Introduction of Reinforcement Learning
Deep Reinforcement Learning: $AI = RL + DL$
Reference

- Textbook: Reinforcement Learning: An Introduction

- Lectures of David Silver
  - http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html (10 lectures, around 1:30 each)
  - http://videolectures.net/rldm2015_silver_reinforcement_learning/ (Deep Reinforcement Learning)

- Lectures of John Schulman
  - https://youtu.be/aUrX-rP_ss4
Scenario of Reinforcement Learning

Agent

Observation

State

Action

Change the environment

Reward

Environment

Don’t do that

Scenario of Reinforcement Learning

Agent

Observation

State

Action

Change the environment

Reward

Environment

Don’t do that
Scenario of Reinforcement Learning

Agent learns to take actions maximizing expected reward.

Observation → Agent → State → Action → Environment → Reward → Thank you.

https://yoast.com/how-to-clean-site-structure/
Machine Learning ≈ Looking for a Function

Action = $\pi(\text{Observation})$

Observation → Actor/Policy → Action

Function input: Used to pick the best function

Function output: Reward

Environment
Learning to play Go

Observation

Action

Reward

Next Move

Environment
Learning to play Go

Agent learns to take actions maximizing expected reward.

Reward

If win, reward = 1
If loss, reward = -1

reward = 0 in most cases

Environment

Observation

Action

AlphaGo
Learning to play Go

• Supervised: Learning from teacher

  Next move: “5-5”

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

  (Two agents play with each other.)

• Reinforcement Learning Learning from experience

  Next move: “3-3”

Alpha Go is supervised learning + reinforcement learning.
Learning a chat-bot

• Machine obtains feedback from user

• Chat-bot learns to maximize the **expected reward**
Learning a chat-bot

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

How old are you?

See you.

How old are you?

I am 16.

See you.

I though you were 12.

See you.

What make you think so?
Learning a chat-bot

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

Machine learns from the evaluation

Deep Reinforcement Learning for Dialogue Generation
Learning a chat-bot

- Supervised
  - "Hello"
  - "Bye bye"
  - Say "Hi"
  - Say "Good bye"

- Reinforcement

Agent

Agent

Hello 😊

......

......

......

Bad
More applications

• Flying Helicopter
  • https://www.youtube.com/watch?v=0JL04JJjocc

• Driving
  • https://www.youtube.com/watch?v=0xo1Ldx3L5Q

• Robot
  • 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
Example: Playing Video Game

• Widely studies:
  • Gym: https://gym.openai.com/
  • Universe: https://openai.com/blog/universe/

Machine learns to play video games as human players

➢ What machine observes is pixels

➢ Machine learns to take proper action itself
Example: Playing Video Game

• Space invader

Termination: all the aliens are killed, or your spaceship is destroyed.
Example: Playing Video Game

• Space invader
  • Play yourself:
    http://www.2600online.com/spaceinvaders.html
  • How about machine:
    https://gym.openai.com/evaluations/eval_Eduo
    zx4HRyqqTCVkJltw
Example: Playing Video Game

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

Example: Playing 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
Example: Playing Video Game

Start with observation $s_1$

After many turns

Observation $s_2$

Game Over (spaceship destroyed)

Observation $s_3$

This is an episode.

Learn to maximize the expected cumulative reward per episode

Obtain reward $r_T$

Action $a_T$
Properties of Reinforcement Learning

• **Reward delay**
  - In space invader, only “fire” obtains reward
    - Although the moving before “fire” is important
  - In Go playing, it may be better to sacrifice immediate reward to gain more long-term reward

• Agent’s actions **affect the subsequent data it receives**
  - E.g. Exploration
Outline

Alpha Go: policy-based + value-based + model-based

Model-free Approach

Model-based Approach

- Policy-based
  - Learning an Actor
- Value-based
  - Actor + Critic
  - Learning a Critic
Policy-based Approach

Learning an Actor
Three Steps for Deep Learning

Step 1: Define a set of functions

Step 2: Goodness of function

Step 3: Pick the best function

Deep Learning is so simple …..
Neural network as Actor

- Input of neural network: the observation of machine represented as a vector or a matrix
- Output neural network: each action corresponds to a neuron in output layer

What is the benefit of using network instead of lookup table? generalization
Three Steps for Deep Learning

Step 1: Neural Network as Actor

Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ......
Goodness of Actor

• Review: Supervised learning

Training Example

Given a set of parameters $\theta$

Softmax

Loss $l$

As close as possible

Target

Total Loss:

$$L = \sum_{n=1}^{N} l_n$$
Goodness of Actor

- Given an actor $\pi_\theta (s)$ with network parameter $\theta$
- Use the actor $\pi_\theta (s)$ to play the video game
  - Start with observation $s_1$
  - Machine decides to take $a_1$
  - Machine obtains reward $r_1$
  - Machine sees observation $s_2$
  - Machine decides to take $a_2$
  - Machine obtains reward $r_2$
  - Machine sees observation $s_3$
  - ......
  - Machine decides to take $a_T$
  - Machine obtains reward $r_T$

Total reward: $R_\theta = \sum_{t=1}^{T} r_t$

Even with the same actor, $R_\theta$ is different each time

Randomness in the actor and the game

We define $\bar{R}_\theta$ as the \textit{expected value} of $R_\theta$

$\bar{R}_\theta$ evaluates the goodness of an actor $\pi_\theta (s)$
Goodness of Actor

We define $\bar{R}_\theta$ as the expected value of $R_\theta$

- $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_T, a_T, r_T\}$

$$P(\tau|\theta) = p(s_1)p(a_1|s_1, \theta)p(r_1, s_2|s_1, a_1)p(a_2|s_2, \theta)p(r_2, s_3|s_2, a_2)\ldots$$

$$= p(s_1) \prod_{t=1}^{T} p(a_t|s_t, \theta)p(r_t, s_{t+1}|s_t, a_t)$$

$p(a_t = "fire"|s_t, \theta) = 0.7$

Control by your actor $\pi_\theta$

Actor $\pi_\theta$

$s_t$ left 0.1

右 0.2

火 0.7

not related to your actor
Goodness of Actor

- An episode is considered as a trajectory $\tau$
  - $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_T, a_T, r_T\}$
  - $R(\tau) = \sum_{t=1}^{T} r_t$
- If you use an actor to play the game, each $\tau$ has a probability to be sampled
  - The probability depends on actor parameter $\theta$:
    - $P(\tau | \theta)$

$$
\bar{R}_\theta = \sum_\tau R(\tau) P(\tau | \theta) \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n)
$$

Sum over all possible trajectory

Use $\pi_\theta$ to play the game N times, obtain $\{\tau^1, \tau^2, \ldots, \tau^N\}$
Sampling $\tau$ from $P(\tau | \theta)$ N times
Three Steps for Deep Learning

1. **Step 1:** Neural Network as Actor
2. **Step 2:** Goodness of Function
3. **Step 3:** Pick the best function

Deep Learning is so simple ......
Gradient Ascent

• Problem statement

\[ \theta^* = \arg \max_\theta \bar{R}_\theta \]

• Gradient ascent
  • Start with \( \theta^0 \)
  • \( \theta^1 \leftarrow \theta^0 + \eta \nabla \bar{R}_{\theta^0} \)
  • \( \theta^2 \leftarrow \theta^1 + \eta \nabla \bar{R}_{\theta^1} \)
  • ......
Policy Gradient

\[
\bar{R}_\theta = \sum_{\tau} R(\tau)P(\tau|\theta) \quad \nabla \bar{R}_\theta = ?
\]

\[
\nabla \bar{R}_\theta = \sum_{\tau} R(\tau)\nabla P(\tau|\theta) = \sum_{\tau} R(\tau)P(\tau|\theta) \frac{\nabla P(\tau|\theta)}{P(\tau|\theta)}
\]

R(\tau) do not have to be differentiable
It can even be a black box.

\[
= \sum_{\tau} R(\tau)P(\tau|\theta) \nabla \log P(\tau|\theta)
\]

\[
\approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla \log P(\tau^n|\theta)
\]

Use \(\pi_\theta\) to play the game \(N\) times,
Obtain \(\{\tau^1, \tau^2, \ldots, \tau^N\}\)
Policy Gradient

\[ \nabla \log P(\tau|\theta) = ? \]

- \[ \tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_T, a_T, r_T\} \]

\[ P(\tau|\theta) = p(s_1) \prod_{t=1}^{T} p(a_t|s_t, \theta)p(r_t, s_{t+1}|s_t, a_t) \]

\[ \log P(\tau|\theta) \]

\[ = \log p(s_1) + \sum_{t=1}^{T} \log p(a_t|s_t, \theta) + \log p(r_t, s_{t+1}|s_t, a_t) \]

\[ \nabla \log P(\tau|\theta) = \sum_{t=1}^{T} \nabla \log p(a_t|s_t, \theta) \]

Ignore the terms not related to \( \theta \)
Policy Gradient

\[ \theta^{new} \leftarrow \theta^{old} + \eta \nabla \overline{R}_{\theta^{old}} \]

\[ \nabla \overline{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla \log p(\tau^n | \theta) = \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \sum_{t=1}^{T_n} \nabla \log p(a_t^n | s_t^n, \theta) \]

What if we replace \( R(\tau^n) \) with \( r_t^n \) ....

If in \( \tau^n \) machine takes \( a_t^n \) when seeing \( s_t^n \) in

\( R(\tau^n) \) is positive \( \rightarrow \) Tuning \( \theta \) to increase \( p(a_t^n | s_t^n) \)

\( R(\tau^n) \) is negative \( \rightarrow \) Tuning \( \theta \) to decrease \( p(a_t^n | s_t^n) \)

It is very important to consider the cumulative reward \( R(\tau^n) \) of the whole trajectory \( \tau^n \) instead of immediate reward \( r_t^n \)
Policy Gradient

Given actor parameter $\theta$

$\tau^1$: $(s^1_1, a^1_1)$ $R(\tau^1)$
$(s^1_2, a^1_2)$ $R(\tau^1)$
$\vdots$ $\vdots$
$\tau^2$: $(s^2_1, a^2_1)$ $R(\tau^2)$
$(s^2_2, a^2_2)$ $R(\tau^2)$
$\vdots$ $\vdots$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta =$$

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p(a^n_t | s^n_t, \theta)$$

Update Model

Data Collection
Policy Gradient

Considered as Classification Problem

Minimize: \(- \sum_{i=1}^{3} \hat{y}_i \log y_i\)

Maximize: \(\log y_i = \log P("left"|s)\)

\(\theta \leftarrow \theta + \eta \nabla \log P("left"|s)\)
**Policy Gradient**

Given actor parameter $\theta$

$\tau^1: (s_1^1, a_1^1) \quad R(\tau^1) \quad (s_2^1, a_2^1) \quad R(\tau^1) \quad \vdots \quad \vdots$

$\tau^2: (s_1^2, a_1^2) \quad R(\tau^2) \quad (s_2^2, a_2^2) \quad R(\tau^2) \quad \vdots \quad \vdots$

\[
\theta \leftarrow \theta + \eta \nabla \tilde{R}_\theta \\
\nabla \tilde{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \nabla \log p(a_t^n | s_t^n, \theta)
\]
Policy Gradient

Given actor parameter $\theta$

$$\tau^1: \quad (s^1_1, a^1_1) \quad R(\tau^1) \quad (s^1_2, a^1_2) \quad R(\tau^1) \quad \cdots$$

$$\tau^2: \quad (s^2_1, a^2_1) \quad R(\tau^2) \quad (s^2_2, a^2_2) \quad R(\tau^2) \quad \cdots$$

Each training data is weighted by $R(\tau^n)$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta =$$

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p(a^*_t | s^*_t, \theta)$$

| $s^1_1$ | $NN$ | $a^1_1 = left$ |
| $s^1_2$ | $NN$ | $a^1_1 = left$ |
| $s^2_1$ | $NN$ | $a^2_1 = fire$ |
| $s^2_2$ | $NN$ | $a^2_1 = fire$ |
Add a Baseline

\[ \theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}} \]

\[ \nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \nabla \log p(a_t^n | s_t^n, \theta) \]

It is possible that \( R(\tau^n) \) is always positive.

Ideal case

Sampling ......

The probability of the actions not sampled will decrease.
Value-based Approach

Learning a Critic
Critic

• A critic does not determine the action.
• Given an actor $\pi$, it evaluates how good the actor is.

An actor can be found from a critic.

  e.g. Q-learning

Critic

- State value function $V^\pi(s)$
  - When using actor $\pi$, the *cumulated* reward expects to be obtained after seeing observation (state) $s$

$s \rightarrow V^\pi \rightarrow V^\pi(s) \rightarrow \text{scalar}$

$V^\pi(s)$ is large $\quad V^\pi(s)$ is smaller
**Critic**

\[
V\text{ 以前的阿光 (大馬步飛)} = \text{bad}
\]

\[
V\text{ 變強的阿光 (大馬步飛)} = \text{good}
\]
How to estimate $V^\pi(s)$

- Monte-Carlo based approach
  - The critic watches $\pi$ playing the game

After seeing $s_a$,
Until the end of the episode, the cumulated reward is $G_a$

After seeing $s_b$,
Until the end of the episode, the cumulated reward is $G_b$
How to estimate $V^\pi(s)$

- Temporal-difference approach

\[ \cdots s_t, a_t, r_t, s_{t+1} \cdots \]
\[ V^\pi(s_t) - V^\pi(s_{t+1}) \]

Some applications have very long episodes, so that delaying all learning until an episode's end is too slow.
MC v.s. TD

\[ s_a \rightarrow V^\pi \rightarrow V^\pi(s_a) \leftrightarrow G_a \]

Larger variance
unbiased

\[ s_t \rightarrow V^\pi \rightarrow V^\pi(s_t) \rightarrow r + V^\pi(s_{t+1}) \rightarrow V^\pi \rightarrow s_{t+1} \]

Smaller variance
May be biased
MC v.s. TD

• The critic has the following 8 episodes
  • $s_a, r = 0, s_b, r = 0, \text{END}$
  • $s_b, r = 1, \text{END}$
  • $s_b, r = 1, \text{END}$
  • $s_b, r = 1, \text{END}$
  • $s_b, r = 1, \text{END}$
  • $s_b, r = 1, \text{END}$
  • $s_b, r = 1, \text{END}$
  • $s_b, r = 0, \text{END}$

Monte-Carlo:

$V^\pi(s_a) = 0$

Temporal-difference:

$$V^\pi(s_b) + r = V^\pi(s_a)$$

$\begin{array}{ccc}
\frac{3}{4} & 0 & \frac{3}{4} \\
\end{array}$

(The actions are ignored here.)
Another Critic

- State-action value function $Q^\pi(s, a)$
- When using actor $\pi$, the \textit{cumulated} reward expects to be obtained after seeing observation $s$ and taking $a$

\[ Q^\pi(s, a) \]

\[ Q^\pi(s, a = left) \]
\[ Q^\pi(s, a = right) \]
\[ Q^\pi(s, a = fire) \]for discrete action only
Q-Learning

\[ \pi' \text{ "better" than } \pi \]

\[ \pi = \pi' \]

Learning \( Q^\pi(s, a) \)

\( \pi \) interacts with the environment

TD or MC

?
Q-Learning

Given $Q^\pi(s, a)$, find a new actor $\pi'$ “better” than $\pi$

“Better”: $V^{\pi'}(s) \geq V^\pi(s)$, for all state $s$

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

- $\pi'$ does not have extra parameters. It depends on $Q$
- Not suitable for continuous action $a$

$\pi$ interacts with the environment

TD or MC

Find a new actor $\pi'$ “better” than $\pi$

Learning $Q^\pi(s, a)$

 Estimate $Q$ instead of $V$
Deep Reinforcement Learning

Actor-Critic
Actor-Critic

\[ \pi \text{ interacts with the environment} \]

\[ \pi = \pi' \]

Update actor from \[ \pi \rightarrow \pi' \text{ based on } Q^\pi(s, a), V^\pi(s) \]

Learning \[ Q^\pi(s, a), V^\pi(s) \]

TD or MC
Actor-Critic

• Tips
  • The parameters of actor $\pi(s)$ and critic $V^\pi(s)$ can be shared
Asynchronous

1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models

\[ \theta^1 + \eta \Delta \theta \]

(other workers also update models)
Demo of A3C

• Racing Car (DeepMind)

https://www.youtube.com/watch?v=0xo1Ldx3L5Q
Demo of A3C

• Visual Doom AI Competition @ CIG 2016
• https://www.youtube.com/watch?v=94EPSjQH38Y
Concluding Remarks

Model-free Approach

- Policy-based
- Value-based

- Learning an Actor
- Actor + Critic
- Learning a Critic

Model-based Approach