Deep Learning

Columbia University
Iddo Drori, Summer 2019
Lectures

- Lecture 12 (Thursday, July 25): Graph Neural Networks
- Lecture 13 (Monday, July 29): Adversarial Examples, Training Poisoning, DMI
- **Lecture 14 (Tuesday, July 30): Reinforcement Learning**
- Lecture 15 (Wednesday, July 31): Reinforcement Learning
- Lecture 16 (Thursday, August 1): Value-based Reinforcement Learning
- Lecture 17 (Monday, August 5): Model-based Reinforcement Learning
- Lecture 18 (Tuesday, August 6): Policy-based Reinforcement Learning
- Lecture 19 (Wednesday, August 7): Deep Reinforcement Learning
- Lecture 20 (Thursday, August 8): Deep Reinforcement Learning
- Lecture 21 (Monday, August 12): Automatic Deep Learning
- Lecture 22 (Tuesday, August 13): Deep Learning for Self Driving Cars
- Lecture 23 (Wednesday, August 14): Quantum Neural Networks
- Lecture 24 (Thursday, August 15): Project Presentations
Reinforcement Learning
Machine Learning

- Supervised
- Unsupervised
- Reinforcement
Reinforcement Learning

- Learning from interaction to achieve goal
- Agent: learner, decision maker
- Environment: what the agent interacts with

- Agent selects actions, environment responds to actions with new state and reward
- Agent tries to maximize rewards over time
Reinforcement Learning

![Diagram of Reinforcement Learning](image)

- **Agent**: $a_t$
- **Environment**: $s_t$, $r_t$

The diagram illustrates the interaction between the agent and the environment in a reinforcement learning setting.
Applications

- Self driving vehicles, unmanned aerial vehicles, ship steering
- Robot walking, grasping
- Game playing
- Portfolio management
- ...

Deep Reinforcement Learning Applications

- Video games
- Board games
- Rubik’s cube
- Protein folding
- Dialogue synthesis
- Automatic machine learning
- Robot control
- Self driving cars
Multi-Armed Bandit

Stateless

- Action: pull one of $k$ arms
- Reward for pulling that arm

at each time step $t$:

  - choose action $a_t$ among $k$ actions
  - receive reward $r_t$ for taking action $a_t$

- Taking action $a$ is pulling arm $i$ which gives reward $r(a)$ with probability $p_i$
- Probabilities distributions $p_1, ..., p_k$ are unknown
- Goal is to maximize total expected return
Multi-Armed Bandit

- Value of action $a$ is expected reward: $Q^*(a) = E[r_t | a_t = a]$
  we don’t know the action values

- Estimate value of action $a$ at time $t$: $Q_t(a)$
- For example keep current mean reward for each action
Greedy Action

- A greedy action takes the best estimate at time $t$, exploiting knowledge
  
  $a_t = \text{argmax}_a Q_t(a)$

- For example choose action with largest mean reward.

- A non-greedy action is exploring.
Greedy Action Selection

\[ v(0) = 0 \]
\[ v(1) = 1 \]
\[ v(2 \text{ mean}) = 2 \]
\[ v(3) = 2 \]

Which to choose next?
**ɛ-Greedy**

- Behave greedily most of the time:

  with probability $\varepsilon$ choose random action
  with probability $1 - \varepsilon$ choose greedy action

Initialization: for each action $a$

$$Q(a) = 0$$

$$N(a) = 0 \ # \text{number of times action is chosen}$$

Loop: for each time step

$$a = \arg\max_a Q(a) \ \text{with probability} \ 1 - \varepsilon; \ \text{random action with probability} \ \varepsilon$$

$$N(a) += 1$$

$$Q(a) = Q(a) + \frac{r(a) - Q(a)}{N(a)}$$
Upper Confidence Bound (UCB)

- Optimism in face of uncertainty
- Use both mean and variance of reward

\[ \text{argmax}_a [\mu(r(a)) + \varepsilon \sigma(r(a))] \]

- Finite-time Analysis of the Multiarmed Bandit Problem, Auer et al, Machine Learning, 2012
- Used in Monte Carlo tree search (MCTS), in expert iteration and AlphaZero.
Markov Model

State

$s_1$ $\rightarrow$ $s_2$ $\Rightarrow$ $s_3$ $\Rightarrow$ $s_4$

$p(s_2|s_1)$ $p(s_3|s_2)$ $p(s_4|s_3)$
Markov Process

\[ p(s_2/s_1, a_1) \]

\[ p(s_3/s_2, a_2) \]

\[ p(s_4/s_3, a_3) \]
Markov Process

\[ \pi(a_1|s_1) \]
\[ \pi(a_2|s_2) \]
\[ \pi(a_3|s_3) \]
\[ \pi(a_4|s_4) \]

\[ p(s_2|s_1, a_1) \]
\[ p(s_3|s_2, a_2) \]
\[ p(s_4|s_3, a_3) \]
Markov Process

\[ p(s_2, a_2 | s_1, a_1) \]
\[ p(s_3, a_3 | s_2, a_2) \]
\[ p(s_4, a_4 | s_3, a_3) \]
Markov Process

\[(s_{t+k}, a_{t+k}) = T^k(s_t, a_t)\]
Markov Process
Reinforcement Learning

agent

environment

$\alpha_t$

$\gamma_t$

$\beta_t$

$\delta_t$

$\sigma_t$

$\sigma_e$

$0_t$

$S_t$
Markov Process
Markov Process

\[
\pi(a_i | o_i)
\]
Reinforcement Learning Policy

At each time step $t$, the agent implements the mapping $\pi_t$ from states to probabilities of selecting each action.

Mapping is agent's policy $\pi_t(a|s) = p(a_t = a | s_t = s)$.
Reinforcement Learning Policy

How agent selects actions map from state to action at time $t$:

- deterministic: $a = \pi_t(s)$
- stochastic: $\pi_t(a|s) = p(a_t = a | s_t = s)$
State Value Function

\[ V_{\pi}(s) \]

is the state-value function for policy \( \pi \),

how good it is for agent to be at in given state, in terms of expected future rewards.

Rewards also depend on agent actions, so value function defined with respect to agent policy \( \pi \):

\[
V_{\pi}(s) = E_{\pi} \left( \sum_{t=0}^{\infty} \gamma^t r_{t+k} \mid s_t = s \right) = E_{\pi} \left( \sum_{t=0}^{\infty} \gamma^t r_{t+k} \mid s_t = s \right)
\]
Action-Value Function

The value of taking action \( a \) in state \( s \) under policy \( \pi \) is the expected return:

\[
q_\pi(s, a) = \mathbb{E}_\pi \left( \sum_{t=0}^{\infty} \gamma^t R_t \mid s_0 = s, a_0 = a \right) = \mathbb{E}_\pi \left( \sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a \right)
\]
Value Functions

State value function $V_{\pi}(s)$

Action value function $Q_{\pi}(s,a)$

$Q_{\pi}(s, \text{left})$ $Q_{\pi}(s, \text{right})$ $Q_{\pi}(s, \text{no-go})$
Reinforcement Learning Return

\[ G_t = r_{t+1} + r_{t+2} + \cdots + r_T \]

Discounted return:
\[ G_t = r_{t+1} + r_{t+2} + \cdots + \sum_{k=0}^{T-t} \delta^k r_{t+k+1} \]

If \( \delta = 0 \) then agent is myopic, maximizing only immediate rewards.
As \( \delta \to 1 \) agent becomes far-sighted.
\[ g_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = r_{t+1} + \gamma (r_{t+2} + \gamma r_{t+3} + \cdots) = r_{t+1} + \gamma g_{t+1} \]
Model

```
predicts what the environment will do next

transitions $P$ predicts next state
rewards $R$ predicts next reward

model is optional, can be model-free method
```
Reinforcement Learning

- $a_t$: agent action: fight or flight, joystick left or right
- $o_t$: agent observation: bushes, game pixels
- $S^e_t$: environment state: included line, other 1024 bit instructions usually not visible to agent
- $S^a_t$: agent state

```
agent
S^a_t

environment
S^e_t
```

History:
$h_t = a_1, o_1, r_1, a_2, o_2, r_2, \ldots, a_t, o_t, r_t$

Agent state:
$S^a_t = f(h_t)$ summary of experience

Algorithm:
$h_t \rightarrow a_{t+1}$
Reinforcement Learning

http://www.youtube.com/watch?v=PSQt5KGv7Vk
Reinforcement Learning

Fully observable:
\[ q_t = s_t^a = s_t \]

Partially observable:
\[ s_t^a \neq s_t \]

Agent state
\[ s_t^a = f(h_t) \]

\[ s_t^a = (p(s_t^a = s_1), p(s_t^a = s_2), ..., p(s_t^a = s_n)) \]

RNN
\[ s_t^a = \sigma \left( WS_{t-1}^a + Uo_t \right) \]
Reinforcement Learning Agents

Value based
- no policy, value function

Policy based
- no value function, policy

Actor critic
- both value and policy based

Model free
- no model, just policy and value
Reinforcement Learning Algorithms

Source: spinningup.openai.com/en/latest/spinningup/rl_intro2.html
Model Based Reinforcement Learning

Source: Model based reinforcement learning for Atari, Kaizer, 2019
Deep Reinforcement Learning

- Deep neural network represents policy, value function, model
- Optimize loss function by stochastic gradient descent
Problems

Reinforcement Learning:
- environment is unknown

Planning:
- environment is known
- action $a$ $\rightarrow$ state $s$, reward $r$

Exploration vs. exploitation:
- Exploration - find more about environment
- Exploitation - use known information to maximize reward

example: show new ad vs. show best ad
Markov Property

\[
p(s_{t+1} | s_t) = p(s_{t+1} | s_1, \ldots, s_t)
\]

\[
p(s_{t+1}, r_{t+1} | s_t, a_t) = p(s_{t+1}, r_{t+1} | s_t, a_t, \ldots, s_t, a_t)
\]

\(s_t^e\) and \(h_t\) are Markov

\(h_{t-t} \rightarrow s_t \rightarrow h_{t+1} \ldots\)
Agent Representation of State

![Diagram](https://via.placeholder.com/150)

- \( r=100 \)
- \( r=-100 \)
- ?
Markov Decision Process (MDP)

**Defined by the probabilities:**

\[ p(s', r | s, a) = p(s'_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a) \]

\[ \sum_{s'} \sum_{r} p(s', r | s, a) = 1 \quad \forall (s, a) \]

**Expected rewards for state-action pairs:**

\[ r(s, a) = E(r_{t+1} | s_t = s, a_t = a) = \sum_{r} \sum_{s'} p(s', r | s, a) \]

**State-transition probabilities:**

\[ p(s' | s, a) = p(s'_{t+1} = s' | s_t = s, a_t = a) = \sum_{s'} \sum_{r} p(s', r | s, a) \]

**Expected rewards for state-action-next state:**

\[ r(s, a, s') = E(r_{t+1} | s_t = s, a_t = a, s'_{t+1} = s') = \sum_{r} \sum_{s'} r p(s', r | s, a) \]
Markov Decision Process (MDP): Example 1

\[ S = \{ \text{high, low} \} \text{ battery levels} \]

\[ A(\text{high}) = \{ \text{search, wait} \} \]

\[ A(\text{low}) = \{ \text{search, wait, recharge} \} \]

Source: Reinforcement Learning, Sutton and Barto, 2nd Ed. 2018.
Markov Decision Process (MDP)

\[ S = \{ \text{high, low} \} \text{ battery levels} \]

\[ A(\text{high}) = \{ \text{search, wait} \} \]

\[ A(\text{low}) = \{ \text{search, wait, recharge} \} \]

Source: Reinforcement Learning, Sutton and Barto, 2nd Ed. 2018.
Markov Decision Process (MDP)

Source: Reinforcement Learning, Sutton and Barto, 2nd Ed. 2018.
Markov Decision Process (MDP): Example 2

$s = \{\text{standing, moving, fallen}\} \quad a = \{\text{slow, fast}\} \quad \text{probability reward}$
MDP: Reward $R(s,a)$

$$R(s, a) = \begin{bmatrix} \frac{1}{5} & 0 \\ 1 & \frac{4}{5} \\ 1 & \frac{7}{5} \end{bmatrix}$$

optimal myopic policy
horizon 1

$$R(s, a) = \begin{bmatrix} \frac{3}{5} \cdot (-1) + \frac{2}{5} \cdot 1 \\ 1 \cdot 1 \\ 1 \cdot 1 \end{bmatrix}$$

slow

fallen

standing

moving

fast

$$R(s, a) = \begin{bmatrix} 1 \cdot 0 \\ \frac{3}{5} \cdot 2 + \frac{2}{5} \cdot (-1) \\ \frac{4}{5} \cdot 1 + \frac{1}{5} \cdot (-1) \end{bmatrix}$$
MDP: Transition $P(s,a,s')$

$$P(s, \text{slow}, s') = \begin{bmatrix} 3/5 & 2/5 & 0 \\ 2/5 & 3/5 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P(s, \text{fast}, s') = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 3 \\ 1/5 & 0 & 4/5 \end{bmatrix}$$
Value Function for Policy $\pi$

- Expected return starting from $s$ and following $\pi$

$$v_\pi(s) = \mathbb{E}_\pi(g_t | s_t = s) = \mathbb{E}_\pi(\sum_k \gamma^k r_{t+k+1} | s_t = s)$$
• Expected return starting from $s$ taken action $a$ and following $\pi$

$$q_\pi(s, a) = \mathbb{E}_\pi(g_t | s_t = s, a_t = a) = \mathbb{E}_\pi(\sum_k \gamma^k r_{t+k+1} | s_t = s, a_t = a)$$
Thank you