Outline

Machine Learning
- Supervised Learning v.s. Reinforcement Learning
- Reinforcement Learning v.s. Deep Learning

Introduction to Reinforcement Learning
- Agent and Environment
- Action, State, and Reward

Markov Decision Process

Reinforcement Learning Approach
- Value-Based
- Policy-Based
- Model-Based

Problems within RL
- Learning and Planning
- Exploration and Exploitation

RL for Unsupervised Model
Outline

Machine Learning
- Supervised Learning v.s. Reinforcement Learning
- Reinforcement Learning v.s. Deep Learning

Introduction to Reinforcement Learning
- Agent and Environment
- Action, State, and Reward

Markov Decision Process

Reinforcement Learning Approach
- Value-Based
- Policy-Based
- Model-Based

Problems within RL
- Learning and Planning
- Exploration and Exploitation

RL for Unsupervised Model
Machine Learning

Supervised Learning

Unsupervised Learning

Reinforcement Learning
Supervised v.s. Reinforcement Learning

Supervised Learning
- Training based on supervisor/label/annotation
- Feedback is instantaneous
- Time does not matter

Reinforcement Learning
- Training only based on reward signal
- Feedback is delayed
- Time matters
- Agent actions affect subsequent data
Supervised v.s. Reinforcement

Supervised
Learning from teacher

"Hello"
"Bye bye"
Say "Hi"
Say "Good bye"

Reinforcement
Learning from critics

Hello 😊
......
......
......

Bad
Reinforcement Learning

RL is a general purpose framework for **decision making**
- RL is for an *agent* with the capacity to *act*
- Each *action* influences the agent’s future *state*
- Success is measured by a scalar *reward* signal
- Goal: *select actions to maximize future reward*
Deep Learning

DL is a general purpose framework for **representation learning**
- Given an **objective**
- Learn **representation** that is required to achieve objective
- Directly from **raw inputs**
- Use minimal domain knowledge

\[
\begin{align*}
\text{vector} & : x = [x_1, x_2, \ldots, x_N] \\
\text{vector} & : y = [y_1, y_2, \ldots, y_M]
\end{align*}
\]
Deep Reinforcement Learning

AI is an agent that can solve human-level task
- RL defines the objective
- DL gives the mechanism
- RL + DL = general intelligence
Deep RL AI Examples

Play games: Atari, poker, Go, ...
Explore worlds: 3D worlds, ...
Control physical systems: manipulate, ...
Interact with users: recommend, optimize, personalize, ...
Introduction to RL

Reinforcement Learning
Outline

Machine Learning
- Supervised Learning v.s. Reinforcement Learning
- Reinforcement Learning v.s. Deep Learning

Introduction to Reinforcement Learning
- Agent and Environment
- Action, State, and Reward

Markov Decision Process

Reinforcement Learning Approach
- Value-Based
- Policy-Based
- Model-Based

Problems within RL
- Learning and Planning
- Exploration and Exploitation

RL for Unsupervised Model
Reinforcement Learning

RL is a general purpose framework for **decision making**
- RL is for an *agent* with the capacity to *act*
- Each *action* influences the agent’s future *state*
- Success is measured by a scalar *reward* signal

Big three: action, state, reward
Agent and Environment

observation $o_t$

action $a_t$

reward $r_t$

MoveRight

MoveLeft
Agent and Environment

At time step $t$

- The agent
  - Executes action $a_t$
  - Receives observation $o_t$
  - Receives scalar reward $r_t$

- The environment
  - Receives action $a_t$
  - Emits observation $o_{t+1}$
  - Emits scalar reward $r_{t+1}$

$t$ increments at env. step
State

Experience is the sequence of observations, actions, rewards

\[ O_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t \]

State is the information used to determine what happens next

- what happens depends on the history experience
  - The agent selects actions
  - The environment selects observations/rewards

The state is the function of the history experience

\[ s_t = f(O_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t) \]
Environment State

The environment state $s^e_t$ is the environment’s *private* representation:

- whether data the environment uses to pick the next observation/reward
- may not be visible to the agent
- may contain irrelevant information
Agent State

The agent state $s_t^a$ is the agent’s *internal* representation

- whether data the agent uses to pick the next action $\rightarrow$ information used by RL algorithms
- can be any function of experience
Information State

An information state (a.k.a. Markov state) contains all useful information from history.

A state is Markov iff \( P(s_{t+1} \mid s_t) = P(s_{t+1} \mid s_1, \ldots, s_t) \)

The future is independent of the past given the present:

\[
H_t = \{ o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t \}
\]

\[
H_{1:t} \rightarrow s_t \rightarrow H_{t+1:}\infty
\]

- Once the state is known, the history may be thrown away.
- The state is a sufficient statistics of the future.
Fully Observable Environment

Full observability: agent \textit{directly} observes environment state

\[ O_t = S^a_t = S^e_t \]

information state = agent state = environment state

This is a Markov decision process (MDP)
Partially Observable Environment

Partial observability: agent \textbf{indirectly} observes environment

\[ s^a_t \neq s^e_t \]

agent state ≠ environment state

This is partially observable Markov decision process (POMDP)

Agent must construct its own state representation \( s^a_t \)

- Complete history: \( s^a_t = H_t \)
- Beliefs of environment state: \( s^a_t = \{ P(s^c_t = s^1), \ldots, P(s^c_t = s^n) \} \)
- Hidden state (from RNN): \( s^a_t = \sigma(W_s \cdot s^a_{t-1} + W_o \cdot o_t) \)
Reward

Reinforcement learning is based on reward hypothesis

A reward $r_t$ is a scalar feedback signal
- Indicates how well agent is doing at step $t$

Reward hypothesis: all agent goals can be desired by maximizing expected cumulative reward
Sequential Decision Making

Goal: select actions to maximize total future reward

◦ Actions may have long-term consequences
◦ Reward may be delayed
◦ It may be better to sacrifice immediate reward to gain more long-term reward
Scenario of Reinforcement Learning

Observation
State
Agent
Don’t do that
Reward
Environment
Action
Change the environment
Scenario of Reinforcement Learning

Agent learns to take actions maximizing expected reward.
Machine Learning ≈ Looking for a Function

Observation → Actor/Policy

Function input → Action = \( \pi(\text{Observation}) \)

Reward → Used to pick the best function

Action output → Environment
Learning to Play Go

Observation → AlphaGo → Action → Reward → Next Move → Environment
Learning to Play Go

Agent learns to take actions maximizing expected reward.

Observation

Action

Environment

Reward

reward = 0 in most cases

If win, reward = 1

If loss, reward = -1
Learning to Play Go

Supervised Learning from teacher

Next move: “5-5”

Next move: “3-3”

Reinforcement Learning Learning from experience

First move ...... many moves ...... Win!

(Two agents play with each other.)

AlphaGo uses supervised learning + reinforcement learning.
Learning a Chatbot

Machine obtains feedback from user

Chatbot learns to maximize the *expected reward*
Learning a Chatbot

Let two agents talk to each other (sometimes generate good dialogue, sometimes bad)

How old are you?

See you.

See you.

See you.

How old are you?

I am 16.

I though you were 12.

What make you think so?
Learning a chat-bot

By this approach, we can generate a lot of dialogues. Use pre-defined rules to evaluate the goodness of a dialogue.

Machine learns from the evaluation as rewards
Learning to Play Video Game

Space invader: terminate when all aliens are killed, or your spaceship is destroyed

Score (reward)

Kill the aliens

Shield

Fire

Play yourself: http://www.2600online.com/spaceinvaders.html
How about machine: https://gym.openai.com/evaluations/eval_Eduozx4HRyqgTCVk9Itw
Learning to Play Video Game

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

Obtain reward $r_1 = 0$

Action $a_1$: “right”

Obtain reward $r_2 = 5$

Action $a_2$: “fire”

(kill an alien)

Usually there is some randomness in the environment
Learning to Play Video Game

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

After many turns

Game Over
(spaceship destroyed)

This is an episode.

Learn to maximize the expected cumulative reward per episode

Action $a_T$

Obtain reward $r_T$
More applications

**Flying Helicopter**
- [https://www.youtube.com/watch?v=0JL04JJjocc](https://www.youtube.com/watch?v=0JL04JJjocc)

**Driving**
- [https://www.youtube.com/watch?v=0xo1Ldx3L5Q](https://www.youtube.com/watch?v=0xo1Ldx3L5Q)

**Robot**
- [https://www.youtube.com/watch?v=370cT-OAzzM](https://www.youtube.com/watch?v=370cT-OAzzM)

**Google Cuts Its Giant Electricity Bill With DeepMind-Powered AI**

**Text Generation**
- [https://www.youtube.com/watch?v=pbQ4qe8EwLo](https://www.youtube.com/watch?v=pbQ4qe8EwLo)
Markov Decision Process

Fully Observable Environment
Outline

Machine Learning
- Supervised Learning v.s. Reinforcement Learning
- Reinforcement Learning v.s. Deep Learning

Introduction to Reinforcement Learning
- Agent and Environment
- Action, State, and Reward

Markov Decision Process

Reinforcement Learning Approach
- Value-Based
- Policy-Based
- Model-Based

Problems within RL
- Learning and Planning
- Exploration and Exploitation

RL for Unsupervised Model
Markov Process

Markov process is a memoryless random process
  ◦ i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property

Sample episodes from $S_1=C1$
  • C1 C2 C3 Pass Sleep
  • C1 FB FB C1 C2 Sleep
  • C1 C2 C3 Pub C2 C3 Pass Sleep
  • C1 FB FB C1 C2 C3 Pub
  • C1 FB FB FB C1 C2 C3 Pub C2 Sleep
Markov Reward Process (MRP)

Markov reward process is a Markov chain with values

- The return $G_t$ is the total discounted reward from time-step $t$

\[ G_t = r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \]
Markov Decision Process (MDP)

Markov decision process is a MRP with decisions

- It is an environment in which all states are Markov
Markov Decision Process (MDP)

$S$: finite set of states/observations

$A$: finite set of actions

$P$: transition probability

$R$: immediate reward

$\gamma$: discount factor

Goal is to choose policy $\pi$ at time $t$ that maximizes expected overall return:

$$\sum_{t' = t}^{T} \gamma^{t' - t} r_{t'}$$
Reinforcement Learning
Outline

Machine Learning
- Supervised Learning v.s. Reinforcement Learning
- Reinforcement Learning v.s. Deep Learning

Introduction to Reinforcement Learning
- Agent and Environment
- Action, State, and Reward

Markov Decision Process

Reinforcement Learning
- Value-Based
- Policy-Based
- Model-Based

Problems within RL
- Learning and Planning
- Exploration and Exploitation

RL for Unsupervised Model
Major Components in an RL Agent

An RL agent may include one or more of these components

- **Value function**: how good is each state and/or action
- **Policy**: agent’s behavior function
- **Model**: agent’s representation of the environment
Value Function

A value function is a prediction of future reward (with action $a$ in state $s$)

Q-value function gives expected total reward

- from state $s$ and action $a$
- under policy $\pi$
- with discount factor $\gamma$

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

Value functions decompose into a Bellman equation

$$Q^\pi(s, a) = \mathbb{E}_{s', a'}[r + \gamma Q^\pi(s', a') \mid s, a]$$
Optimal Value Function

An optimal value function is the maximum achievable value

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a) = Q^{\pi^*}(s, a)$$

The optimal value function allows us act optimally

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

The optimal value informally maximizes over all decisions

$$Q^*(s, a) = r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \ldots$$

$$= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$$

Optimal values decompose into a Bellman equation

$$Q^*(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s', a') \mid s, a]$$
Value Function Approximation

Value functions are represented by a lookup table

\[ Q(s, a) \quad \forall s, a \]

- too many states and/or actions to store
- too slow to learn the value of each entry individually

Values can be estimated with function approximation
Q-Networks

Q-networks represent value functions with weights $\mathbf{w}$

$$Q(s, a, w) \approx Q^*(s, a)$$

- generalize from seen states to unseen states
- update parameter $\mathbf{w}$ for function approximation
Q-Learning

Goal: estimate optimal Q-values

- Optimal Q-values obey a Bellman equation
  \[ Q^*(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s', a') \mid s, a] \]

- Value iteration algorithms solve the Bellman equation
  \[ Q_{i+1}(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q_i(s', a') \mid s, a] \]
Policy

A policy is the agent’s behavior

A policy maps from state to action

- Deterministic policy: \( a = \pi(s) \)
- Stochastic policy: \( \pi(a) = P(a \mid s) \)
Policy Networks

Represent policy by a network with weights $u$

$$a = \pi(a \mid s, u) \quad a = \pi(s, u)$$

stochastic policy  
deterministic policy

Objective is to maximize total discounted reward by SGD

$$L(u) = \mathbb{E} \left[ r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid \pi(\cdot, u) \right]$$
Policy Gradient

The gradient of a stochastic policy $\pi(a \mid s, u)$ is given by

$$\frac{\partial L(u)}{\partial u} = \mathbb{E}_s \left[ \frac{\partial \log \pi(a \mid s, u)}{\partial u} Q^\pi(s, a) \right]$$

The gradient of a deterministic policy $\pi(s, u)$ is given by

$$\frac{\partial L(u)}{\partial u} = \mathbb{E}_s \left[ \frac{\partial Q^\pi(s, a)}{\partial a} \frac{\partial a}{\partial u} \right] \quad a = \pi(s, u)$$

How to deal with continuous actions
Model

A model predicts what the environment will do next

- \( P \) predicts the next state
  \[
P_{ss'}^a = \mathbb{P}[s_{t+1} = s' \mid s_t = s, a_t = a]
  \]

- \( R \) predicts the next immediate reward
  \[
  R_s^a = \mathbb{E}[r_{t+1} \mid s_t = s, a_t = a]
  \]
Reinforcement Learning Approach

Value-based RL
- Estimate the optimal value function $Q^*(s, a)$

Policy-based RL
- Search directly for optimal policy $\pi^*$

Model-based RL
- Build a model of the environment
- Plan (e.g. by lookahead) using model

$Q^*(s, a)$ is maximum value achievable under any policy

$\pi^*$ is the policy achieving maximum future reward
Maze Example

Rewards: -1 per time-step
Actions: N, E, S, W
States: agent’s location
Maze Example: Value Function

Rewards: -1 per time-step
Actions: N, E, S, W
States: agent’s location

Numbers represent value $Q_\pi(s)$ of each state $s$
Maze Example: Policy

Rewards: -1 per time-step
Actions: N, E, S, W
States: agent’s location

Arrows represent policy $\pi(s)$ for each state $s$
Maze Example: Value Function

Rewards: -1 per time-step
Actions: N, E, S, W
States: agent’s location

Grid layout represents transition model $P$
Numbers represent immediate reward $R$ from each state $s$ (same for all $a$)
### Categorizing RL Agents

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value-Based</strong></td>
<td></td>
</tr>
<tr>
<td>- No Policy (implicit)</td>
<td></td>
</tr>
<tr>
<td>- Value Function</td>
<td></td>
</tr>
<tr>
<td><strong>Policy-Based</strong></td>
<td></td>
</tr>
<tr>
<td>- Policy</td>
<td></td>
</tr>
<tr>
<td>- No Value Function</td>
<td></td>
</tr>
<tr>
<td><strong>Actor-Critic</strong></td>
<td></td>
</tr>
<tr>
<td>- Policy</td>
<td></td>
</tr>
<tr>
<td>- Value Function</td>
<td></td>
</tr>
<tr>
<td><strong>Model-Free</strong></td>
<td></td>
</tr>
<tr>
<td>- Policy and/or Value Function</td>
<td></td>
</tr>
<tr>
<td>- No Model</td>
<td></td>
</tr>
<tr>
<td><strong>Model-Based</strong></td>
<td></td>
</tr>
<tr>
<td>- Policy and/or Value Function</td>
<td></td>
</tr>
<tr>
<td>- Model</td>
<td></td>
</tr>
</tbody>
</table>
RL Agent Taxonomy
Problems within RL
Outline

Machine Learning
- Supervised Learning v.s. Reinforcement Learning
- Reinforcement Learning v.s. Deep Learning

Introduction to Reinforcement Learning
- Agent and Environment
- Action, State, and Reward

Markov Decision Process

Reinforcement Learning
- Value-Based
- Policy-Based
- Model-Based

Problems within RL
- Learning and Planning
- Exploration and Exploitation

RL for Unsupervised Model
Learning and Planning

In sequential decision making

- Reinforcement learning
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy

- Planning
  - A model of the environment is known
  - The agent performs computations with its model (w/o any external interaction)
  - The agent improves its policy (a.k.a. deliberation, reasoning, introspection, pondering, thought, search)
Atari Example: Reinforcement Learning

Rules of the game are unknown
Learn directly from interactive game-play
Pick actions on joystick, see pixels and scores
Atari Example: Planning

Rules of the game are known

Query emulator based on the perfect model inside agent’s brain
  ◦ If I take action $a$ from state $s$:
    • what would the next state be?
    • what would the score be?

Plan ahead to find optimal policy e.g. tree search
Exploration and Exploitation

Reinforcement learning is like trial-and-error learning.

The agent should discover a good policy from the experience without losing too much reward along the way.

*Exploration* finds more information about the environment.

*Exploitation* exploits known information to maximize reward.

When to try?

It is usually important to explore as well as exploit.
Outline

Machine Learning
  • Supervised Learning v.s. Reinforcement Learning
  • Reinforcement Learning v.s. Deep Learning

Introduction to Reinforcement Learning
  • Agent and Environment
  • Action, State, and Reward

Markov Decision Process

Reinforcement Learning
  • Policy-Based
  • Value-Based
  • Model-Based

Problems within RL
  • Learning and Planning
  • Exploration and Exploitation

RL for Unsupervised Model
RL for Unsupervised Model: Modularizing Unsupervised Sense Embeddings (MUSE)
Word2Vec Polysemy Issue

Words are polysemy
- An *apple* a day, keeps the doctor away.
- Smartphone companies including *apple*, ...

If words are polysemy, are their embeddings polysemy?
- No 😞
- What’s the problem?
Modular Framework

Two key mechanisms
- **Sense selection** given a text context
- **Sense representation** to embed statistical characteristics of sense identity

Smartphone companies including **apple** blackberry, and sony will be invited.

[Diagram of modular framework with labels: sense embedding, sense selection, apple-1, apple-2, sense representation, reinforcement learning, sense identity]
Sense Selection Module

Input: a text context $\overline{C_t} = [C_{t-m}, ..., C_t = w_i, ..., C_{t+m}]$

Output: the fitness for each sense $z_{i1}, ..., z_{i3}$

Model architecture: Continuous Bag-of-Words (CBOW) for efficiency

Sense selection

- Policy-based
  $$\pi(z_{ik} \mid \overline{C_t}) = \frac{\exp(Q^T_{ik} \sum_j \overline{C_t} P_j)}{\sum_{k' \in Z_i} \exp(Q^T_{ik'} \sum_j \overline{C_t} P_j)}$$

- Value-based (Q-value)
  $$q(z_{ik} \mid \overline{C_t}) = \sigma(Q^T_{ik} \sum_{j \in \overline{C_t}} P_j)$$

Diagram: Sense selection for target word $C_t$
Sense Representation Module

Input: sense collocation $z_{ik}, z_{jl}$

Output: collocation likelihood estimation

Model architecture: skip-gram architecture

Sense representation learning

$$\log \mathcal{L}(z_{jl} | z_{ik}) = \log \sigma(U^T_{z_{ik}} V z_{jl})$$

$$+ \sum_{v=1}^{M} \mathbb{E}_{z_{uv} \sim p_{neg}(z)}[\log \sigma(-U^T_{z_{ik}} V z_{uv})]$$
A Summary of MUSE

The first purely sense-level embedding learning with efficient sense selection.

Corpus: { Smartphone companies including apple, blackberry, and sony will be invited.}
## Qualitative Analysis

<table>
<thead>
<tr>
<th>Context</th>
<th>... braves finish the season in <strong>tie</strong> with the los angeles dodgers ...</th>
<th>... his later years proudly wore <strong>tie</strong> with the chinese characters for ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-NN</td>
<td>scoreless otl shootout 6-6 hingis 3-3 7-7 0-0</td>
<td>pants trousers shirt juventus blazer socks anfield</td>
</tr>
<tr>
<td>Figure</td>
<td><img src="image.png" alt="image of scoreboard" /></td>
<td><img src="image.png" alt="image of ties" /></td>
</tr>
</tbody>
</table>
## Qualitative Analysis

<table>
<thead>
<tr>
<th>Context</th>
<th>... of the mulberry or the <strong>blackberry</strong> and minos sent him to ...</th>
<th>... of the large number of <strong>blackberry</strong> users in the us federal ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-NN</td>
<td>cranberries maple vaccinium apricot apple</td>
<td>smartphones sap microsoft ipv6 smartphone</td>
</tr>
<tr>
<td>Figure</td>
<td><img src="image1.png" alt="Cranberries" /></td>
<td><img src="image2.png" alt="Blackberry Smartphone" /></td>
</tr>
</tbody>
</table>
Concluding Remarks

RL is a general purpose framework for **decision making** under interactions between *agent* and *environment*

- RL is for an *agent* with the capacity to *act*
- Each *action* influences the agent’s future *state*
- Success is measured by a scalar *reward* signal
- Goal: *select actions to maximize future reward*

An RL agent may include one or more of these components

- **Value function**: how good is each state and/or action?
- **Policy**: agent’s behavior function
- **Model**: agent’s representation of the environment
References

Course materials by David Silver: http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html