Generative Adversarial Network (GAN)

Outlook:
- Restricted Boltzmann Machine:
  http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RBM%20(v2).ecm.mp4/index.html
- Gibbs Sampling:
  http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/MRF%20(v2).ecm.mp4/index.html
NIPS 2016 Tutorial: Generative Adversarial Networks

• Author: Ian Goodfellow
• Paper: https://arxiv.org/abs/1701.00160
• Video: https://channel9.msdn.com/Events/Neural-Information-Processing-Systems-Conference/Neural-Information-Processing-Systems-Conference-NIPS-2016/Generative-Adversarial-Networks

You can find tips for training GAN here: https://github.com/soumith/gan hacks
Review
Generation

Writing Poems?

Drawing?
Review: Auto-encoder

As close as possible

Randomly generate a vector as code

Image?
Review: Auto-encoder

-1.5 → [1.5, 0] → 2D code

NN Decoder → [0, 0] → NN Decoder

1.5 → [-1.5, 0] → NN Decoder

Diagram showing the process of encoding and decoding in an auto-encoder.
Review: Auto-encoder
Auto-encoder

**Encoder**

Input → NN Encoder → Code → NN Decoder → Output

**VAE**

From a normal distribution

Input → NN Encoder → $m_1$ $m_2$ $m_3$ $\sigma_1$ $\sigma_2$ $\sigma_3$ $e_1$ $e_2$ $e_3$ → $c_1$ $c_2$ $c_3$ → NN Decoder → Output

Minimize reconstruction error

$\sum_{i=1}^{3} (exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)$

Auto-Encoding Variational Bayes, https://arxiv.org/abs/1312.6114
Problems of VAE

- It does not really try to simulate real images

![Diagram showing the process of VAE with code, NN Decoder, Output, and examples of realistic and fake outputs with one pixel difference from the target.]
The evolution of generation

NN Generator v1

Discriminator v1

Binary Classifier

Real images: 5 0 4 1

NN Generator v2

Discriminator v2

NN Generator v3

Discriminator v3
The evolution of generation

- NN Generator v1
- Discriminator v1
- Real images: 2399

- NN Generator v2
- Discriminator v2
- Real images: 0000

- NN Generator v3
- Discriminator v3
- Real images: 0000
GAN - Discriminator

Randomly sample a vector

Something like Decoder in VAE

NN Generator v1

Real images:

1/0 (real or fake)
GAN - Generator

Updating the parameters of generator

The output be classified as “real” (as close to 1 as possible)

Generator + Discriminator = a network

Using gradient descent to update the parameters in the generator, but fix the discriminator

Randomly sample a vector
GAN – 二次元人物頭像鍊成

Source of images: https://zhuanlan.zhihu.com/p/24767059
DCGAN: https://github.com/carpedm20/DCGAN-tensorflow
GAN – 二次元人物頭像鍊成

100 rounds
GAN – 二次元人物頭像鍊成

1000 rounds
GAN – 二次元人物頭像鍊成

2000 rounds
GAN – 二次元人物頭像鍊成

5000 rounds
GAN - 二次元人物頭像鍊成

10,000 rounds
GAN – 二次元人物頭像鍊成

20,000 rounds
GAN – 二次元人物頭像鍊成

50,000 rounds
Basic Idea of GAN
Maximum Likelihood Estimation

- Given a data distribution $P_{data}(x)$
- We have a distribution $P_G(x; \theta)$ parameterized by $\theta$
  - E.g. $P_G(x; \theta)$ is a Gaussian Mixture Model, $\theta$ are means and variances of the Gaussians
  - We want to find $\theta$ such that $P_G(x; \theta)$ close to $P_{data}(x)$

Sample $\{x^1, x^2, \ldots, x^m\}$ from $P_{data}(x)$

We can compute $P_G(x^i; \theta)$

Likelihood of generating the samples

$$L = \prod_{i=1}^{m} P_G(x^i; \theta)$$

Find $\theta^*$ maximizing the likelihood
Maximum Likelihood Estimation

\[ \theta^* = \arg \max_{\theta} \prod_{i=1}^{m} P_G(x^i; \theta) = \arg \max_{\theta} \log \prod_{i=1}^{m} P_G(x^i; \theta) \]

\[ = \arg \max_{\theta} \sum_{i=1}^{m} \log P_G(x^i; \theta) \]

\[ \approx \arg \max_{\theta} E_{x \sim P_{data}} [\log P_G(x; \theta)] \]

\[ = \arg \max_{\theta} \int_{x} P_{data}(x) \log P_G(x; \theta) dx - \int_{x} P_{data}(x) \log P_{data}(x) dx \]

\[ = \arg \min_{\theta} KL(P_{data}(x) || P_G(x; \theta)) \]

How to have a very general \( P_G(x; \theta) \)?
Now $P_G(x; \theta)$ is a NN

$$P_G(x; \theta) = \int P_{\text{prior}}(z) I_{[G(z)=x]} \, dz$$

It is difficult to compute the likelihood.

https://blog.openai.com/generative-models/
Basic Idea of GAN

- **Generator G**
  - G is a function, input z, output x
  - Given a prior distribution $P_{\text{prior}}(z)$, a probability distribution $P_{G}(x)$ is defined by function G

- **Discriminator D**
  - D is a function, input x, output scalar
  - Evaluate the “difference” between $P_{G}(x)$ and $P_{\text{data}}(x)$

- There is a function $V(G,D)$.

$$G^* = \underset{G}{\text{arg min}} \underset{D}{\text{max}} V(G, D)$$
Basic Idea

\[ G^* = \arg \min_G \max_D V(G, D) \]

\[
V = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log (1 - D(x))] 
\]

Given a generator \( G \), \( \max_D V(G, D) \) evaluate the “difference” between \( P_G \) and \( P_{data} \).

Pick the \( G \) defining \( P_G \) most similar to \( P_{data} \).

Diagram:

- \( V(G_1, D) \)
- \( V(G_2, D) \)
- \( V(G_3, D) \)
$$\max_D V(G, D) \quad G^* = \arg \min_G \max_D V(G, D)$$

• Given $G$, what is the optimal $D^*$ maximizing

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

$$= \int_{x} P_{data}(x) \log D(x) \, dx + \int_{x} P_{G}(x) \log(1 - D(x)) \, dx$$

$$= \int_{x} [P_{data}(x) \log D(x) + P_{G}(x) \log(1 - D(x))] \, dx$$

Assume that $D(x)$ can have any value here

• Given $x$, the optimal $D^*$ maximizing

$$P_{data}(x) \log D(x) + P_{G}(x) \log(1 - D(x))$$
\[
\max_D V(G, D) \quad G^* = \arg\min_G \max_D V(G, D)
\]

- Given \(x\), the optimal \(D^*\) maximizing

\[
P_{\text{data}}(x) \log D(x) + P_G(x) \log (1 - D(x))
\]

- Find \(D^*\) maximizing:

\[
f(D) = a \log(D) + b \log(1 - D)
\]

\[
\frac{df(D)}{dD} = a \times \frac{1}{D} + b \times \frac{1}{1 - D} \times (-1) = 0
\]

\[
a \times \frac{1}{D^*} = b \times \frac{1}{1 - D^*} \quad a \times (1 - D^*) = b \times D^* \quad a - aD^* = bD^*
\]

\[
D^* = \frac{a}{a + b} \quad D^*(x) = \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_G(x)} < 1
\]
\[
\max_D V(G, D) \quad G^* = \arg \min_G \max_D V(G, D)
\]

\[
D_1^*(x) = \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_{G_1}(x)}
\]

\[
D_2^*(x) = \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_{G_2}(x)}
\]

“difference” between \(P_{G_1}\) and \(P_{\text{data}}\)
\[
\begin{align*}
\max_D V(G, D) &= V(G, D^*) \\
&= E_{x \sim P_{data}} \left[ \log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} \right] \\
&= \int_{x} P_{data}(x) \log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} \, dx \\
&\quad + 2 \log \frac{1}{2} - 2 \log 2 \\
&\quad + \int_{x} P_G(x) \log \frac{P_G(x)}{P_{data}(x) + P_G(x)} \, dx
\end{align*}
\]

\[
V = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log (1 - D(x))]
\]

\[
D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}
\]
\[
\max_D V(G, D) = V(G, D^*) = -2\log 2 + \int x P_{\text{data}}(x) \log \frac{P_{\text{data}}(x)}{(P_{\text{data}}(x) + P_G(x))/2} dx \\
+ \int x P_G(x) \log \frac{P_G(x)}{(P_{\text{data}}(x) + P_G(x))/2} dx \\
= -2\log 2 + \text{KL}\left( P_{\text{data}}(x) || \frac{P_{\text{data}}(x) + P_G(x)}{2} \right) \\
+ \text{KL}\left( P_G(x) || \frac{P_{\text{data}}(x) + P_G(x)}{2} \right) \\
= -2\log 2 + 2\text{JSD}\left( P_{\text{data}}(x) || P_G(x) \right) \text{ Jensen-Shannon divergence}
\]

\[
\text{JSD}(P \parallel Q) = \frac{1}{2} D(P \parallel M) + \frac{1}{2} D(Q \parallel M)
\]

\[
M = \frac{1}{2} (P + Q)
\]
In the end ...

• Generator G, Discriminator D
• Looking for G* such that

\[ \begin{align*}
V &= E_{x \sim P_{\text{data}}} [\log D(x)] \\
&\quad + E_{x \sim P_G} [\log (1 - D(x))] \\
G^* &= \arg\min_G \max_D V(G, D)
\end{align*} \]

• Given G, \( \max_D V(G, D) \) = 
\[ -2 \log 2 + 2 JSD(P_{\text{data}}(x) || P_G(x)) \]

• What is the optimal G?

\[ P_G(x) = P_{\text{data}}(x) \]
Algorithm

\[ G^* = \arg \min_G \max_D \mathcal{V}(G, D) \]

- To find the best \( G \) minimizing the loss function \( L(G) \),

\[ \theta_G \leftarrow \theta_G - \eta \frac{\partial L(G)}{\partial \theta_G} \]

\[ f(x) = \max\{D_1(x), D_2(x), D_3(x)\} \]

\[ \frac{df(x)}{dx} = \text{?} \quad dD_i(x)/dx \]

If \( D_i(x) \) is the max one
Algorithm

• Given $G_0$
• Find $D_0^*$ maximizing $V(G_0, D)$

$V(G_0, D_0^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_0}(x)$

• $\theta_G \leftarrow \theta_G - \eta \frac{\partial V(G, D_0^*)}{\partial \theta_G}$ Obtain $G_1$ Decrease JS divergence(?)
• Find $D_1^*$ maximizing $V(G_1, D)$

$V(G_1, D_1^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_1}(x)$

• $\theta_G \leftarrow \theta_G - \eta \frac{\partial V(G, D_1^*)}{\partial \theta_G}$ Obtain $G_2$ Decrease JS divergence(?)
• ......
Algorithm

• Given $G_0$
• Find $D_0^*$ maximizing $V(G_0, D)$

$V(G_0, D_0^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_0}(x)$

• $\theta_G \leftarrow \theta_G - \eta \frac{\partial V(G, D_0^*)}{\partial \theta_G}$ Obtain $G_1$

Decrease JS divergence(?)

Assume $D_0^* \approx D_1^*$

Don’t update G too much
In practice...

• Given G, how to compute \( \max_D V(G, D) \)

• Sample \( \{x^1, x^2, ..., x^m\} \) from \( P_{data}(x) \), sample \( \{\tilde{x}^1, \tilde{x}^2, ..., \tilde{x}^m\} \) from generator \( P_G(x) \)

Maximize

\[
\tilde{V} = \frac{1}{m} \sum_{i=1}^{m} \log D(x^i) + \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(\tilde{x}^i)\right)
\]

Binary Classifier

Output is \( D(x) \)  

Minimize Cross-entropy

If \( x \) is a positive example  Minimize \(-\log D(x)\)

If \( x \) is a negative example  Minimize \(-\log(1-D(x))\)

\[
V = \mathbb{E}_{x \sim P_{data}} \left[ \log D(x) \right] + \mathbb{E}_{x \sim P_G} \left[ \log (1 - D(x)) \right]
\]
D is a binary classifier (can be deep) with parameters $\theta_d$

\[
\{x^1, x^2, \ldots, x^m\} \text{ from } P_{\text{data}}(x) \quad \rightarrow \quad \text{Positive examples}
\]

\[
\{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^m\} \text{ from } P_G(x) \quad \rightarrow \quad \text{Negative examples}
\]

Minimize

\[
L = \frac{1}{m} \sum_{i=1}^{m} \log D(x^i) + \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(\tilde{x}^i)\right)
\]

Maximize

\[
\tilde{V} = \frac{1}{m} \sum_{i=1}^{m} \log D(x^i) + \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(\tilde{x}^i)\right)
\]
**Algorithm**

Initialize $\theta_d$ for D and $\theta_g$ for G

- In each training iteration:
  - Sample $m$ examples $\{x^1, x^2, ..., x^m\}$ from data distribution $P_{data}(x)$
  - Sample $m$ noise samples $\{z^1, z^2, ..., z^m\}$ from the prior $P_{prior}(z)$
  - Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, ..., \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
  - Update discriminator parameters $\theta_d$ to maximize
    \[
    \tilde{V} = \frac{1}{m} \sum_{i=1}^{m} \log D(x^i) + \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(\tilde{x}^i)\right)
    \]
    \[
    \theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)
    \]
  - Sample another $m$ noise samples $\{z^1, z^2, ..., z^m\}$ from the prior $P_{prior}(z)$
  - Update generator parameters $\theta_g$ to minimize
    \[
    \tilde{V} = \frac{1}{m} \sum_{i=1}^{m} \log D(x^i) + \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(G(z^i))\right)
    \]
    \[
    \theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)
    \]

**Learning D**

Repeat $k$ times

**Learning G**

Only Once

**Can only find lower found of** $\max_D V(G, D)$
Objective Function for Generator in Real Implementation

\[ V = E_{x \sim P_{data}} [\log D(x)] \]
\[ + E_{x \sim P_G} [\log (1 - D(x))] \]

Slow at the beginning

\[ V = E_{x \sim P_G} [-\log (D(x))] \]

Real implementation: label x from \( P_G \) as positive
Demo

- The code used in demo from:
  - https://github.com/osh/KerasGAN/blob/master/MNIST_CNN_GAN_v2.ipynb
Issue about Evaluating the Divergence
Evaluating JS divergence

Martin Arjovsky, Léon Bottou, Towards Principled Methods for Training Generative Adversarial Networks, 2017, arXiv preprint
Evaluating JS divergence

- JS divergence estimated by discriminator telling little information

https://arxiv.org/abs/1701.07875

Weak Generator

Strong Generator
Discriminator

\[ V = E_{x \sim P_{\text{data}}}[\log D(x)] + E_{x \sim P_G}[\log (1 - D(x))] \]

\[ \approx \frac{1}{m} \sum_{i=1}^{m} \log D(x^i) + \frac{1}{m} \sum_{i=1}^{m} \log (1 - D(\tilde{x}^i)) \]

\[ \max_D V(G, D) = -2 \log 2 + 2 \text{JS}D(P_{\text{data}}(x) || P_G(x)) = 0 \]

Reason 1. Approximate by sampling

Weaken your discriminator?
Can weak discriminator compute JS divergence?
Discriminator

\[
V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log (1 - D(x))]
\]

\[
\approx \frac{1}{m} \sum_{i=1}^{m} \log D(x^i) + \frac{1}{m} \sum_{i=1}^{m} \log (1 - D(\tilde{x}^i))
\]

\[
\max_D V(G, D) = -2 \log 2 + 2 \text{JS}(P_{data}(x) \| P_G(x)) = 0
\]

Reason 2. the nature of data

Both \( P_{data}(x) \) and \( P_G(x) \) are low-dim manifold in high-dim space

Usually they do not have any overlap
Evaluation

http://www.guokr.com/post/773890/

Better
Evaluation

Better

\[ \text{Better} \]

\[ \text{Not really better} \]

\[ JS(P_{G_1} \| P_{data}) = \log 2 \]

\[ JS(P_{G_2} \| P_{data}) = \log 2 \]

\[ JS(P_{G_2} \| P_{data}) = 0 \]
Add Noise

• Add some artificial noise to the inputs of discriminator

• Make the labels noisy for the discriminator

Discriminator cannot perfectly separate real and generated data

\( P_{data}(x) \) and \( P_{G}(x) \) have some overlap

Noises decay over time
Mode Collapse
Mode Collapse

Generated Distribution

Data Distribution

[Image of generated distribution and data distribution with anime character images]
Mode Collapse

What we want ...

In reality ...

$P_{data}$
Flaw in Optimization?

\[ KL = \int P_{data} \log \frac{P_{data}}{P_G} \, dx \]

Reverse KL

\[ Reverse \, KL = \int P_G \log \frac{P_G}{P_{data}} \, dx \]

Maximum likelihood
(minimize \( KL(P_{data} \| P_G) \))

Minimize KL\((P_G \| P_{data})\)
(reverse KL)

This may not be the reason (based on Ian Goodfellow’s tutorial)
So many GANs ......

| Modifying the Optimization of GAN | Different Structure from the Original GAN |
|-----------------------------------|------------------------------------------|
| fGAN                              | Conditional GAN                          |
| WGAN                              | Semi-supervised GAN                      |
| Least-square GAN                  | InfoGAN                                  |
| Loss Sensitive GAN                | BiGAN                                    |
| Energy-based GAN                  | Cycle GAN                                |
| Boundary-seeking GAN              | Disco GAN                                |
| Unroll GAN                        | VAE-GAN                                  |
| ......                             | ......                                    |
Conditional GAN
Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, Honglak Lee, “Generative Adversarial Text-to-Image Synthesis”, ICML 2016

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, Dimitris Metaxas, “StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks”, arXiv prepring, 2016

Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, Honglak Lee, “Learning What and Where to Draw”, NIPS 2016
Motivation

• Challenge

Text $c$ → NN → Image $x$

(a point, not a distribution)

Text: “train”
Conditional GAN

Training data: \((\hat{c}, \hat{x})\)

- **Condition** \(c\)
- **Prior distribution** \(z\)

Learn to approximate \(P(x|c)\)

\((\hat{c}, x = G(\hat{c}))\) classified as positive

Learn to ignore this term ...

Dropout

- **D (v1)**
  - Input: \(x\)
  - Output: scalar
  - Can generated \(x\) not related to \(c\)

- **D (v2)**
  - Input: \(c\), \(x\)
  - Output: scalar
  - Positive example: \((\hat{c}, \hat{x})\)
  - Negative example: \((\hat{c}, G(\hat{c})), (\hat{c}', \hat{x})\)
## Text to Image - Results

| Caption | Image |
|---------|-------|
| a pitcher is about to throw the ball to the batter |
| a group of people on skis stand in the snow |
| a man in a wet suit riding a surfboard on a wave |
| Caption                                                      | Image                                      |
|--------------------------------------------------------------|--------------------------------------------|
| this flower has white petals and a yellow stamen             | ![Image of white flowers with yellow centers](image1) |
| the center is yellow surrounded by wavy dark purple petals   | ![Image of flowers with purple petals](image2) |
| this flower has lots of small round pink petals              | ![Image of pink flowers](image3)           |
Image-to-image Translation

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks”, arXiv preprint, 2016
Positive examples

Real or fake pair?

D

D

G tries to synthesize fake images that fool D

D tries to identify the fakes

Negative examples

Real or fake pair?

D

G

D

D

G
Image-to-image Translation
- Results
Speech Enhancement

GAN

https://arxiv.org/abs/1703.09452
Speech Enhancement GAN

Using Least-square GAN
Least-square GAN

• For discriminator

$$\min_D \frac{1}{2} E_{x \sim P_{data}} [(D(x) - b)^2] + \frac{1}{2} E_{x \sim P_G} [(D(x) - a)^2]$$

D has linear output

• For Generator

$$\min_D \frac{1}{2} E_{z \sim P_{data}} [(D(G(z)) - c)^2]$$
Least-square GAN

• The code used in demo from:
  • https://github.com/osh/KerasGAN/blob/master/MNIST_CNN_GAN_v2.ipynb