Introduction of Generative Adversarial Network (GAN)

Hung-yi Lee

李宏毅

Hung-yi Lee
Generative Adversarial Network (GAN)

• How to pronounce “GAN”?

Google 小姐
What are some recent and potentially upcoming breakthroughs in unsupervised learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU
Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Huang Xiao

Adversarial training is the coolest thing since sliced bread.

I’ve listed a bunch of relevant papers in a previous answer.

Expect more impressive results with this technique in the coming years.

What’s missing at the moment is a good understanding of it so we can make it work reliably. It’s very finicky. Sort of like ConvNet were in the 1990s, when I had the reputation of being the only person who could make them work (which wasn’t true).

Yann LeCun’s comment

What are some recent and potentially upcoming breakthroughs in deep learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU
Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Nikhil Garg, I lead a team of Quora engineers working on ML/NLP problems

......

The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.

Many kinds of network structures:

- Fully connected feedforward network
- Convolutional neural network (CNN)
- Recurrent neural network (RNN)

Different networks can take different kinds of input/output.

Vector

Matrix

Vector Seq

(speech, video, sentence)

How to find the function?

Given the example inputs/outputs as training data: \((x_1, y_1), (x_2, y_2), \ldots, (x_{1000}, y_{1000})\)
Creation

Anime Face Generation

Drawing?

何之源的知乎: https://zhuanlan.zhihu.com/p/24767059
DCGAN: https://github.com/carpedm20/DCGAN-tensorflow
Basic Idea of GAN

It is a neural network (NN), or a function.

Each dimension of input vector represents some characteristics.

- Longer hair
- Blue hair
- Open mouth
Basic Idea of GAN

It is a neural network (NN), or a function.

Larger value means real, smaller value means fake.
Basic Idea of GAN

Butterflies are not brown

Butterflies do not have veins

Generator

veins

………

Discriminator
Basic Idea of GAN

NN Generator v1 → NN Generator v2 → NN Generator v3

Discriminator v1 → Discriminator v2 → Discriminator v3

Real images:

This is where the term “adversarial” comes from. You can explain the process in different ways......
Basic Idea of GAN

Generator (student)  Discriminator (teacher)

Generator v1

Generator v2

Generator v3

Discriminator v1

Discriminator v2

No eyes

No mouth
Questions

Q1: Why generator cannot learn by itself?

Q2: Why discriminator don’t generate object itself?

Q3: How discriminator and generator interact?
Anime Face Generation

100 updates
Anime Face Generation

1000 updates
Anime Face Generation

2000 updates
Anime Face Generation

5000 updates
Anime Face Generation

10,000 updates
Anime Face Generation

20,000 updates
Anime Face Generation

50,000 updates
感謝陳柏文同學提供實驗結果
Outline

Lecture 1: Introduction of GAN
Lecture 2: Variants of GAN
Lecture 3: Making Decision and Control
Lecture I: Introduction of GAN

To learn more theory:
https://www.youtube.com/watch?v=0CKeqXl5lY0&lc=z13zuxbglpvsgbgpo04cg1bxuoraejdpapo0k
https://www.youtube.com/watch?v=KSN4QYgAtao&lc=z13kz1nqvuqsipqfn23phthasre4evrdo
Lecture I

When can I use GAN?

Generation by GAN

Improving GAN
Structured Learning

Machine learning is to find a function $f$

$$f : X \rightarrow Y$$

**Regression**: output a scalar

**Classification**: output a “class” (one-hot vector)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Class 1  
Class 2  
Class 3

**Structured Learning/Prediction**: output a sequence, a matrix, a graph, a tree ......

Output is composed of components with dependency
Regression, Classification
Output Sequence \[ f : X \rightarrow Y \]

**Machine Translation**

\( X \) : “機器學習及其深層與結構化”
(sentence of language 1)

\( Y \) : “Machine learning and having it deep and structured”
(sentence of language 2)

**Speech Recognition**

\( X \) : (speech)

\( Y \) : 感謝大家來上課”
(transcription)

**Chat-bot**

\( X \) : “How are you?”
(what a user says)

\( Y \) : “I’m fine.”
(response of machine)
Output Matrix

\[ f : X \rightarrow Y \]

**Image to Image**

\[ X : \quad Y : \]


**Text to Image**

\[ X : \quad Y : \]


“this white and yellow flower have thin white petals and a round yellow stamen”
Decision Making and Control

GO Playing is the same.

Action: “right”

Action: “fire”

Action: “left”
Why Structured Learning Interesting?

• **One-shot/Zero-shot Learning:**
  • In classification, each class has some examples.
  • In structured learning,
    • If you consider each possible output as a “class” ......
    • Since the output space is huge, most “classes” do not have any training data.
    • Machine has to create new stuff during testing.
    • Need more intelligence
Why Structured Learning Interesting?

- Machine has to learn to *planning*
  - Machine can generate objects component-by-component, but it should have a big picture in its mind.
  - Because the output components have dependency, they should be considered globally.

**Image Generation**

1 ✅

**Sentence Generation**

這個婆娘不是人 ✗

九天玄女下凡塵 ✓
Structured Learning Approach

**Generator**
Learn to generate the object at the component level

**Discriminator**
Evaluating the whole object, and find the best one
Lecture I

When can I use GAN?

Generation by GAN

Improving GAN
Generation

We will control what to generate latter. → Conditional Generation

**Image Generation**

\[
\begin{bmatrix}
0.3 \\
-0.1 \\
\vdots \\
-0.7
\end{bmatrix}
\begin{bmatrix}
0.1 \\
-0.1 \\
\vdots \\
0.7
\end{bmatrix}
\begin{bmatrix}
-0.3 \\
0.1 \\
\vdots \\
0.9
\end{bmatrix}
\]

In a specific range

**Sentence Generation**

\[
\begin{bmatrix}
0.3 \\
-0.1 \\
\vdots \\
-0.7
\end{bmatrix}
\begin{bmatrix}
0.1 \\
-0.1 \\
\vdots \\
0.2
\end{bmatrix}
\begin{bmatrix}
-0.3 \\
0.1 \\
\vdots \\
0.5
\end{bmatrix}
\]

How are you? Good morning. Good afternoon.
So many questions ......

Q1: Why generator cannot learn by itself?

Q2: Why discriminator don’t generate object itself?

Q3: How discriminator and generator interact?
Generator

code: $\begin{bmatrix} 0.1 \\ -0.5 \end{bmatrix}$

Image:

NN Generator

code vectors

$\begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$ $\begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix}$ $\begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix}$

As close as possible

NN Classifier

c.f.

$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix}$ $\begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix}$

(where does them come from?)
Generator

Encoder in auto-encoder provides the code 😊

Image:

- Code: $\begin{bmatrix} 0.1 \\ -0.5 \end{bmatrix}$
- Code: $\begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$
- Code: $\begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix}$
- Code: $\begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix}$
Auto-encoder

28 x 28 = 784

Compact representation of the input object

Learn together

Can reconstruct the original object

Trainable
Auto-encoder - Example

Pixel -> tSNE

PCA降到32-dim

Unsupervised Learning without annotation
Auto-encoder

As close as possible

Randomly generate a vector as code

= Generator

= Generator

= Generator

Image ?
Auto-encoder
Auto-encoder
Auto-encoder

code: 
\[
\begin{bmatrix}
0.1 \\
-0.5
\end{bmatrix}
\]

Image:
\[
\begin{bmatrix}
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.3 \\
0.2
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.1 \\
0.9
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.2 \\
-0.1
end{bmatrix}
**Auto-encoder**

- Input \( \rightarrow \) NN Encoder \( \rightarrow \) NN Decoder \( \rightarrow \) Output

**Variational Auto-encoder (VAE)**

- Input \( \rightarrow \) NN Encoder \( \rightarrow \) Code \( = \) Generator
- Minimize reconstruction error

From a normal distribution

\[
\begin{align*}
    \mathbf{e}_1 &= \exp(\sigma_1) \times e_1 + m_1 \\
    \mathbf{c}_1 &= \mathbf{e}_1 \\
    \mathbf{c}_2 &= \mathbf{e}_2 \\
    \mathbf{c}_3 &= \mathbf{e}_3 \\
    \sum_{i=1}^{3} (\exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)
\end{align*}
\]
Why VAE?

code

decode

?
Ref: http://www.wired.co.uk/article/google-artificial-intelligence-poetry
What do we miss?

It will be fine if the generator can truly copy the target image.
What if the generator makes some mistakes ........

Some mistakes are serious, while some are fine.
What do we miss?

1 pixel error
我覺得不行

6 pixel errors
我覺得其實 OK

1 pixel error
我覺得不行

6 pixel errors
我覺得其實 OK
What do we miss?

The relation between the components are critical.

The last layer generates each components independently.

Need deep structure to catch the relation between components.

Each neural in output layer corresponds to a pixel.
(Variational) Auto-encoder

\[
\begin{bmatrix}
    z_1 \\
    z_2
\end{bmatrix}
\xrightarrow{G}
\begin{bmatrix}
    x_1 \\
    x_2
\end{bmatrix}
\]
So many questions ……

Q1: Why generator cannot learn by itself?

Q2: Why discriminator don’t generate object itself?

Q3: How discriminator and generator interact?
Discriminator

- Discriminator is a function $D$ (network, can be deep)

\[ D : X \rightarrow \mathbb{R} \]

- Input $x$: an object $x$ (e.g. an image)
- Output $D(x)$: scalar which represents how “good” an object $x$ is

Can we use the discriminator to generate objects?

Yes.
Discriminator

- It is easier to catch the relation between the components by top-down evaluation.

This CNN filter is good enough.
Discriminator

• Suppose we already have a good discriminator \( D(x) \) ...

\[
\tilde{x} = \arg \max_{x \in X} D(x)
\]

Enumerate all possible \( x \) !!!

It is feasible ???

How to learn the discriminator?
Discriminator - Training

• I have some real images

Discriminator only learns to output “1” (real).

Discriminator training needs some negative examples.
In practice, you cannot decrease all the $x$ other than real examples.
Discriminator - Training

- Negative examples are critical.

How to generate realistic negative examples?
Discriminator - Training

- **General Algorithm**
  - Given a set of **positive examples**, randomly generate a set of **negative examples**.
  - In each iteration
    - Learn a discriminator $D$ that can discriminate positive and negative examples.
    - Generate negative examples by discriminator $D$

\[
\tilde{x} = \arg \max_{x \in X} D(x)
\]
Discriminator
- Training

In the end ......
Structured Learning ➢ Structured Perceptron ➢ Structured SVM ➢ Gibbs sampling ➢ Hidden information ➢ Application: sequence labelling, summarization

Graphical Model

Bayesian Network (Directed Graph)

Markov Random Field (Undirected Graph)

Conditional Random Field

Markov Logic Network

Segmental CRF

Boltzmann Machine

Restricted Boltzmann Machine

Generator v.s. Discriminator

• **Generator**
  • Pros:
    • Easy to generate even with deep model
  • Cons:
    • Imitate the appearance
    • Hard to learn the correlation between components

• **Discriminator**
  • Pros:
    • Considering the big picture
  • Cons:
    • Generation is not always feasible
      • Especially when your model is deep
    • How to do negative sampling?
So many questions ……

Q1: Why generator cannot learn by itself?

Q2: Why discriminator don’t generate object itself?

Q3: How discriminator and generator interact?
Discriminator - Training

**General Algorithm**

- Given a set of **positive examples**, randomly generate a set of **negative examples**.
- In each iteration
  - Learn a discriminator $D$ that can discriminate positive and negative examples.
  - Generate negative examples by discriminator $D$

$$
\tilde{x} = \arg \max_{x \in X} D(x)
$$
Generating Negative Examples

\[ \tilde{x} = \arg \max_{x \in X} D(x) \]

\[ \tilde{x} = \arg \max_{x \in X} \]

\[ G \rightarrow \tilde{x} \]

- NN Generator
- update
- hidden layer
- Discriminator fix
- Gradient Ascent

- code
- image

0.13
0.9
**Algorithm**

- Initialize generator and discriminator
- In each training iteration:

  - Sample some real objects:
  - Generate some fake objects:

  Update
  fix

  update
Algorithm

- Initialize generator and discriminator
- In each training iteration:

Sample some real objects:
Generate some fake objects:

Update
Benefit of GAN

• From Discriminator’s point of view
  • Using generator to generate negative samples
    \[ \tilde{x} = \arg \max_{x \in X} D(x) \]
    efficient

• From Generator’s point of view
  • Still generate the object component-by-component
  • But it is learned from the discriminator with global view.
GAN

Samples and Decision Boundary
G: 2*20; D: 4*10; prior dim: 2

Iter: 99500; D loss: -0.04111; G loss: 20.36
KLD(r,g)=[ 0. 0.]; KLD(g,r)=[ 0.6510948 0.72137838]


感謝 段逸林 同學提供結果
Lecture I

- When can I use GAN?
- Generation by GAN
- Improving GAN
GAN

- Discriminator leads the generator.
Binary Classifier as Discriminator

Typical binary classifier uses sigmoid function at the output layer.

1 is the largest, 0 is the smallest.

They don’t move.

You cannot train your classifier too good......
Binary Classifier as Discriminator

- Don’t let the discriminator perfectly separate real and generated data
  - Weaken your discriminator?
Binary Classifier as Discriminator

- Don’t let the discriminator perfectly separate real and generated data
  - Add noise to input or label?
Least Square GAN (LSGAN)

- Replace sigmoid with linear (replace classification with regression)

They don’t move.
• We want the scores of the real examples as large as possible, generated examples as small as possible.

Move towards Max

+∞

Any problem?
WGAN

The discriminator should be a 1-Lipschitz function.

It should be smooth.

How to realize?

**Lipschitz Function**

\[ \|D(x_1) - D(x_2)\| \leq K \|x_1 - x_2\| \]

Output change

Input change

K=1 for "1 – Lipschitz"

Do not change fast
WGAN

• Original WGAN $\rightarrow$ Weight Clipping
  Force the parameters $w$ between $c$ and $-c$
  After parameter update, if $w > c$, $w = c$; if $w < -c$, $w = -c$
  Do not truly maximize (minimize) the real (generated) examples

• Improved WGAN $\rightarrow$ Gradient Penalty

\[ D(x) \quad \text{Keep the gradient close to 1} \quad D(x) \]

Move towards Max

It should be smooth enough.

Easy to move
DCGAN: CNN, D: CNN

LSGAN: CNN (no normalization), D: CNN (no normalization)

Original WGAN: CNN (tanh), D: CNN(tanh)

Improved WGAN: CNN (tanh), D: CNN(tanh)
DCGAN
G: MLP, D: CNN

LSGAN
G: CNN (bad structure), D: CNN

Original WGAN

Improved WGAN
G: 101 layer, D: 101 layer
Sentence Generation

- good bye.

Consider this matrix as an “image”

I will talk about RNN later.
Sentence Generation

• Real sentence

• Generated

I will talk about RNN later.

A binary classifier can immediately find the difference.

No overlap

WGAN is helpful
Improved WGAN successfully generating sentences
W-GAN – 唐詩鍊成

感謝 李仲翊 同學提供實驗結果
輸出 32 個字 (包含標點)

• 升雲白遲丹齋取，此酒新巷市入頭。黃道故海歸中後，不驚入得韻子門。
• 據口容章藩翎翎，邦貸無遊隔將毬。外蕭曾臺遙出畧，此計推上呂天夢。
• 新來寶伎泉，手雪泓臺蓑。曾子花路魏，不謀散薦船。
• 功持牧度機邈爭，不躚官嬉牧涼散。不迎白旅今掩冬，盡蘸金祇可停。
• 玉十洪沄爭春風，溪子風佛挺橫鞋。盤盤稅焰先花齋，誰過飄鶴一丞憧。
• 海人依野庇，為阻例沉辮。座花不佐樹，弟闌十名儂。
• 入維當興日世瀕，不評進。頭醉空其杯，騨園凋送頭。
• 釧笙動春枝，寶叅潔長知。官為室爛去，絆粒薛一靜。
• 吾涼腕不楚，縱先待旅知。楚人縱酒待，一蔓飄聖猜。
• 折幕故瘝應韻子，徑頭霜瓊老徑徑。尚錯春銅熊悽梅，去吹依能九將香。
• 通可矯目鷓須浄，丹迤挈花一抵嫖。外子當目中前醒，迎日幽筆鈎弧前。
• 庭愛四樹人庭好，無衣服仍繡秋州。更怯風流欲鳬雲，帛陽舊據 오히婷儷。
Loss-sensitive GAN (LSGAN)

**WGAN**

- \( D(x) \)
- \( x \)
- \( x'' \)

**LSGAN**

- \( D(x) \)
- \( x \)
- \( x' \)
- \( x'' \)
- \( \Delta(x, x') \)
- \( \Delta(x, x') \)
- \( \Delta(x, x') \)
Energy-based GAN (EBGAN)

- Using an autoencoder as discriminator $D$

An image is good. $\Rightarrow$ It can be reconstructed by autoencoder.

Generator is the same.
EBGAN

Auto-encoder based discriminator only give limited region large value.

0 is for the best.

Do not have to be very negative

Hard to reconstruct, easy to destroy
Mode Collapse
Missing Mode?

Mode collapse is easy to detect.
Missing Mode?

- E.g. BEGAN on CelebA

陳柏文同學提供實驗結果
Ensemble

Generator 1

Generator 2

圖片來源: 柯達方 同學
Lecture II:
Variants of GAN
Lecture II

Conditional Generation

Sequence Generation

A Little Bit of Theory (option)
Conditional Generation

**Generation**

\[
\begin{bmatrix}
0.3 \\
-0.1 \\
\vdots \\
-0.7
\end{bmatrix}
\begin{bmatrix}
0.1 \\
-0.1 \\
\vdots \\
0.7
\end{bmatrix}
\begin{bmatrix}
-0.3 \\
0.1 \\
\vdots \\
0.9
\end{bmatrix}
\]

In a specific range

**Conditional Generation**

“Girl with red hair and red eyes”

“Girl with yellow ribbon”
Conditional Generation

- We don’t want to simply generate some random stuff.
- Generate objects based on conditions:

**Caption Generation**

**Given condition:**

“A young girl is dancing.”

**Chat-bot**

**Given condition:**

“Hello”

“Hello. Nice to see you.”
Conditional Generation

Modifying input code
- Making code has influence (InfoGAN)
- Connection code space with attribute

Controlling by input objects
- Paired data
- Unpaired data
- Unsupervised

Feature extraction
- Domain Independent Feature
- Improving Auto-encoder (VAE-GAN, BiGAN)
Modifying Input Code

The input code determines the generator output.

Understand the meaning of each dimension to control the output.
Conditional Generation

Modifying input code
- Making code has influence (InfoGAN)
- Connection code space with attribute

Controlling by input objects
- Paired data
- Unpaired data
- Unsupervised

Feature extraction
- Domain Independent Feature
- Improving Auto-encoder (VAE-GAN, BiGAN)
InfoGAN

(What we expect)

(The colors represent the characteristics.)

What we expect

Actually ...

Modifying a specific dimension, no clear meaning
What is InfoGAN?

\[ z = z' \]

\[ c \]

Generator

\[ x \]

Discriminator

encoder

decoder

"Auto-encoder"

Predict the code \( c \) that generates \( x \)

Parameter sharing (only the last layer is different)
What is InfoGAN?

$\mathbf{z} = \mathbf{c} + \mathbf{z}'$

**Generator**

Encoder

Decoder

**Classifier**

Predict the code $\mathbf{c}$ that generates $\mathbf{x}$

$c$ must have clear influence on $\mathbf{x}$

The classifier can recover $\mathbf{c}$ from $\mathbf{x}$. 
(a) Varying $c_1$ on InfoGAN (Digit type)  
(b) Varying $c_1$ on regular GAN (No clear meaning)  
(c) Varying $c_2$ from $-2$ to $2$ on InfoGAN (Rotation)  
(d) Varying $c_3$ from $-2$ to $2$ on InfoGAN (Width)
Conditional Generation

Modifying input code

- Making code has influence (InfoGAN)
- Connection code space with attribute

Controlling by input objects

- Paired data
- Unpaired data
- Unsupervised

Feature extraction

- Domain Independent Feature
- Improving Auto-encoder (VAE-GAN, BiGAN)
Connecting Code and Attribute

(c) Hair style
(d) Emotion

<table>
<thead>
<tr>
<th>Image</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td>Arched eyebrows, attractive, brown hair, heavy makeup, high cheekbones, mouth slightly open, no beard, pointy nose, smiling, straight hair, wearing earrings, wearing lipstick, young.</td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
<td>5 o’clock shadows, attractive, bags under eyes, big lips, big nose, black hair, bushy eyebrows, male, no beard, pointy nose, straight hair, young.</td>
</tr>
</tbody>
</table>
GAN+Autoencoder

• We have a generator (input $z$, output $x$)
• However, given $x$, how can we find $z$?
  • Learn an encoder (input $x$, output $z$)

![Diagram of GAN+Autoencoder]

Different structures? init fixed

As close as possible

Autoencoder
Attribute Representation

\[ z_{\text{long}} = \frac{1}{N_1} \sum_{x \in \text{long}} \text{En}(x) - \frac{1}{N_2} \sum_{x' \notin \text{long}} \text{En}(x') \]

\[ x \rightarrow \text{En}(x) + z_{\text{long}} = z' \rightarrow \text{Gen}(z') \]

Short Hair

Long Hair
Photo Editing

https://www.youtube.com/watch?v=kPEIJJsQr7U
Conditional Generation

Modifying input code

• Making code has influence (InfoGAN)
• Connection code space with attribute

Controlling by input objects

• Paired data
• Unpaired data
• Unsupervised

Feature extraction

• Domain Independent Feature
• Improving Auto-encoder (VAE-GAN, BiGAN)
Conditional Generation

Modifying input code
- Making code has influence (InfoGAN)
- Connection code space with attribute

Controlling by input objects
- Paired data
- Unpaired data
- Unsupervised

Feature extraction
- Domain Independent Feature
- Improving Auto-encoder (VAE-GAN, BiGAN)
Conditional GAN

- Generating images based on text description

- Diagram:
  - Sentence: "a dog is sleeping"
  - Code input to NN Encoder
  - Code output from NN Encoder
  - Code input to NN Generator
  - Image output from NN Generator
Conditional GAN

- Generating images based on text description

\[ c^1: \text{a dog is running} \quad \hat{x}^1: \]
\[ c^2: \text{a bird is flying} \quad \hat{x}^2: \]

- Sentence: "a dog is running"
  "a bird is flying"

- Code

- NN Encoder

- Code

- NN Generator

- Image: Dog running
  Image: Bird flying
Conditional GAN

- **Text to image** by traditional supervised learning

- $c^1$: a dog is running  \( \hat{x}^1: \) 
- $c^2$: a bird is flying  \( \hat{x}^2: \)

Text: “train”

A blurry image!
Conditional GAN

Prior distribution $z$ \quad \rightarrow \quad G \quad \rightarrow \quad Image \quad x = G(c,z)$

Text: “train”

A blurry image!

Target of NN output

It is a distribution.
Approximate the distribution below
Conditional GAN

Prior distribution $z$ $\Rightarrow$ $G \Rightarrow$ Image $\quad x = G(c, z)$

$x$ is realistic or not $\Rightarrow$ $\text{D (type 1)} \Rightarrow$ scalar

$x$ is realistic or not + $c$ and $x$ are matched or not $\Rightarrow$ $\text{WIN}$ $\Rightarrow$ scalar

Positive example: (train, )
Negative example: (train, Image) (cat, )
### Text to Image - Results

<table>
<thead>
<tr>
<th>Caption</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>this flower has white petals and a yellow stamen</td>
<td>![Images of white flowers with yellow centers]</td>
</tr>
<tr>
<td>the center is yellow surrounded by wavy dark purple petals</td>
<td>![Images of purple flowers with yellow centers]</td>
</tr>
<tr>
<td>this flower has lots of small round pink petals</td>
<td>![Images of pink flowers]</td>
</tr>
</tbody>
</table>
## Text to Image - Results

<table>
<thead>
<tr>
<th>Caption</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>a pitcher is about to throw the ball to the batter</td>
<td><img src="image1.png" alt="Images" /></td>
</tr>
<tr>
<td>a group of people on skis stand in the snow</td>
<td><img src="image2.png" alt="Images" /></td>
</tr>
<tr>
<td>a man in a wet suit riding a surfboard on a wave</td>
<td><img src="image3.png" alt="Images" /></td>
</tr>
</tbody>
</table>
Conditional GAN

- Black hair, blue eyes
- Blue hair, green eyes
- Red hair, long hair
Data Collection

http://konachan.net/post/show/239400/aikatsu-clouds-flowers-hikami_sumire-hiten_goane_r

感謝曾柏翔助教、樊恩宇助教蒐集資料
Released Training Data

• Data download link: https://drive.google.com/open?id=0BwJmB7alR-AvMHEtczZZN0EtdzQ

• Anime Dataset:
  • training data: 33.4k (image, tags) pair

• Training tags file format
  • img_id <comma> tag1 <colon> #_post <tab> tag2 <colon> …

```
1,q,touhou:17705,ch:423,moneti daifuku:60,animal ears:12241,catgirl:49033
2,1,touhou:17697,onozuka komachi:224,shikieiki yamaxanadu:217
3,2,original:25774,blonde hair:25457,doll:1040,dress:16585,pink eyes:3896
4,3,amagi brilliant park:111,musaigen no phantom world:39,nichijou:142,kawakami
```
tags.csv
Image-to-image

\[ x = G(c,z) \]
Image-to-image

- Traditional supervised approach

Testing:

It is blurry because it is the average of several images.
Image-to-image

- Experimental results

Testing:

- input
- close
- GAN
- GAN + close
Image super resolution


![Image showing bicubic, SRResNet, SRGAN, and original images with PSNR and SSIM values.](image)

Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]
Video Generation

Generator

Discriminator

Last frame is real or generated

Minimize distance

Discriminator thinks it is real

target
https://github.com/dyelax/Adversarial_Video_Generation
Speech Enhancement

• Typical deep learning approach

Noisy  Clean

Enhancing

Using CNN

Output
Speech Enhancement

• Conditional GAN

- noisy
- output
- clean

training data

noisy

clean

scalar
(fake pair or not)
Speech Enhancement

Noisy Speech

Enhanced Speech

Which Enhanced Speech is better?

感謝廖峴峰同學提供實驗結果 (和中研院曹昱博士共同指導)
More about Speech Processing

• Speech synthesis
  • Yuki Saito, Shinnosuke Takamichi, and Hiroshi Saruwatari, "Training algorithm to deceive anti-spoofing verification for DNN-based speech synthesis, ”, ICASSP 2017

• Voice Conversion
  • Chin-Cheng Hsu, Hsin-Te Hwang, Yi-Chiao Wu, Yu Tsao, Hsin-Min Wang, Voice Conversion from Unaligned Corpora using Variational Autoencoding Wasserstein Generative Adversarial Networks, Interspeech 2017

• Speech Enhancement
  • Santiago Pascual, Antonio Bonafonte, Joan Serrà, SEGAN: Speech Enhancement Generative Adversarial Network, Interspeech 2017
Conditional Generation

Modifying input code

• Making code has influence (InfoGAN)
• Connection code space with attribute

Controlling by input objects

• Paired data
• Unpaired data
• Unsupervised

Feature extraction

• Domain Independent Feature
• Improving Auto-encoder (VAE-GAN, BiGAN)
Cycle GAN, Disco GAN

Transform an object from one domain to another \textit{without paired data}
Cycle GAN

https://arxiv.org/abs/1703.10593
https://junyanz.github.io/CycleGAN/

Domain X

\[ G_{X \rightarrow Y} \]

ignore input

Become similar to domain Y

Not what we want

Input image belongs to domain Y or not

Domain Y

\[ D_Y \]

scalar
Cycle GAN

$G_{X \rightarrow Y}$

$G_{Y \rightarrow X}$

$D_Y$

as close as possible

Lack of information for reconstruction

Input image belongs to domain Y or not

scalar

Domain X

Domain Y
c.f. Dual Learning

Cycle GAN

as close as possible

scalar: belongs to domain X or not

scalar: belongs to domain Y or not

as close as possible
動畫化的世界

- Using the code: https://github.com/Hi-king/kawaii_creator
- It is not cycle GAN, Disco GAN
Conditional Generation

Modifying input code
- Making code has influence (InfoGAN)
- Connection code space with attribute

Controlling by input objects
- Paired data
- Unpaired data
  - Unsupervised

Feature extraction
- Domain Independent Feature
- Improving Auto-encoder (VAE-GAN, BiGAN)
Basic Idea

Why move on the code space?

Fulfill the constraint
Back to $z$

- **Method 1**
  
  $z^* = \arg\min_z L(G(z), x^T)$

  Gradient Descent

  ➢ Difference between $G(z)$ and $x^T$
    ➢ Pixel-wise
    ➢ By another network

- **Method 2**

  as close as possible

- **Method 3**

  Using the results from **method 2** as the initialization of **method 1**
Editing Photos

• $z_0$ is the code of the input image

$$z^* = \arg\min_z U(G(z)) + \lambda_1 \left\| z - z_0 \right\|^2 - \lambda_2 D(G(z))$$

Does it fulfill the constraint of editing?

Using discriminator to check the image is realistic or not

Not too far away from the original image
Conditional Generation

Modifying input code
- Making code has influence (InfoGAN)
- Connection code space with attribute

Controlling by input objects
- Paired data
- Unpaired data
- Unsupervised

Feature extraction
- Domain Independent Feature
- Improving Auto-encoder (VAE-GAN, BiGAN)
Domain Independent Features

• Training and testing data are in different domains
Domain Independent Features
Domain Independent Features

Too easy to feature extractor .......

Domain classifier
Domain-adversarial training

Maximize label classification accuracy + minimize domain classification accuracy

Not only cheat the domain classifier, but satisfying label classifier at the same time

Maximize domain classification accuracy

This is a big network, but different parts have different goals.
Domain-adversarial training

Domain classifier fails in the end
It should struggle ......

Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016
Domain-adversarial training

Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016
Conditional Generation

Modifying input code

• Making code has influence (InfoGAN)
• Connection code space with attribute

Controlling by input objects

• Paired data
• Unpaired data
• Unsupervised

Feature extraction

• Domain Independent Feature
• Improving Auto-encoder (VAE-GAN, BiGAN)
VAE-GAN

➢ Minimize reconstruction error
➢ z close to normal


Discriminator provides the similarity measure

Discriminator provides the similarity measure
BiGAN

Jeff Donahue, Philipp Krähenbühl, Trevor Darrell, “Adversarial Feature Learning”, ICLR, 2017
Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, Aaron Courville, “Adversarially Learned Inference”, ICLR, 2017

Encoder

Decoder

Discriminator

(from prior distribution)

code z

code z

from encoder or decoder?

Image x (real)

Image x (generated)

Image x

code z
Lecture II

Conditional Generation

Sequence Generation

A Little Bit of Theory (option)
Sentence Generation

• *Training*

Reference:

```
A
B

A
B

A
B

A
B

<BOS>
```
Sentence Generation

• *Generation (Testing)*
Sentence Generation

Generator → sentence x

sentence x → Discriminator → Real or fake

Original GAN
Sentence Generation

- Initialize generator $G$ and discriminator $D$
- In each iteration:
  - Sample real sentences $x$ from database
  - Generate sentences $\tilde{x}$ by $G$
  - Update $D$ to increase $D(x)$ and decrease $D(\tilde{x})$
- Update Gen such that
Can we do backpropagation? **NO!**

Tuning generator a little bit will not change the output.

In the paper of improved WGAN ... *(ignoring sampling process)*
Sentence Generation - SeqGAN

- Using Reinforcement learning
  - Consider the discriminator as reward function
  - Consider the output of discriminator as total reward
  - Update generator to increase discriminator = to get maximum total reward

Conditional Generation

- Represent the input condition as a vector, and consider the vector as the input of RNN generator.
- E.g. Machine translation / Chat-bot

Diagram:
- Encoder and Decoder jointly train information of whole sentences and generate responses like "I'm fine."
Chat-bot with GAN

Input sentence/history h

Chatbot

(response sentence x)

Input sentence/history h

Discriminator

Real or fake

(response sentence x)

human dialogues


Conditional GAN
Example Results

input | I love you.

input | Do you like machine learning?

input | I thought I have met you before.

input | Let’s go to the party.

input | How do you feel about the president?
**Cycle GAN**

Negative sentence to positive sentence:
- it's a crappy day → it's a great day
- i wish you could be here → you could be here
- it's not a good idea → it's good idea
- i miss you → i love you
- i don't love you → i love you
- i can't do that → i can do that
- i feel so sad → i happy
- it's a bad day → it's a good day
- it's a crappy day → it's a great day
- sorry for doing such a horrible thing → thanks for doing a great thing
- my doggy is sick → my doggy is my doggy
- i am so hungry → i am so
- my little doggy is sick → my little doggy is my little doggy
Lecture II

- Conditional Generation
- Sequence Generation
- A Little Bit of Theory (option)
Theory behind GAN

• The data we want to generate has a distribution $P_{data}(x)$
Theory behind GAN

• A generator G is a network. The network defines a probability distribution.

Normal Distribution $\mathbb{Z}$

$\mathbb{Z} \rightarrow \text{generator } G$

$x = G(\mathbb{Z})$

$P_G(x)$

$P_{\text{data}}(x)$

As close as possible

It is difficult to compute $P_G(x)$
We do not know what the distribution looks like.

https://blog.openai.com/generative-models/
When the discriminator is well trained, its **loss** represents a specific divergence measure between $P_G$ and $P_{\text{data}}$.

Update discriminator is to minimize the divergence measure.

Binary classifier evaluates the JS divergence.

You can design the discriminator to evaluate other divergence.

https://blog.openai.com/generative-models/
Why GAN is hard to train?

Better
Why GAN is hard to train?

\[
J_S(P_{G_1} \mid \mid P_{data}) = \log 2
\]

Not really better......
Evaluating JS divergence

- JS divergence estimated by discriminator telling little information

https://arxiv.org/abs/1701.07875
Earth Mover’s Distance

- Considering one distribution $P$ as a pile of earth, and another distribution $Q$ as the target.
- The average distance the earth mover has to move the earth.

$W(P, Q) = d$
Why Earth Mover’s Distance?

\[ D_f(P_{data} \parallel P_G) \]

\[ W(P_{data}, P_G) \]

\[ d_0 \]

\[ P_{G_0} \quad P_{data} \quad \ldots \quad P_{G_{50}} \quad P_{data} \quad \ldots \quad P_{G_{100}} \quad P_{data} \]

\[ JS(P_{G_0}, P_{data}) = \log 2 \]

\[ JS(P_{G_{50}}, P_{data}) = \log 2 \]

\[ JS(P_{G_{100}}, P_{data}) = 0 \]

\[ W(P_{G_0}, P_{data}) = d_0 \]

\[ W(P_{G_{50}}, P_{data}) = d_{50} \]

\[ W(P_{G_{100}}, P_{data}) = 0 \]
\[ W(P_{\text{data}}, P_G) = \max_{D \in 1-Lipschitz} \left\{ E_{x \sim P_{\text{data}}} [D(x)] - E_{x \sim P_G} [D(x)] \right\} \]
To Learn more ......

- GAN Zoo
  - https://github.com/hindupuravinash/the-gan-zoo
- Tricks: https://github.com/soumith/ganhacks
Lecture III:
Decision Making and Control
Decision Making and Control

• Machine observe some inputs, takes an action, and finally achieve the target.

• E.g. Go playing

State: summarization of observation

Target: win the game

observation

action
Decision Making and Control

• Machine plays video games
• 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 are pixels
➢ Machine learns to take proper action itself
Decision Making and Control

- E.g. self-driving car

observation

Object Recognition

Action: Stop!

Decision Making

a giant network?
Decision Making and Control

- E.g. dialogue system

我想訂11月5日到台北的機票

抵達時間：11月5日目的地：台北

請問您要從哪裡出發呢？
How to solve this problem?

• Network as a function, learn as typical supervised tasks

Target: 3-3

Target: Stop!

Target: 請問您要從哪裡出發呢？

我想訂11月5日到台北的機票
Behavior Cloning

Machine do not know some behavior must copy, but some can be ignored.

https://www.youtube.com/watch?v=j2FSB3bseek
Properties of Decision Making and Control

What do we miss?

- Agent’s actions affect the subsequent data it receives
- Reward delay
  - In space invader, only “fire” obtains reward
  - Although the moving before “fire” is important
  - In Go playing, sacrificing immediate reward to gain more long-term reward

Machine does not know the influence of each action.
Better Way ......

A sequence of decision

Network

Next move

Classification
Two Learning Scenarios

• **Scenario 1: Reinforcement Learning**
  • Machine interacts with the environment.
  • Machine obtains the reward from the environment, so it knows its performance is good or bad.

• **Scenario 2: Learning by demonstration**
  • Also known as imitation learning, apprenticeship learning
  • An expert demonstrates how to solve the task, and machine learns from the demonstration.
Lecture III

Reinforcement Learning

Inverse Reinforcement Learning
RL

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

Model-free Approach

Policy-based

Value-based

Learning an Actor
Actor + Critic
Learning a Critic

Model-based Approach
Basic Components

<table>
<thead>
<tr>
<th>Actor</th>
<th>Env</th>
<th>Reward Function</th>
</tr>
</thead>
</table>

- **Video Game**
  - You cannot control
  - 殺一隻怪得 20 分 ...

- **Go**
  - 遊棋規則
  - AlphaGo, Google DeepMind
Scenario

Observation

Function input

Used to pick the best function

Actor/Policy

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

Reward

Environment

Action

Function output
Learning to play Go

Observation

Action

Reward

Next Move

Environment
Learning to play Go

Agent learns to take actions maximizing expected reward.

Observation

Action

Reward

If win, reward = 1

If loss, reward = -1

reward = 0 in most cases
Actor, Environment, Reward

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

Total reward:
\[ R(\tau) = \sum_{t=1}^{T} r_t \]
Example: Playing Video Game

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

Example: Playing Video Game

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 $s_1$

Observation $s_2$

Observation $s_3$

After many turns

Game Over (spaceship destroyed)

Obtain reward $r_T$

Action $a_T$

This is an *episode*.

Total reward:

$$R = \sum_{t=1}^{T} r_t$$

We want the total reward be maximized.
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?

Take the action based on the probability.

Score of an action

Generalization
Actor, Environment, Reward

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

Policy Gradient
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 ......
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

- 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 expected value of $R_\theta$

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

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

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

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

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

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

Control by your actor \( \pi_\theta \)

Actor

\[ \pi_\theta \]

\[ s_t \rightarrow \text{left} 0.1 \]

\[ \text{right} 0.2 \]

\[ \text{fire} 0.7 \]

not related to your actor
Goodness of Actor

• An episode is considered as a trajectory $\tau$
  • $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$
  • $R(\tau) = \sum_{t=1}^{T} r_t$

• 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 $\{\tau^1, \tau^2, \cdots, \tau^N\}$

Sampling $\tau$ from $P(\tau|\theta)$ N times
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 ......
Gradient Ascent

• Problem statement
  \[ \theta^* = \arg \max_{\theta} R_\theta \]

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

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

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

\[ \theta^{\text{new}} \leftarrow \theta^{\text{old}} + \eta \nabla R_{\theta}^{\text{old}} \]

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

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 \( r_t^n \)
Given actor parameter $\theta$

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

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

$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p(a_t^n | s_t^n, \theta)$
Policy Gradient

Considered as Classification Problem

\[
\begin{align*}
\text{Minimize:} & \quad - \sum_{i=1}^{3} \hat{y}_i \log y_i \\
\text{Maximize:} & \quad \log y_i = \log P(\text{"left"} | s) \\
\theta & \leftarrow \theta + \eta \nabla \log P(\text{"left"} | s)
\end{align*}
\]
Policy Gradient

Given actor parameter $\theta$

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

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

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

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

$\begin{align*}
a^1_1 &= left \\
s^1_1 &\rightarrow NN \\
\quad &\rightarrow left \\
\quad &\rightarrow right \\
\quad &\rightarrow fire \\
\vdots \\
s^2_1 &\rightarrow NN \\
\quad &\rightarrow left \\
\quad &\rightarrow right \\
\quad &\rightarrow fire \\
\end{align*}$
Given actor parameter $\theta$

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

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

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

\[ \theta \leftarrow \theta + \eta \nabla \tilde{R}_\theta \]
\[ \nabla \tilde{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p(a^n_t | s^n_t, \theta) \]
End of Warning
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) $s$
以前的阿光 (大馬步飛) = bad
變強的阿光 (大馬步飛) = 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

\[ \cdots s_t, a_t, r_t, s_{t+1} \cdots \]

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

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

Larger variance unbiased

Smaller variance

May be biased
The critic has the following 8 episodes

- \( s_a, r = 0, s_b, r = 0, \text{ END} \)
- \( s_b, r = 1, \text{ END} \)
- \( s_b, r = 1, \text{ END} \)
- \( s_b, r = 1, \text{ END} \)
- \( s_b, r = 1, \text{ END} \)
- \( s_b, r = 0, \text{ END} \)

Monte-Carlo: \( V^\pi(s_a) = 0 \)

Temporal-difference:

\[
V^\pi(s_b) + r = V^\pi(s_a) \\
\frac{3}{4} + 0 = \frac{3}{4}
\]

(The actions are ignored here.)
Actor-Critic

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

\[ \pi = \pi' \]

Update actor from
\[ \pi \rightarrow \pi' \text{ based on Critic} \]

Learning Critic

Actor = Generator?

Critic = Discriminator?

https://arxiv.org/abs/1610.01945
Playing On-line Game

劉廷緯、溫明浩
https://www.youtube.com/watch?v=8iRD1w73fDo&feature=youtu.be
Lecture III

Reinforcement Learning

Inverse Reinforcement Learning
Imitation Learning

We have demonstration of the expert.

\[ s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{\ldots} \]

reward function is not available

Self driving: record human drivers
Robot: grab the arm of robot

\[ \{\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_N\} \]

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

• It is hard to define reward in some tasks.
  • Hand-crafted rewards can lead to uncontrolled behavior.
Inverse Reinforcement Learning

➢ Using the reward function to find the \textit{optimal} actor.

➢ Modeling reward can be easier. Simple reward function can lead to complex policy.
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

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

Expert \( \hat{\pi} \) → \( \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_N\} \) → Obtain Reward Function R

\( \{\tau_1, \tau_2, \ldots, \tau_N\} \) → Reward Function R

Actor \( \pi \) → Find an actor based on reward function R

Actor = Generator
Reward function = Discriminator

By Reinforcement learning
GAN

High score for real, low score for generated

D

Find a G whose output obtains large score from D

IRL

Find a G whose output obtains large score from D

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

Reward Function

Find a Actor obtains large reward

Expert

$\hat{\tau}_1, \hat{\tau}_2, \cdots, \hat{\tau}_N$

$\tau_1, \tau_2, \cdots, \tau_N$

Actor

$\tau_1, \tau_2, \cdots, \tau_N$
Teaching Robot

• In the past ......  https://www.youtube.com/watch?v=DEGbtjTOIB0
http://rll.berkeley.edu/gcl/
Path Planning

http://martin.zinkevich.org/publications/maximummarginplanning.pdf
Third Person Imitation Learning


**First Person**

http://lasa.epfl.ch/research_new/ML/index.php

**Third Person**

https://kknews.cc/sports/q5kbb8.html

http://sc.chinaz.com/Files/pic/icons/1913/%E6%9C%BA%E5%99%A8%E4%BA%BA%E5%9B%BE%E6%A0%87%E4%B8%8B%E8%BD%BD34.png
Third Person Imitation Learning
Concluding Remarks

Lecture 1: Introduction of GAN

Lecture 2: Variants of GAN

Lecture 3: Making Decision and Control
To Learn More ...

• Machine Learning
  • Slides: http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html
  • Video: https://www.youtube.com/watch?v=fegAeph9UaA&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49

• Machine Learning and Having it Deep and Structured
  • Slides: http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS17.html
  • Video: https://www.youtube.com/watch?v=IzHoNwlCGnE&list=PLJV_el3uVTsPMxPbjeX7PicgWbY7F8wW9