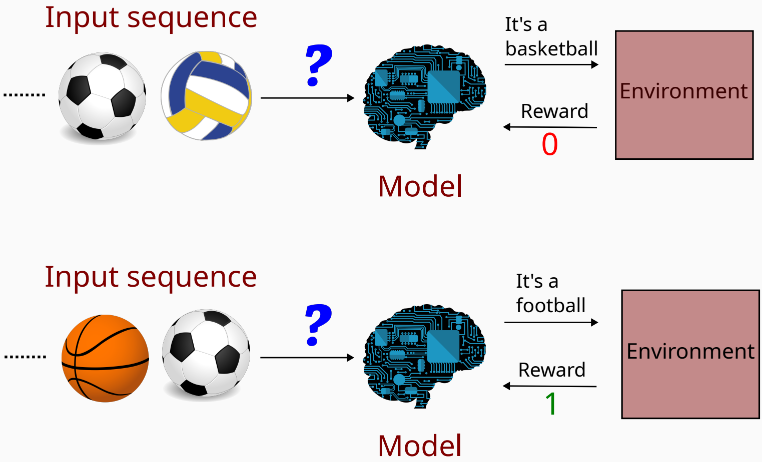
Supervised Learning

Learning from training data of labeled examples provided by a knowledgeable external supervisor.



Reinforcement Learning

Learning from the feedback provided by the environment in response to the model's behavior to optimize the reward.



Examples of Reinforcement Learning

- Make a humanoid robot walk.

Examples of Reinforcement Learning

- Make a humanoid robot walk.
- Manage an investment portfolio.

Examples of Reinforcement Learning

- Make a humanoid robot walk.
- Manage an investment portfolio.
- Play many different Atari games.

Examples of Reinforcement Learning

- Make a humanoid robot walk.
- Manage an investment portfolio.
- Play many different Atari games.
- Ad placement.

Examples of Reinforcement Learning

- Make a humanoid robot walk.
- Manage an investment portfolio.
- Play many different Atari games.
- Ad placement.
- Fly stunt manoeuvres in a helicopter.

Features of Reinforcement Learning

- Learning through a reward signal.

Features of Reinforcement Learning

- Learning through a reward signal.
- Feedback can be delayed.

Features of Reinforcement Learning

- Learning through a reward signal.
- Feedback can be delayed.
- Interactions are often sequential.

Features of Reinforcement Learning

- Learning through a reward signal.
- Feedback can be delayed.
- Interactions are often sequential.
- It is active, rather than passive.

Distinguishing Factors of Reinforcement Learning

- Differences from supervised learning :
  - No external supervisor, only a reward signal.
  - No need to obtain representative and correct training samples.

Distinguishing Factors of Reinforcement Learning

- Differences from supervised learning :
  - No external supervisor, only a reward signal.
  - No need to obtain representative and correct training samples.
- Differences from unsupervised learning :
  - Goal-directed.
  - Finding structures in input data not sufficient for maximizing the reward.

Distinguishing Factors of Reinforcement Learning

- Differences from supervised learning :
  - No external supervisor, only a reward signal.
  - No need to obtain representative and correct training samples.
- Differences from unsupervised learning :
  - Goal-directed.
  - Finding structures in input data not sufficient for maximizing the reward.
- Exploration/exploitation dilemma.

Exploration/Exploitation Dilemma

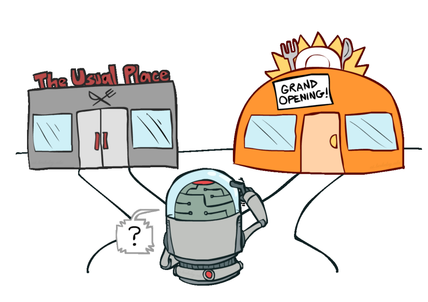


Image source: *UC Berkeley AI course, lecture 11*

Exploration/Exploitation Dilemma

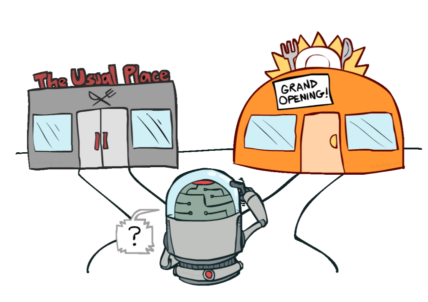


Image source: *UC Berkeley AI course, lecture 11*

- Exploit. Choose actions tried in the past and found to be rewarding.

Exploration/Exploitation Dilemma

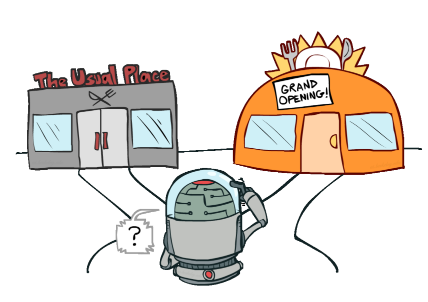


Image source: *UC Berkeley AI course, lecture 11*

- Exploit. Choose actions tried in the past and found to be rewarding.
- Explore. Choose unexplored actions to see if they are more rewarding.

Exploration/Exploitation Dilemma

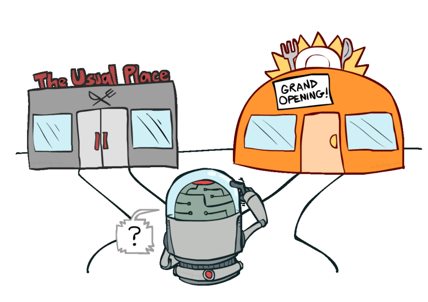
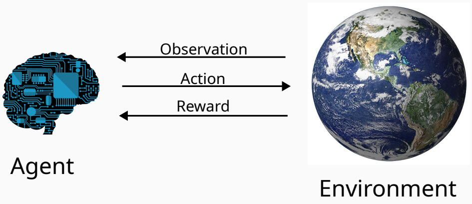


Image source: *UC Berkeley AI course, lecture 11*

- Exploit. Choose actions tried in the past and found to be rewarding.
- Explore. Choose unexplored actions to see if they are more rewarding.
- Neither exploration nor exploitation can be pursued exclusively.

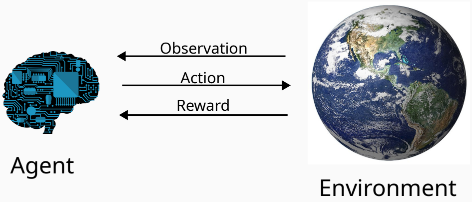
Reinforcement Learning : Problem Formulation

Reinforcement Learning : Agent and Environment



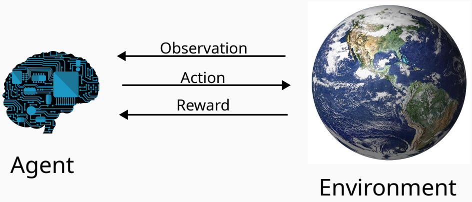
- Emits observation $o(t)$.

Reinforcement Learning : Agent and Environment



- Receives observation $o(t)$.
- Emits observation $o(t)$.

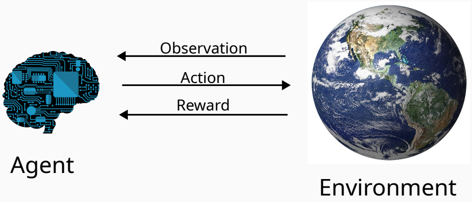
Reinforcement Learning : Agent and Environment



- Receives observation $o(t)$.
- Executes action $a(t)$.

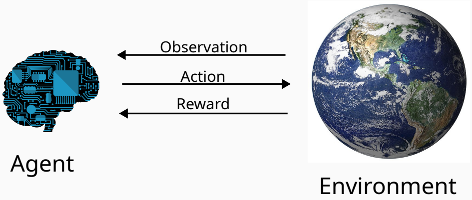
- Emits observation $o(t)$.

Reinforcement Learning : Agent and Environment



- Receives observation $o(t)$.
- Executes action $a(t)$.

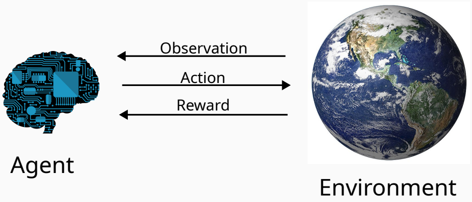
- Emits observation $o(t)$.
- Receives action $a(t)$.



- Receives observation $o(t)$.
- Executes action $a(t)$.

- Emits observation $o(t)$.
- Receives action $a(t)$.
- Emits reward $r(t)$.

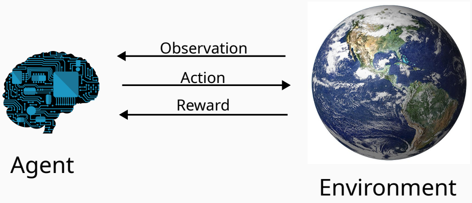
Reinforcement Learning : Agent and Environment



- Receives observation $o(t)$.
- Executes action $a(t)$.
- Receives reward $r(t)$.

- Emits observation $o(t)$.
- Receives action $a(t)$.
- Emits reward $r(t)$.

Reinforcement Learning : Agent and Environment



- Receives observation $o(t)$.
- Executes action $a(t)$.
- Receives reward $r(t)$.

- Emits observation $o(t)$.
- Receives action $a(t)$.
- Emits reward $r(t)$.

- Horizon T : time step when the process ends.

Elements of a Reinforcement Learning (RL) Problem

- State and action.

Elements of a Reinforcement Learning (RL) Problem

- State and action.
- Reward : Indicates what is good and bad for the agent in an immediate sense.

Elements of a Reinforcement Learning (RL) Problem

- State and action.
- Reward : Indicates what is good and bad for the agent in an immediate sense.
- Value : Indicates long-term desirability of a state.

Elements of a Reinforcement Learning (RL) Problem

- State and action.
- Reward : Indicates what is good and bad for the agent in an immediate sense.
- Value : Indicates long-term desirability of a state.
- Policy : Encoding of the agent's behavior.

Elements of a Reinforcement Learning (RL) Problem

- State and action.
- Reward : Indicates what is good and bad for the agent in an immediate sense.
- Value : Indicates long-term desirability of a state.
- Policy : Encoding of the agent's behavior.
- Model : Interpretation of the environment's behaviour.

Elements of a RL Problem : State and Action

Image source: *Chess.com*

- State $s \in \mathcal{S}$ describes the current situation.
- Examples : Chess position, robot's current position.

Elements of a RL Problem : State and Action

Image source: *Chess.com*

- State $s \in \mathcal{S}$ describes the current situation.
- Examples : Chess position, robot's current position.
- Actions $a \in \mathcal{A}$ are the choices available to the agent.
- Actions are permitted to affect the future state.

Elements of a RL Problem : Reward I

Image source: *Chess.com*

- A numerical feedback signal $r(t)$.

Elements of a RL Problem : Reward I

Image source: *Chess.com*

- A numerical feedback signal $r(t)$.
- Indicates if the agent's action $a(t)$ was good i.e. defines the goal : to maximize the cumulative sum of rewards.

Elements of a RL Problem : Reward I

Image source: *Chess.com*

- A numerical feedback signal $r(t)$.
- Indicates if the agent's action $a(t)$ was good i.e. defines the goal : to maximize the cumulative sum of rewards.
- Reward hypothesis : *Any goal can be formalized as the outcome of maximizing a cumulative reward.*

Elements of a RL Problem : Reward II

Image source: *Chess.com*

- Dependent on the current state and the current action.

Elements of a RL Problem : Reward II

Image source: *Chess.com*

- Dependent on the current state and the current action.
- The agent can only influence the reward through its actions.

Elements of a RL Problem : Reward II

Image source: *Chess.com*

- Dependent on the current state and the current action.
- The agent can only influence the reward through its actions.
- Examples of reward

Elements of a RL Problem: Reward II

Image source: *Chess.com*

- Dependent on the current state and the current action.
- The agent can only influence the reward through its actions.
- Examples of reward
  - Manage an investment portfolio: $+$ve reward for every € gained, $-$ve reward for every € lost.

Elements of a RL Problem : Reward II

Image source: *Chess.com*

- Dependent on the current state and the current action.
- The agent can only influence the reward through its actions.
- Examples of reward
    - Manage an investment portfolio : +ve reward for every € gained,
      −ve reward for every € lost.
    - Make a humanoid robot walk : +ve reward for every forward step,
      −ve reward for every fall.

Elements of a RL Problem : Reward II

Image source: *Chess.com*

- Dependent on the current state and the current action.
- The agent can only influence the reward through its actions.
- Examples of reward
  - Manage an investment portfolio : +ve reward for every € gained,
−ve reward for every € lost.
  - Make a humanoid robot walk : +ve reward for every forward step,
−ve reward for every fall.
  - Ad placement : +ve reward for every click,
−ve reward for every time it is not clicked.

Elements of a RL Problem : Reward III

Image source: *Chess.com*

- Rewards might be delayed.

Elements of a RL Problem : Reward III

Image source: *Chess.com*

- Rewards might be delayed.
- Greedy strategy : Make the locally optimal choice at each time step.

Elements of a RL Problem : Reward III

Image source: *Chess.com*

- Rewards might be delayed.
- Greedy strategy : Make the locally optimal choice at each time step.
- Being greedy might not work : sometimes better to sacrifice short term reward to gain more long-term reward.

Elements of a RL Problem : Policy

- The agent's way of behaving at a given time.

Elements of a RL Problem : Policy

- The agent's way of behaving at a given time.
- Mapping π from states to actions.

Elements of a RL Problem : Policy

- The agent's way of behaving at a given time.
- Mapping π from states to actions.
- Deterministic $\pi_s = a$.

Elements of a RL Problem : Policy

- The agent's way of behaving at a given time.
- Mapping π from states to actions.
- Deterministic $\pi_s = a$.
- Stochastic $\pi_s(a) = \mathbb{P}(a \mid s)$.

Elements of a RL Problem : Value

- Values of states specify what is good in the long run.

Elements of a RL Problem : Value

- Values of states specify what is good in the long run.
- Total amount of reward an agent can expect to accumulate over the future, starting from that state.

Elements of a RL Problem : Value

- Values of states specify what is good in the long run.
- Total amount of reward an agent can expect to accumulate over the future, starting from that state.
- Value v depends on the policy π.

Elements of a RL Problem : Value

- Values of states specify what is good in the long run.
- Total amount of reward an agent can expect to accumulate over the future, starting from that state.
- Value v depends on the policy π.
- $v_\pi(s) = \mathbb{E}[r(t+1) + r(t+2) + r(t+3) + \ldots \mid \pi, s(t) = s]$

Elements of a RL Problem : Value

- Values of states specify what is good in the long run.
- Total amount of reward an agent can expect to accumulate over the future, starting from that state.
- Value v depends on the policy π.
- $v_\pi(s) = \mathbb{E}[r(t+1) + r(t+2) + r(t+3) + \dots \mid \pi, s(t) = s]$
- Infinite (or very large) horizon T? Trade-off of importance between immediate vs long-term rewards - *discount factor* $\gamma \in (0, 1)$.

Elements of a RL Problem: Value

- Values of states specify what is good in the long run.
- Total amount of reward an agent can expect to accumulate over the future, starting from that state.
- Value v depends on the policy π.
- $v_\pi(s) = \mathbb{E}[r(t+1) + r(t+2) + r(t+3) + \dots \mid \pi, s(t) = s]$
- Infinite (or very large) horizon T? Trade-off of importance between immediate vs long-term rewards - *discount factor* $\gamma \in (0, 1)$.
- $v_\pi(s) = \mathbb{E}[r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \dots \mid \pi, s(t) = s]$

Elements of a RL Problem: Value

- Values of states specify what is good in the long run.
- Total amount of reward an agent can expect to accumulate over the future, starting from that state.
- Value v depends on the policy π.
- $v_\pi(s) = \mathbb{E}[r(t+1) + r(t+2) + r(t+3) + \ldots \mid \pi, s(t) = s]$
- Infinite (or very large) horizon T? Trade-off of importance between immediate vs long-term rewards - *discount factor* $\gamma \in (0, 1)$.
- $v_\pi(s) = \mathbb{E}[r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \ldots \mid \pi, s(t) = s]$
- Can be used to evaluate desirability of states and choose between actions.

Elements of a RL Problem : Model

- Model is a representation of the environment's behaviour.

Elements of a RL Problem : Model

- Model is a representation of the environment's behaviour.
- Reward function R provides the immediate reward given the current state and action. $R(s, a) = \mathbb{E}[r(t+1) \mid s(t) = s, a(t) = a]$.

Elements of a RL Problem : Model

- Model is a representation of the environment's behaviour.
- Reward function R provides the immediate reward given the current state and action. $R(s, a) = \mathbb{E}[r(t+1) \mid s(t) = s, a(t) = a]$.
- Transition function P provides the next state given the current state and action. $P(s' \mid s, a) = \mathbb{P}(s(t+1) = s' \mid s(t) = s, a(t) = a)$.

Elements of a RL Problem : Model

- Model is a representation of the environment's behaviour.
- Reward function R provides the immediate reward given the current state and action. $R(s, a) = \mathbb{E}[r(t+1) \mid s(t) = s, a(t) = a]$.
- Transition function P provides the next state given the current state and action. $P(s' \mid s, a) = \mathbb{P}(s(t+1) = s' \mid s(t) = s, a(t) = a)$.

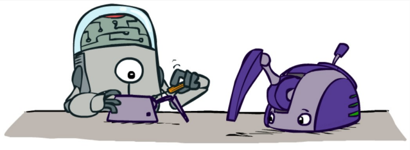


Figure 2: Model-based learning

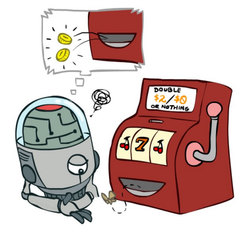


Figure 3: Model-free learning

An Example of a RL Problem I

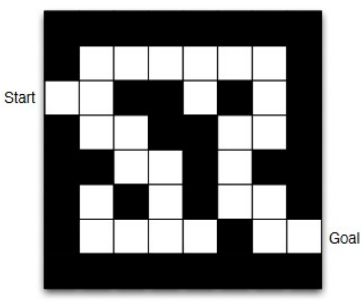


Image source: *DeepMind RL course*

- Move from start state to goal state as quickly as possible.

An Example of a RL Problem I

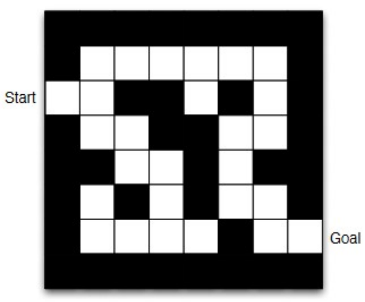


Image source: *DeepMind RL course*

- Move from start state to goal state as quickly as possible.
- Reward : -1 per time step.

An Example of a RL Problem I

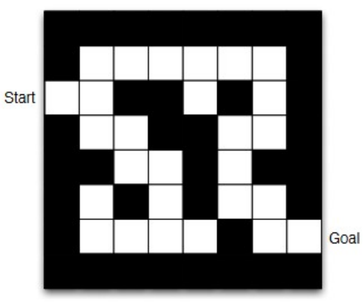


Image source: *DeepMind RL course*

- Move from start state to goal state as quickly as possible.
- Reward : -1 per time step.
- State : agent's location.

An Example of a RL Problem I

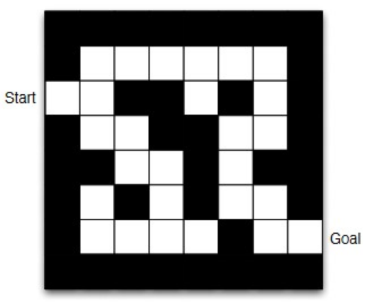


Image source: *DeepMind RL course*

- Move from start state to goal state as quickly as possible.
- Reward : -1 per time step.
- State : agent's location.
- Actions : $\uparrow$, $\downarrow$, $\leftarrow$, $\rightarrow$.

An Example of a RL Problem II

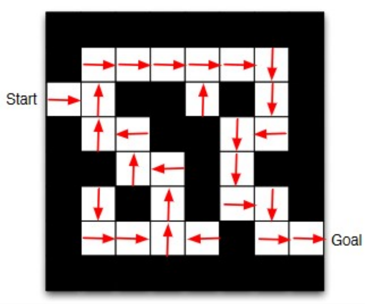


Image source: *DeepMind RL course*

- Arrows represent policy $\pi(s)$ for each state s.

An Example of a RL Problem III

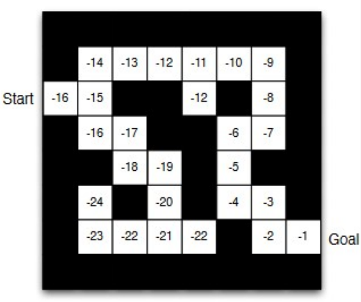


Image source: *DeepMind RL course*

- Numbers represent values $v_\pi(s)$ of each state s.

Break

We start again after a break.

Measuring the Performance: Optimal Value Function

- Recall that,

  undiscounted value for policy π is,

$$v_\pi(s) = \mathbb{E}[r(t+1) + r(t+2) + r(t+3) + \ldots \mid \pi, s(t) = s],$$

Measuring the Performance: Optimal Value Function

- Recall that,

  undiscounted value for policy π is,

  $$v_\pi(s) = \mathbb{E}[r(t+1) + r(t+2) + r(t+3) + \dots \mid \pi, s(t) = s],$$

  discounted value for policy π is,

  $$v_\pi(s) = \mathbb{E}[r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \dots \mid \pi, s(t) = s]$$

Measuring the Performance: Optimal Value Function

- Recall that,

  undiscounted value for policy π is,

$$v_\pi(s) = \mathbb{E}[r(t+1) + r(t+2) + r(t+3) + \ldots \mid \pi, s(t) = s],$$

  discounted value for policy π is,

$$v_\pi(s) = \mathbb{E}[r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \ldots \mid \pi, s(t) = s]$$

Definition

The optimal value function $v_*(s) = max_\pi v_\pi(s)$.

- The optimal value function specifies the best possible performance.

Measuring the Performance: Optimal Policy

- There exists an optimal policy π_* that is better than or equal to all other policies[1].

$$\pi_* \geq \pi, \forall \pi$$

where $\pi_1 \geq \pi_2$ if $v_{\pi_1}(s) \geq v_{\pi_2}(s), \forall s$

[1] For almost all the problems that we will encounter in this course.

Measuring the Performance: Optimal Policy

- There exists an optimal policy π_* that is better than or equal to all other policies[1].

$$\pi_* \geq \pi, \forall \pi$$

  where $\pi_1 \geq \pi_2$ if $v_{\pi_1}(s) \geq v_{\pi_2}(s), \forall s$

- An optimal policy π_* achieves the optimal value function $v_*(s)$.

$$v_{\pi_*}(s) = v_*(s).$$

[1] For almost all the problems that we will encounter in this course.

Measuring the Performance: Optimal Policy

- There exists an optimal policy π_* that is better than or equal to all other policies[1].

$$\pi_* \geq \pi, \forall \pi$$

  where $\pi_1 \geq \pi_2$ if $v_{\pi_1}(s) \geq v_{\pi_2}(s), \forall s$

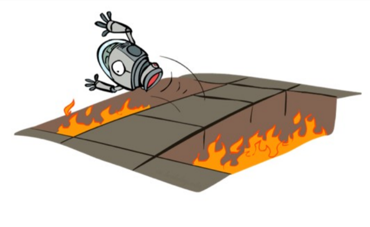
- An optimal policy π_* achieves the optimal value function $v_*(s)$.

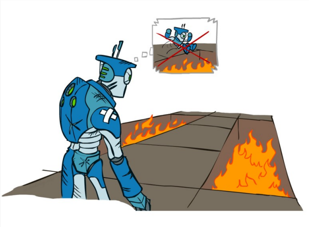
$$v_{\pi_*}(s) = v_*(s).$$

- A RL problem is "solved" when the agent finds an optimal policy.

[1] For almost all the problems that we will encounter in this course.

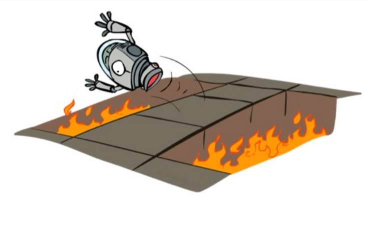
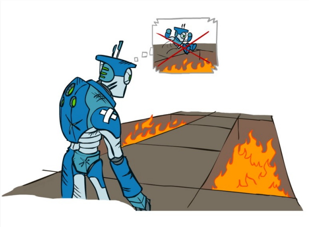
Measuring the Performance : Regret





- Even if the agent learns the optimal policy eventually, it still makes mistakes during the learning process.

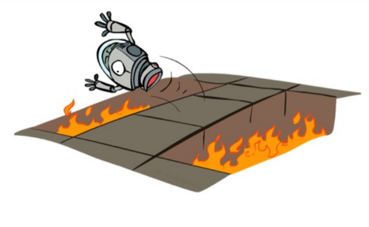
Measuring the Performance: Regret

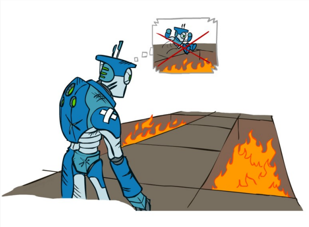



- Even if the agent learns the optimal policy eventually, it still makes mistakes during the learning process.
- Regret is the difference between the optimal (expected) rewards and the agent's (expected) rewards.

$$\text{Regret}_{\pi} = v_{*}(s) - v_{\pi}(s), \text{ where } s \text{ is the starting state}$$

Measuring the Performance : Regret



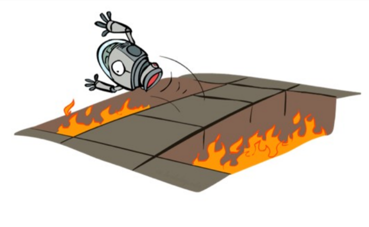


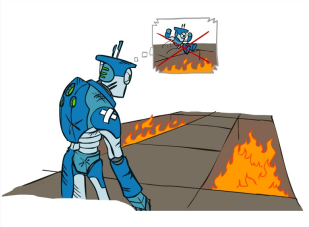
- Even if the agent learns the optimal policy eventually, it still makes mistakes during the learning process.
- Regret is the difference between the optimal (expected) rewards and the agent's (expected) rewards.

$$\text{Regret}_{\pi} = v_{*}(s) - v_{\pi}(s), \text{ where } s \text{ is the starting state}$$

- Regret is a measure of the total mistake cost.

Measuring the Performance : Regret





- Even if the agent learns the optimal policy eventually, it still makes mistakes during the learning process.
- Regret is the difference between the optimal (expected) rewards and the agent's (expected) rewards.

$$\text{Regret}_\pi = v_*(s) - v_\pi(s), \text{ where } s \text{ is the starting state}$$

- Regret is a measure of the total mistake cost.
- Minimizing regret equivalent to maximizing cumulative reward.

Non-associative RL : Multi-Armed Bandits

Multi-armed bandits

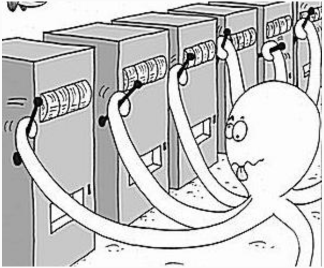


Image source: *Microsoft research*

- Learning to act in a single situation i.e. state.

Multi-armed bandits

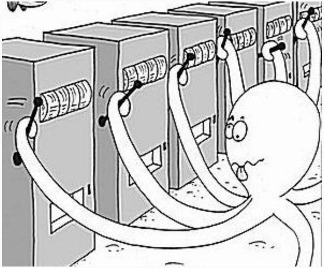


Image source: *Microsoft research*

- Learning to act in a single situation i.e. state.
- Agent faces repeated choice among K different actions/arms.

Multi-armed bandits

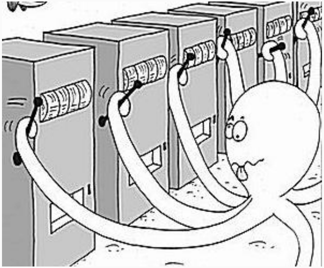


Image source: *Microsoft research*

- Learning to act in a single situation i.e. state.
- Agent faces repeated choice among K different actions/arms.
- After each choice, the learner receives a numerical reward.

Multi-armed bandits

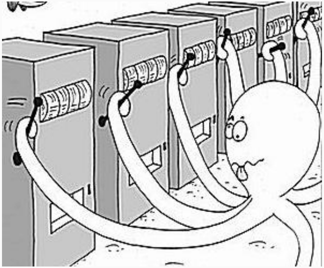


Image source: *Microsoft research*

- Learning to act in a single situation i.e. state.
- Agent faces repeated choice among K different actions/arms.
- After each choice, the learner receives a numerical reward.
- Goal : Maximize the cumulative reward or minimize the regret.

Stationary Stochastic Bandits

- Reward for arm a drawn i.i.d. from an unknown stationary distribution.

Stationary Stochastic Bandits

- Reward for arm a drawn i.i.d. from an unknown stationary distribution.
- Each action $a \in \mathcal{A}$ has expected or mean reward μ_a.

Stationary Stochastic Bandits

- Reward for arm a drawn i.i.d. from an unknown stationary distribution.
- Each action $a \in \mathcal{A}$ has expected or mean reward μ_a.
- Practical scenario : A doctor choosing between experimental drugs for a series of samples exhibiting a disease.

Stationary Stochastic Bandits

- Reward for arm a drawn i.i.d. from an unknown stationary distribution.
- Each action $a \in \mathcal{A}$ has expected or mean reward μ_a.
- Practical scenario: A doctor choosing between experimental drugs for a series of samples exhibiting a disease.

- A variant: Non-stationary stochastic bandits - rewards are drawn from distributions which may change over time.

Adversarial Bandits

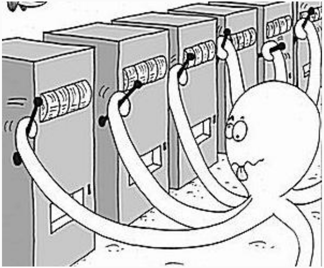


Image source: *Microsoft research*

- The assumption of stationary stochastic distributions is optimistic and sometimes unrealistic.

Adversarial Bandits

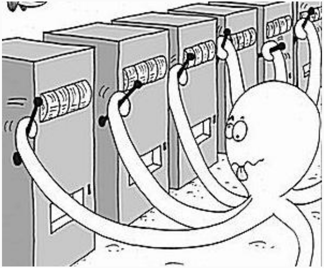


Image source: *Microsoft research*

- The assumption of stationary stochastic distributions is optimistic and sometimes unrealistic.
- Pessimistic assumption : rewards are chosen adversarially.

Adversarial Bandits

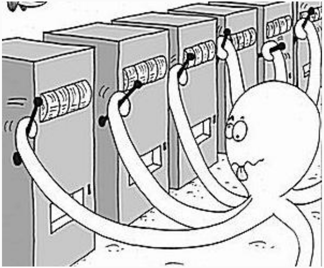


Image source: *Microsoft research*

- The assumption of stationary stochastic distributions is optimistic and sometimes unrealistic.
- Pessimistic assumption : rewards are chosen adversarially.
- Oblivious adversary : rewards for all arms and all rounds are chosen in advance.

Dueling Bandits

- Relative feedback not absolute feedback.

Dueling Bandits

- Relative feedback not absolute feedback.
- E.g. : *"I like football more than tennis"* instead of *"I value football at 48/50 and tennis at 33/50"*.

Dueling Bandits

- Relative feedback not absolute feedback.
- E.g. : *“I like football more than tennis”* instead of *“I value football at 48/50 and tennis at 33/50”*.

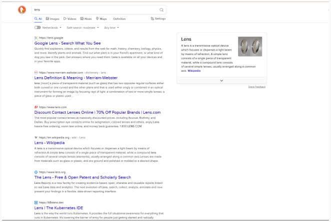


Figure 4: DuckDuckGo search results

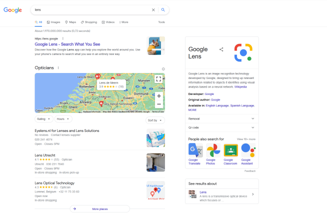


Figure 5: Google search results

- Practical scenario : Information retrieval in search engines.
- Relative feedback by interleaved filtering [Radlinski and Joachims, 2007]

Contextual Bandits

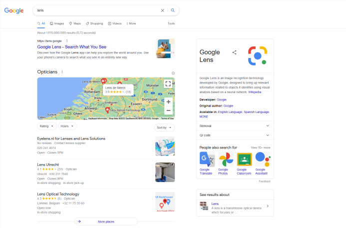


Figure 6: Google search results

- Observation of extra information (*context*) before choosing an action.
- Practical scenario : News recommendation, ad selection.

Associative RL: Markov Decision Processes

History and State

- History is the sequence of observations, actions and rewards.

$$\mathcal{F}_t = o(1), r(1), a(1), \ldots, a(t-1), o(t), r(t).$$

History and State

- History is the sequence of observations, actions and rewards.

$$\mathcal{F}_t = o(1), r(1), a(1), \ldots, a(t-1), o(t), r(t).$$

- Future depends on the history :
  - Agent selects an action $a(t)$.
  - Environment selects a reward/observation.

History and State

- History is the sequence of observations, actions and rewards.

$$\mathcal{F}_t = o(1), r(1), a(1), \ldots, a(t-1), o(t), r(t).$$

- Future depends on the history:
  - Agent selects an action $a(t)$.
  - Environment selects a reward/observation.
- State is the information used to determine what happens next.

History and State

- History is the sequence of observations, actions and rewards.

$$\mathcal{F}_t = o(1), r(1), a(1), \ldots, a(t-1), o(t), r(t).$$

- Future depends on the history:
    - Agent selects an action $a(t)$.
    - Environment selects a reward/observation.
- State is the information used to determine what happens next.
- Formally, state is a function of the history: $s(t) = f(\mathcal{F}_t)$.

Markov Property

- "The future is independent of the past given the present".

Andrey Markov(1856-1922)

Markov Property

- "The future is independent of the past given the present".
- The state $s(t)$ is Markov if and only if

$$\mathbb{P}(s(t+1) \,|\, s(t)) = \mathbb{P}(s(t+1) \,|\, s(t), s(t-1), \ldots, s(1)).$$

Andrey Markov(1856-1922)

Markov Property

- "The future is independent of the past given the present".
- The state $s(t)$ is Markov if and only if

$$\mathbb{P}(s(t+1) \mid s(t)) = \mathbb{P}(s(t+1) \mid s(t), s(t-1), \ldots, s(1)).$$

- The present state is a sufficient statistic of the future.

Andrey Markov(1856-1922)

Markov Process

A Markov process is a memory-less random process i.e. a sequence of random states $s(1), s(2), \ldots$ with the Markov property.

Definition

A Markov process (or a Markov chain) is a tuple $\langle \mathcal{S}, P \rangle$ *where*

- $\mathcal{S}$ *is a (finite) set of states, and*
- P *is a state transition probability function,*
  $P_{ss'} = \mathbb{P}[s(t+1) = s' \mid s(t) = s]$.

Markov Decision Process

A Markov decision process (MDP) is a Markov process with rewards and decisions.

Definition

A Markov decision process is a tuple $\langle \mathcal{S}, \mathcal{A}, R, P \rangle$ *where*

- $\mathcal{S}$ *is a (finite) set of states,*
- $\mathcal{A}$ *is a (finite) set of actions,*
- $R(s, a)$ *is a reward function,*
- P *is a state transition probability function,*
  $P^a_{ss'} = \mathbb{P}[s(t+1) = s' \mid s(t) = s, a(t) = a]$.

- Practical scenario : Learning to play chess.

Discounted-reward Markov Decision Process

A Markov decision process (MDP) is a Markov process with rewards and decisions.

Definition

A Markov decision process is a tuple $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$ *where*

- $\mathcal{S}$ *is a (finite) set of states,*
- $\mathcal{A}$ *is a (finite) set of actions,*
- $R(s, a)$ *is a reward function,*
- P *is a state transition probability function,* $P^a_{ss'} = \mathbb{P}[s(t+1) = s' \mid s(t) = s, a(t) = a]$, *and*
- $\gamma \in (0, 1)$ *is a discount factor.*

- Practical scenario: Portfolio management. Why discounted? Distant reward not as valuable as immediate reward due to inflation.

Summary

- Introduction to reinforcement learning.
- Mathematical formulation of a RL problem.
- Formulating RL with multi-armed bandits and its variants.
- Formulating RL with Markov decision processes.

Next Lecture

- Simple solutions to bandits (and why they are sub-optimal?)
- An optimal solution : Upper confidence bound (UCB) algorithm.
- Proving the performance bound for UCB.

References

Tor Lattimore and Csaba Szepesvari. *Bandit Algorithms*. Cambridge University Press, 2020. doi: 10.1017/9781108571401.

Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997. ISBN 0070428077.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley amp; Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779.

F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *KDD 2007*, pages 570–579. ACM, 2007. doi: 10.1145/1281192.1281254.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.