Machine Learning I (DATS 6202)
Generative Adversarial Network

Yuxiao Huang

Data Science, Columbian College of Arts & Sciences
George Washington University

Spring 2020

Reference

- This set of slices was largely built on the following 8 wonderful books and a wide range of fabulous papers:

HML	Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd Edition)
PML	Python Machine Learning (3rd Edition)
ESL	The Elements of Statistical Learning (2nd Edition)
PRML	Pattern Recognition and Machine Learning
LFD	Learning From Data
NND	Neural Network Design (2nd Edition)
NNDL	Neural Network and Deep Learning
RL	Reinforcement Learning: An Introduction

- For most materials covered in the slides, we will specify their corresponding books and papers for further reference.

Code Example & Case Study

- See code example of topics discussed in this section in github repository: /p2_c3_s2_generative_adversarial_network/code_example
- See case study of Kaggle Competition related to this section in github repository: /p2_c3_s2__generative_adversarial_network/case_study

Overview

Learning Objectives

- It is **expected** to understand
  - the architecture and idea of GAN
  - the good practices for designing DCGAN
- It is **recommended** to understand
  - the practical way of training of GAN

The Idea

- Generative Adversarial Network (GAN) [1] has received great attention for its good performance in generating artificial data resembling the real one.
- There are two components in GAN:
    - a *Generator* that tries to generate data similar to the real one
    - a *Discriminator* that tries to discriminate between the real data and the generated one

The Idea

- The idea of training GAN is based on an adversarial game between the two components:
  - the generator gets better (by deceiving the discriminator) only when the discriminator gets better (by debunking the generator)
  - the discriminator gets better (by debunking the generator) only when the generator gets better (by deceiving the discriminator)
- In theory, as training advances GAN eventually reaches a *Nash Equilibrium* where:
  - the generator generates data that perfectly mimic the real one
  - the discriminator discriminates the two kinds of data with accuracy 0.5

An Example of GAN

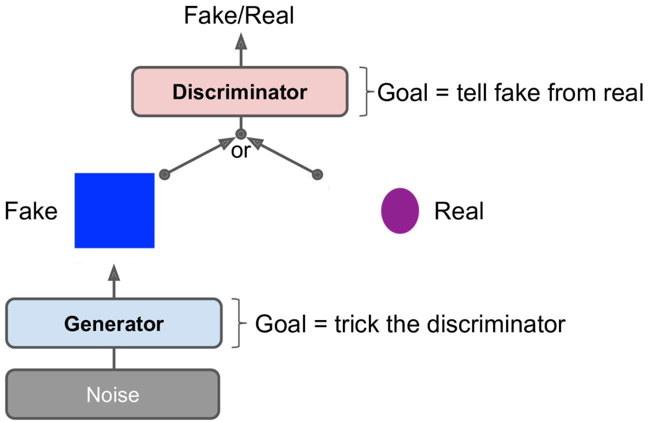

Figure 1: An Example of GAN. Picture revised from the one in *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd Edition)*.

An Example of GAN

- Fig. 1 shows an example illustrating how GAN works.
- In this example, we use an adversarial game between a *Forger* and an *Investigator* to mimic the game between the generator and discriminator in GAN:
  - a forger (generator) attempts to make fake pearls and sell them at a high price
  - however, before selling them, the pearls have to be run by an investigator (discriminator), who decides whether the pearls are real
- Both the forger and investigator are in the early stage of their career. That is, neither of them is good at what they are doing:
  - the forger makes big square-shaped pearls
  - the investigator has no idea how to distinguish real pearls from fake ones
- The good news is, both of them are eager to learn.

An Example of GAN

1. The investigator learns by comparing the fake pearls with the real ones. The first thing caught the investigator's eye is the size: real ones are small but fake ones are big.
2. This insight was later leaked to the forger. He then starts making small square-shaped pearls to deceive the investigator.
3. This time, by comparing the fake pearls with the real ones, the investigator finds that he can also use the shape to separate the two kinds of pears: real ones are oval but fakes ones are square.
4. This insight was again leaded to the forger. He then starts making small oval-shaped pears to deceive the investigator.
5. By repeating this process, the investigator learns the differences between the real and fake pearls and the generator uses such information to narrow the differences.
6. Eventually, the fake pearls are so perfect such that the discriminator cannot tell any difference anymore.

The Objective Function

- The objective function of GAN, $V(\theta^{(D)}, \theta^{(G)})$, is

$$E_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x})\right] + E_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\log\Big(1 - D\big(G(\mathbf{z})\big)\Big)\right], \quad \text{where} \qquad (1)$$

  - D is the discriminator
  - G is the generator
  - $\mathbf{x}$ is the real data
  - $\mathbf{z}$ is the latent vector and $G(\mathbf{z})$ the fake (generated) data
  - $D(\mathbf{x})$ is the probability of $\mathbf{x}$ being real or fake
  - $D\big(G(\mathbf{z})\big)$ is the probability of $G(\mathbf{z})$ being real or fake
  - $E_{\mathbf{x}\sim p_{data}(\mathbf{x})}$ is the expectation with respect to the distribution of $\mathbf{x}$
  - $E_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}$ is the expectation with respect to the distribution of $\mathbf{z}$
- In other words, the objective function is a sum of two expectations:
  - the first expectation, $E_{\mathbf{x}\sim p_{data}(\mathbf{x})}$, is the average of the probability of the discriminator correctly predicting real data ($\mathbf{x}$ in Eq. (1)) as real
  - the second expectation, $E_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}$, is the average of the probability of the discriminator correctly predicting fake data ($G(\mathbf{z})$ in Eq. (1)) as fake
- It is worth noting that while the discriminator is related to both expectations (to predict the data as real or fake), the generator is only related to the second expectation (to generate the fake data).

The Optimization

- The optimization of GAN includes the following two steps (in order):
  1. maximize the objective function, $V(\theta^{(D)}, \theta^{(G)})$, with respect to the discriminator, D
  2. minimize the objective function, $V(\theta^{(D)}, \theta^{(G)})$, with respect to the generator, G
- The two steps above can be written as

$$\min_{G} \max_{D} V(\theta^{(D)}, \theta^{(G)}). \tag{2}$$

- At a glance, it is a bit strange that we optimize the objective using two completely opposite ways:
  - maximize it with respect to the discriminator
  - minimize it with respect to the generator
- Here is why the optimization makes sense.

Maximization regarding the Discriminator

- The objective function of GAN, $V(\theta^{(D)}, \theta^{(G)})$, is

$$E_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + E_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \left[\log \Big(1 - D\big(G(\mathbf{z})\big)\Big)\right].$$

- In the first expectation of the objective, $D(\mathbf{x})$ is the probability of the discriminator predicting the real data, $\mathbf{x}$, as real.
- The discriminator wants this probability as close to 1 (the maximum value of a probability) as possible, where 1 means the discriminator is certain that the real data is real.
- Since $\log D(\mathbf{x})$ is strictly increasing (i.e., the higher $D(\mathbf{x})$ the higher $\log D(\mathbf{x})$), making $D(\mathbf{x})$ as large as possible equates making $\log D(\mathbf{x})$ as large as possible, hence maximizing the first expectation.

Maximization regarding the Discriminator

- The objective function of GAN, $V(\theta^{(D)}, \theta^{(G)})$, is

$$E_{\mathbf{x} \sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x})\right] + E_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}\left[\log\Big(1 - D\big(G(\mathbf{z})\big)\Big)\right].$$

- In the second expectation of the objective, $D\big(G(\mathbf{z})\big)$ is the probability of the discriminator predicting the fake data, $G(\mathbf{z})$, as real.
- The discriminator wants this probability as close to 0 (the minimum value of a probability) as possible, where 0 means the discriminator is certain that the fake data is not real.
- Since $\log\Big(1 - D\big(G(\mathbf{z})\Big)$ is strictly increasing (i.e., the higher $1 - D\big(G(\mathbf{z})$ the higher $\log\Big(1 - D\big(G(\mathbf{z})\big)\Big)$), making $D\big(G(\mathbf{z})\big)$ as small as possible equates making $1 - D\big(G(\mathbf{z})$ as large as possible, which in turn, equates making $\log\Big(1 - D\big(G(\mathbf{z})\Big)$ as large as possible, hence maximizing the second expectation.

Maximization regarding the Discriminator

- To sum up, since we want to maximize both the first and the second expectation in the objective function, we want to maximize the whole objective function with respect to the discriminator.
- This is why we have the maximization in Eq. (2)

$$\min_{G} \max_{D} V(\theta^{(D)}, \theta^{(G)}).$$

Minimization regarding the Generator

- The objective function of GAN, $V(\theta^{(D)}, \theta^{(G)})$, is

$$E_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + E_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \left[\log \Big(1 - D\big(G(\mathbf{z})\big)\Big)\right].$$

- Since only the second expectation in the objective is related to the generator, we only need to discuss why we should minimize the second expectation with respect to the generator.
- As mentioned earlier, in the second expectation $D\big(G(\mathbf{z})\big)$ is the probability of the discriminator predicting the fake data, $G(\mathbf{z})$, as real.
- The generator wants this probability as close to 1 as possible, where 1 means that discriminator is certain that the fake data is real.
- Since $\log \Big(1 - D\big(G(\mathbf{z})\big)\Big)$ is strictly increasing, making $D\big(G(\mathbf{z})\big)$ as large as possible equates making $1 - D\big(G(\mathbf{z})\big)$ as small as possible, hence minimizing the second expectation.
- This is why we have the minimization in eq. (2)

$$\min_G \max_D V(\theta^{(D)}, \theta^{(G)}).$$

Training GAN

- A practical way of training GAN is to alternate between the two optimization steps (in order):
  1. freeze the parameters of the generator and update the parameters of the discriminator
  2. freeze the parameters of the discriminator and update the parameters of the generator

Training GAN

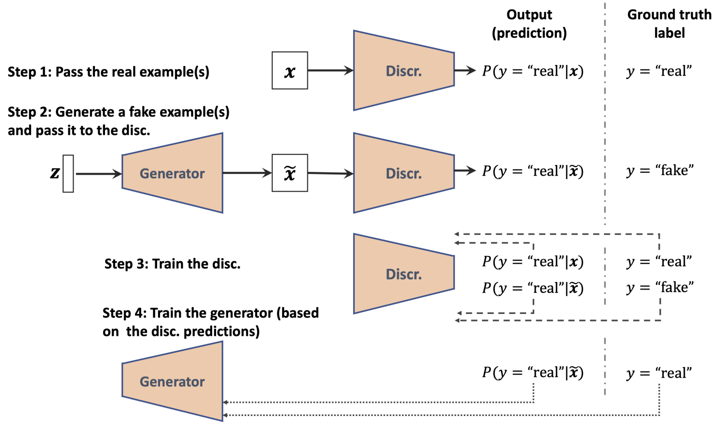

Figure 2: One iteration of training GAN. Picture courtesy of *Python Machine Learning (3rd Edition)*.

Deep Convolutional GAN

- GAN based on deep CNNs have been proposed to generate large images.
- One popular model is Deep Convolutional GAN (DCGAN) [2].
- **Good Practice**: The main guidelines for building stable DCGAN:
  - replace any pooling layer with strided convolutions (in the discriminator) and transposed convolutions (in the generator)
  - use *Batch Normalization* in both the generator and the discriminator, except in the generator's output layer and discriminator's input layer
  - remove fully connected hidden layers
  - use `ReLU` activation in the generator for all layers except the output layer, which should use tanh
  - use `leaky ReLU` activation in the discriminator for all layers

Further Reading

- See a detailed discussion of GAN in a NeurIPS 2016 tutorial: https://arxiv.org/pdf/1701.00160.pdf

References

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.
Generative Adversarial Nets.
In *NeurIPS*, pages 2672–2680, 2014.

A. Radford, L. Metz, and S. Chintala.
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.
arXiv preprint arXiv:1511.06434, 2015.