Introduction of Deep Reinforcement Learning (RL)

Hung-yi Lee
Supervised Learning → RL

It is challenging to label data in some tasks.

...... machine can know the results are good or not.
Outline

What is RL? (Three steps in ML)
- Policy Gradient
- Actor-Critic
- Reward Shaping
- No Reward: Learning from Demonstration
Machine Learning
≈ Looking for a Function

Function input
Find a policy maximizing total reward

Action = \( f( \text{Observation} ) \)

Action output

Environment

Actor

Reward
Example: Playing Video Game

• Space invader

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

Observation

Actor

Environment

Action

“right”

Reward

reward = 0

Example: Playing Video Game

Observation

Actor

Environment

Action

“right”

Reward

reward = 0
Example: Playing Video Game

Find an actor maximizing expected reward.

Observation

Actor

Action

“fire”

Reward

reward = 5 if killing an alien.

Environment
Example: Learning to play Go

Observation

Action

Reward

Next Move

Environment
Example: Learning to play Go

Find an actor maximizing expected reward.

Observation → Action

- If win, reward = 1
- If loss, reward = -1
- reward = 0 in most cases
Machine Learning is so simple ......
Step 1: Function with Unknown

- 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

Policy Network (Actor)

Classification Task!!

<table>
<thead>
<tr>
<th>Action</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>0.7</td>
</tr>
<tr>
<td>right</td>
<td>0.2</td>
</tr>
<tr>
<td>fire</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Sample based on scores

Scores of actions
Step 2: Define “Loss”

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

Action $a_1$: “right”

Obtain reward $r_1 = 0$

Obtain reward $r_2 = 5$

Action $a_2$: “fire” (kill an alien)
Step 2: Define “Loss”

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

After many turns

Game Over (spaceship destroyed)

Obtain reward $r_T$

Action $a_T$

Total reward (return): $R = \sum_{t=1}^{T} r_t$

What we want to maximize

This is an episode.
Step 3: Optimization

Trajectory
\[ \tau = \{s_1, a_1, s_2, a_2, \ldots \} \]

They are black box ... ... with randomness

How to do the optimization here is the main challenge in RL.

c.f. GAN

Mathematical notation:

\[ R(\tau) = \sum_{t=1}^{T} r_t \]
Outline

What is RL? (Three steps in ML)

Policy Gradient

Actor-Critic

Reward Shaping

No Reward: Learning from Demonstration

To learn more about policy gradient:
https://youtu.be/W8XF3ME8G2I
How to control your actor

• Make it take (or don’t take) a specific action $\hat{a}$ given specific observation $s$.

Take action $\hat{a}$

$$L = e$$

Don’t take action $\hat{a}$

$$L = -e$$

Cross-entropy

$$\theta^* = arg\min_\theta L$$
How to control your actor

Take action $\hat{a}$ given $s$

Don’t take action $\hat{a}'$ given $s'$

$L = e_1 - e_2 \quad \theta^* = \arg\min_{\theta} L$
How to control your actor

Training Data

<table>
<thead>
<tr>
<th>Data</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>{s_1, \hat{a}_1}</td>
<td>+1  Yes</td>
</tr>
<tr>
<td>{s_2, \hat{a}_2}</td>
<td>-1  No</td>
</tr>
<tr>
<td>{s_3, \hat{a}_3}</td>
<td>+1  Yes</td>
</tr>
<tr>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>{s_N, \hat{a}_N}</td>
<td>-1  No</td>
</tr>
</tbody>
</table>

\[
L = + e_1 - e_2 + e_3 \cdots - e_N
\]

\[
\theta^* = \arg \min_\theta L
\]
How to control your actor

Training Data

\[
\begin{align*}
{s_1, \hat{a}_1} & \quad A_1 +1.5 \\
{s_2, \hat{a}_2} & \quad A_2 -0.5 \\
{s_3, \hat{a}_3} & \quad A_3 +0.5 \\
\vdots & \quad \vdots \\
{s_N, \hat{a}_N} & \quad A_N -10
\end{align*}
\]

\[L = \sum A_n e_n\]

\[\theta^* = \arg\min_{\theta} L\]
Version 0

Training Data

\[
\{s_1, a_1\}, \{s_2, a_2\}, \ldots, \{s_N, a_N\}, \quad A_1 = r_1, A_2 = r_2, \ldots, A_N = r_N
\]

many episodes

Short-sighted Version!
• An action affects the subsequent observations and thus subsequent rewards.

• *Reward delay*: Actor has to sacrifice immediate reward to gain more long-term reward.

• In space invader, only “fire” yields positive reward, so vision 0 will learn an actor that always “fire”.

Version 1

Training Data

\[
\begin{align*}
\{s_1, a_1\} & \quad A_1 = G_1 \\
\{s_2, a_2\} & \quad A_2 = G_2 \\
\{s_3, a_3\} & \quad A_3 = G_3 \\
\vdots & \quad \vdots \\
\{s_N, a_N\} & \quad A_N = G_N
\end{align*}
\]

\[G_1 = r_1 + r_2 + r_3 + \ldots + r_N\]
\[G_2 = r_2 + r_3 + \ldots + r_N\]
\[G_3 = r_3 + \ldots + r_N\]
\[G_t = \sum_{n=t}^{N} r_n\]

cumulated reward
Version 2

Training Data

\[
\begin{align*}
\{s_1, a_1\} & \quad A_1 = G'_1 \\
\{s_2, a_2\} & \quad A_2 = G'_2 \\
\{s_3, a_3\} & \quad A_3 = G'_3 \\
\vdots & \quad \vdots \\
\{s_N, a_N\} & \quad A_N = G'_N
\end{align*}
\]

\[
G_1 = r_1 + r_2 + r_3 + \ldots + r_N
\]

\[
G'_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots
\]

Discount factor \( \gamma < 1 \)
Version 3

Training Data

\{s_1, a_1\} \quad A_1 = G_1' - b
\{s_2, a_2\} \quad A_2 = G_2' - b
\{s_3, a_3\} \quad A_3 = G_3' - b
\vdots \quad \vdots
\{s_N, a_N\} \quad A_N = G_N' - b

If all the \(r_n \geq 10\)

\(r_n = 10\) is negative ... 

Minus by a baseline \(b\)

Make \(G_t'\) have positive and negative values

\[ G_t' = \sum_{n=t}^{N} \gamma^{n-t} r_n \]
Policy Gradient

• Initialize actor network parameters $\theta^0$
• For training iteration $i = 1$ to $T$
  • Using actor $\theta^{i-1}$ to interact
  • Obtain data $\{s_1, a_1\}, \{s_2, a_2\}, \ldots, \{s_N, a_N\}$
  • Compute $A_1, A_2, \ldots, A_N$
  • Compute loss $L$
  • $\theta^i \leftarrow \theta^{i-1} - \eta \nabla L$

Data collection is in the “for loop” of training iterations.
Policy Gradient

Training Data

\{s_1, a_1\} \quad A_1
\{s_2, a_2\} \quad A_2
\{s_3, a_3\} \quad A_3
\vdots \quad \vdots
\{s_N, a_N\} \quad A_N

Actor
\theta \quad \rightarrow \quad a

L = \sum A_n e_n

\theta^i \leftarrow \theta^{i-1} - \eta \nabla L

only update once

Each time you update the model parameters, you need to collect the whole training set again.
Policy Gradient

- Initialize actor network parameters $\theta^0$
- For training iteration $i = 1$ to $T$
  - Using actor $\theta^{i-1}$ to interact
  - Obtain data $\{s_1, a_1\}, \{s_2, a_2\}, \ldots, \{s_N, a_N\}$
  - Compute $A_1, A_2, \ldots, A_N$
  - Compute loss $L$
  - $\theta^i \leftarrow \theta^{i-1} - \eta \nabla L$

Experience of $\theta^{i-1}$

May not be good for $\theta^i$

One man's meat is another man's poison.
棋魂第八集

※ 小馬步飛：跟將棋一樣，將棋子放在斜一格；大馬步飛則是放在斜好幾格。

啊！
我覺得小馬也不错

大馬？

阿光，這時候不要下小馬步飛，而是要大馬步飛。
棋魂第八集
Policy Gradient

• Initialize actor network parameters $\theta^0$
• For training iteration $i = 1$ to $T$
  • Using actor $\theta^{i-1}$ to interact
  • Obtain data $\{s_1, a_1\}, \{s_2, a_2\}, \ldots, \{s_N, a_N\}$
  • Compute $A_1, A_2, \ldots, A_N$
  • Compute loss $L$
  • $\theta^i \leftarrow \theta^{i-1} - \eta \nabla L$

Trajectory of $\theta^{i-1}$

May not observe by $\theta^i$
On-policy v.s. Off-policy

• The actor to train and the actor for interacting is the same. → On-policy

• Can the actor to train and the actor for interacting be different? → Off-policy

In this way, we do not have to collection data after each update.
Off-policy → Proximal Policy Optimization (PPO)

• The **actor to train** has to know its difference from the **actor to interact**.

video:
https://youtu.be/OAKAZhFmYoI

Not apply to everyone
https://disp.cc/b/115-bLHe
Collection Training Data: Exploration

The actor needs to have randomness during data collection.
A major reason why we sample actions. 😊

Suppose your actor always takes “left”.
We never know what would happen if taking “fire”.

Enlarge output entropy
Add noises onto parameters
DeepMind - PPO

https://youtu.be/gn4nRCC9TwQ

It might look goofy ...
OpenAI - PPO

https://blog.openai.com/openai-baselines-ppo/
Outline

What is RL? (Three steps in ML)

Policy Gradient

Actor-Critic

Reward Shaping

No Reward: Learning from Demonstration
Critic

$G'_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots$

- Critic: Given actor $\theta$, how good it is when observing $s$ (and taking action $a$)
- Value function $V^\theta(s)$: When using actor $\theta$, the discounted \textit{cumulated} reward expects to be obtained after seeing $s$

The output values of a critic depend on the actor evaluated.
How to estimate $V^\theta(s)$

- **Monte-Carlo (MC) based approach**
  The critic watches actor $\theta$ to interact with the environment.

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 (TD) approach**

  $\cdots S_t, a_t, r_t, S_{t+1} \cdots$

  \[
  V^\theta(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdots \\
  V^\theta(s_{t+1}) = r_{t+1} + \gamma r_{t+2} + \cdots \\
  V^\theta(s_t) = \gamma V^\theta(s_{t+1}) + r_t
  \]

  $V^\theta(s_t) - \gamma V^\theta(s_{t+1}) \not\rightarrow r_t$
MC v.s. TD

- The critic has observed the following 8 episodes
  - $s_a, r = 0, s_b, r = 0$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 0$, END

\[
V^\theta(s_b) = \frac{3}{4}
\]

\[
V^\theta(s_a) = ? \quad 0? \quad \frac{3}{4}?
\]

(Assume $\gamma = 1$, and the actions are ignored here.)

Monte-Carlo: \[V^\theta(s_a) = 0\]

Temporal-difference:

\[
V^\theta(s_a) = V^\theta(s_b) + r
\]

\[
\frac{3}{4} \quad \frac{3}{4} \quad 0
\]
Training Data

\[
\begin{align*}
\{s_1, a_1\} & \quad A_1 = G'_1 - b \\
\{s_2, a_2\} & \quad A_2 = G'_2 - b \\
\{s_3, a_3\} & \quad A_3 = G'_3 - b \\
& \quad \vdots \\
\{s_N, a_N\} & \quad A_N = G'_N - b
\end{align*}
\]
Training Data

\[
\begin{align*}
\{s_1, a_1\} & \quad A_1 = G_1' - V^\theta(s_1) \\
\{s_2, a_2\} & \quad A_2 = G_2' - V^\theta(s_2) \\
\{s_3, a_3\} & \quad A_3 = G_3' - V^\theta(s_3) \\
\vdots & \quad \vdots \\
\{s_N, a_N\} & \quad A_N = G_N' - V^\theta(s_N)
\end{align*}
\]
\[ \{s_t, a_t\} \quad A_t = G'_t - V^\theta(s_t) \]

\[ G = 100 \]
\[ G = 3 \]
\[ G = 1 \]
\[ G = 2 \]
\[ G = -10 \]

\( V^\theta(s_t) \)

(not necessary take \( a_t \))

(You sample the actions based on a distribution)

\( A_t > 0 \)
\( a_t \) is better than average.

\( A_t < 0 \)
\( a_t \) is worse than average.
Version 4

Advantage Actor-Critic

\[ r_t + V^\theta(s_{t+1}) - V^\theta(s_t) \]

\( \{s_t, a_t\} \)
\[ A_t = G_t' - V^\theta(s_t) \]

\( s_t \)
\( a_t \)

\( s_t \)
\( s_{t+1} \)

G = 100
G = 3
G = 1
G = 2
G = 101
G = 4
G = 3
G = 1
G = -5
G = -10

(not necessary take \( a_t \))
Tip of Actor-Critic

- The parameters of actor and critic can be shared.
Outlook: Deep Q Network (DQN)

Video:
https://youtu.be/o_g9JUMw1Oc
https://youtu.be/2-zGCx4iv_k

https://arxiv.org/abs/1710.02298
Outline

What is RL? (Three steps in ML)

Policy Gradient

Actor-Critic

Reward Shaping

No Reward: Learning from Demonstration
Sparse Reward

\[ A_t = r_t + V^\theta(s_{t+1}) - V^\theta(s_t) \]

Training Data
\[
\{s_1, a_1\} \quad A_1 \\
\{s_2, a_2\} \quad A_2 \\
\{s_3, a_3\} \quad A_3 \\
\vdots \quad \vdots \\
\{s_N, a_N\} \quad A_N
\]

If \( r_t = 0 \) in most cases

We don’t know actions are good or bad.

e.g., robot arm to bolt on the screws

The developers define extra rewards to guide agents.

\[ \rightarrow \text{reward shaping} \]
Reward Shaping

VizDoom  https://openreview.net/forum?id=Hk3mPK5gg&noteId=Hk3mPK5gg

Visual Doom AI Competition @ CIG 2016
https://www.youtube.com/watch?v=94EPSjQH38Y
# Reward Shaping

**VizDoom**  
https://openreview.net/forum?id=Hk3mPK5gg&noteId=Hk3mPK5gg

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>FlatMap</th>
<th>CIGTrack1</th>
</tr>
</thead>
<tbody>
<tr>
<td>living</td>
<td>Penalize agent who just lives</td>
<td>-0.008 / action</td>
<td></td>
</tr>
<tr>
<td>health_loss</td>
<td>Penalize health decrement</td>
<td>-0.05 / unit</td>
<td></td>
</tr>
<tr>
<td>ammo_loss</td>
<td>Penalize ammunition decrement</td>
<td>-0.04 / unit</td>
<td></td>
</tr>
<tr>
<td>health_pickup</td>
<td>Reward for medkit pickup</td>
<td>0.04 / unit</td>
<td></td>
</tr>
<tr>
<td>ammo_pickup</td>
<td>Reward for ammunition pickup</td>
<td>0.15 / unit</td>
<td></td>
</tr>
<tr>
<td>dist_penalty</td>
<td>Penalize the agent when it stays</td>
<td>-0.03 / action</td>
<td></td>
</tr>
<tr>
<td>dist_reward</td>
<td>Reward the agent when it moves</td>
<td>9e-5 / unit distance</td>
<td></td>
</tr>
</tbody>
</table>

https://bair.berkeley.edu/blog/2017/12/20/reverse-curriculum/
Reward Shaping - Curiosity

Obtaining extra reward when the agent sees something new (but meaningful).

Curiosity Driven Exploration by Self-Supervised Prediction

ICML 2017

Deepak Pathak, Pulkit Agrawal, Alexei Efros, Trevor Darrell
UC Berkeley

Source of video: https://pathak22.github.io/noreward-rl/
Outline

What is RL? (Three steps in ML)
Policy Gradient
Actor-Critic
Reward Shaping
No Reward: Learning from Demonstration
Motivation

• Even define reward can be challenging in some tasks.
• Hand-crafted rewards can lead to uncontrolled behavior.

Three Laws of Robotics:
1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

restraining individual human behavior and sacrificing some humans will ensure humanity's survival
Imitation Learning

Actor can interact with the environment, but reward function is not available

We have demonstration of the expert.

\{\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_K\}

Each \(\hat{t}\) is a trajectory of the export.

- Self driving: record human drivers
- Robot: grab the arm of robot
Isn’t it Supervised Learning?

• Self-driving cars as example

\[ \hat{t} = \{s_1, \hat{a}_1, s_2, \hat{a}_2, \ldots \} \]

Yes, also known as Behavior Cloning

Problem: The experts only sample limited observation.
More problem ...... 

The agent will copy every behavior, even irrelevant actions.

https://www.youtube.com/watch?v=j2FSB3bseek
Inverse Reinforcement Learning

Using the reward function to find the optimal actor.
Inverse Reinforcement Learning

• Principle: *The teacher is always the best.*

• Basic idea:
  • Initialize an actor
  • In each iteration
    • The actor interacts with the environments to obtain some trajectories.
    • Define a reward function, which makes the trajectories of the teacher better than the actor.
    • The actor learns to maximize the reward based on the new reward function.
  • Output the reward function and the actor learned from the reward function.
Framework of IRL

\[ \sum_{n=1}^{K} R(\hat{\tau}_n) > \sum_{n=1}^{K} R(\tau) \]

Expert $\hat{\pi}$

\( \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_K\} \)

Obtain Reward Function $R$

\( \{\tau_1, \tau_2, \ldots, \tau_K\} \)

Actor $\pi$

Actor = Generator

Reward function = Discriminator

Find an actor based on reward function $R$

By Reinforcement learning
**GAN**

High score for real, low score for generated

**IRL**

Find a G whose output obtains large score from D

Larger reward for $\hat{\tau}$, Lower reward for $\tau$

Find a Actor obtains large reward
Robot

- How to teach robots? https://www.youtube.com/watch?v=DEGbtjTOIB0
http://rll.berkeley.edu/gcl/
To Learn More ...

Visual Reinforcement Learning with Imagined Goals, NIPS 2018
https://arxiv.org/abs/1807.04742

Skew-Fit: State-Covering Self-Supervised Reinforcement Learning, ICML 2020
https://arxiv.org/abs/1903.03698

Reinforcement learning with Imagined Goals (RIG)
Concluding Remarks

What is RL? (Three steps in ML)
- Policy Gradient
- Actor-Critic
- Sparse Reward
- No Reward: Learning from Demonstration