Generative Adversarial Nets

Yunhao Li, Siheng Huang, Yicong Li

Department of Statistics,
Rutgers University

May, 2023

1. generator
2. discriminator
3. adversary
4. Theoretical Results

Recall - classical machine learning

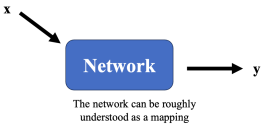


Figure 1: classical machine learning mechanism

As we learned before with the classical machine learning model, after a set of data x is input, a mod will output some values y.

In a neural network, x can be a set of data, a set of pictures, or even a set of voice messages, y can be a category or a set of sequence

New Topic - Use the network as a generator

Special - A random variable z will be added

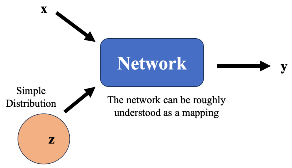


Figure 2: Special - A random variable z will be added

z comes from another simple distribution. At this point, network's input is not just a set of x, but x and z.

New Topic - Use the network as a generator

z comes from another simple distribution. At this point, network's input is not just a set of x, but x and z.

- The special thing about z is that it is not fixed, nd each time network is used, a random set of z

- The restriction of simple distribution is that it must be simple enough, i.e., we know how its formulation, and we can generate samples from this distribution

New Topic - Use the network as a generator

With different z, the output y varies

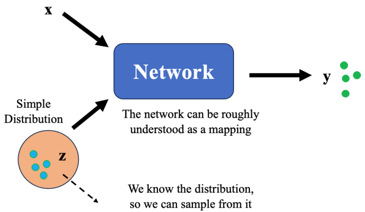


Figure 3: With different z, the output y varies

New Topic - Use the network as a generator

the output of the network is no longer a single value, but a complex distribution

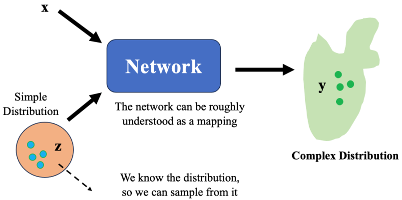


Figure 4: the output of the network is no longer a single value, but a complex distribution

> Generator: a network that can output a complex distribution

Why output a distribution?

Example Video Prediction: Source Link

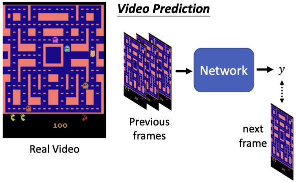


Figure 5: predict the next frame

Why output a distribution?

Example Video Prediction

How to implement video prediction?

1. Give your network the previous frames
2. The network should predict the elf in the corner should be to the left or to the right
3. And its output will be a new frame(next moment's frame)

It is not hard to do, you should only give the network enough previous frames, and then the network can be trained so that its output y is as close to our goal as possible.

What problems will it occur?

If you follow this method of training network, that is, supervisor learning training, the elf will split into two at the corner, and sometimes even disappear as they walk.

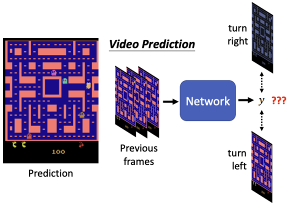


Figure 6: the elf split as he walked to the corner

What problems will it occur?

Why did the elf split as he walked?

For such a network, sometimes the same input, i.e. the same corner, the elf may go to the left or to the right. These two possibilities exist simultaneously in the training set.

When you are training your network, it is given the instruction to turn left given a piece of the training set, and turn right given another piece of the training set. When both are present in a training profile, your network learns both sides of the coin, because in this way, it can be closest to the left turn or the right turn at the same time $\rightarrow$ Turn left and right at the same time(split into 2)

The network will get the result, turning left is right, turning right is also right, but turning left and right at the same time is wrong instead.

How to deal with it?

Let the machine output be probabilistic, not singularly, but output a distribution of probabilities.

When we add a set of z to this network, its output can be a distribution.

That is, its output contains the possibility to turn left as well as right

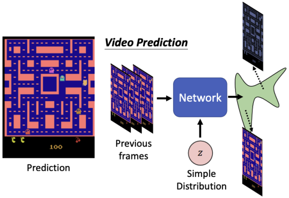


Figure 7: make the network output a set of probability

When do we need a generator?

When our mission requires creativity

- image compositing

- style transfer

- video generation

When do we need a generator?

When our mission requires creativity

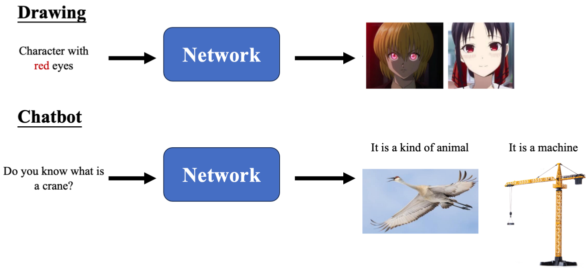


Figure 8: some example while using generator

Unconditional Generation

Here a series of vectors (from a normal distribution) are input to the network, and then the network generates a series of cartoon images.

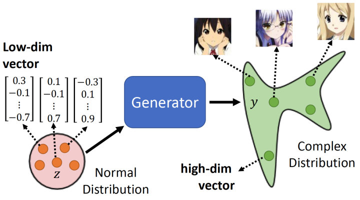


Figure 9: example:Unconditional Generation

Unconditional Generation

At this point, it is mandatory that whatever z is input, the output is cartoon characters.

- About normal distribution

In fact, the distribution on this side just needs to be simple enough, because your generator will figure out how to output cartoon characters(complex distribution).

discriminator

The special thing in GAN is that in addition to the generator, there is an additional discriminator to be trained.

What is the usage of a discriminator?

It will take a picture as input and the output is a value(scalar), which itself is a neural network and can be seen as a function.

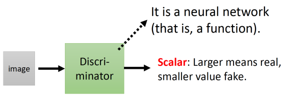


Figure 10: the usage of a discriminator

discriminator

The larger the scaler, the more the input image looks like a cartoon character.

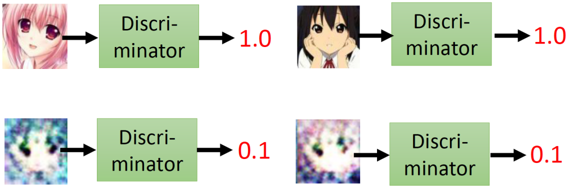


Figure 11: discriminate whether the input image is a cartoon character

Why discriminator

When we want to copy an image from the original, we want to evaluate the similarity with the original image, then we need the discriminator.

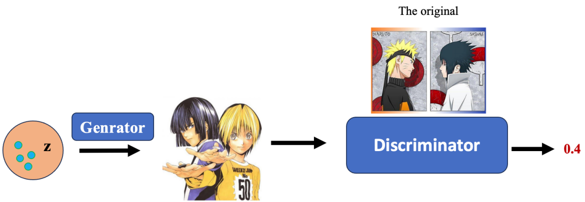


Figure 12: evaluate the similarity with the original

Non-cooperative game(Nash Equilibrium)

The sum of the interests of the two sides in a game is constant.

Figure 13: Example of Nash equilibrium: arm wrestling

For example, in a game of arm wrestling between two people, assuming the total space is fixed, if you are stronger, you will get more space and I will get less space accordingly. Conversely, if I am stronger, I will get more space and you will get less.

However, one thing is certain: the total space of the two of us is constant, which is the essence of a two-player game.

How generator learn from a picture and the copy?

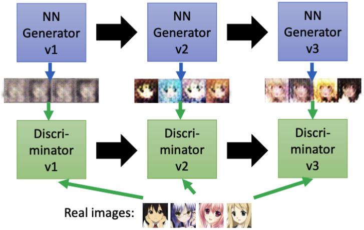
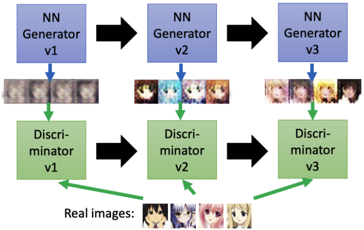


Figure 14: the equilibrium between generators and discriminators

1. At the beginning, the parameters of the first generation generator $G1$ were completely random, i.e. there was no way to know how to draw cartoon characters, so what was drawn was basically something inexplicable

How generator learn from a picture and the copy?



2. At this point, the first generation discriminator $D1$ has to do is to tell the difference between what the first generator draws and the real picture

3. With the information from the discriminator $D1$, the second generator $G2$ updates the parameters to adjust the target, with the aim of fooling the first-generation discriminator $D1$

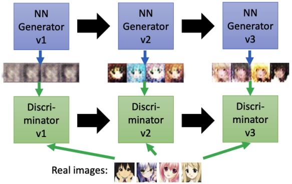
How generator learn from a picture and the copy?

3. With the information from the discriminator $D1$, the second generator $G2$ updates the parameters to adjust the target, with the aim of fooling the first-generation discriminator $D1$
- For example, If the first generation discriminator $D1$ determines whether it is a cartoon character by comparing whether it has eyes, the second generator $G2$ Will generate eyes to fool the first generation discriminator $D1$

4. However, the discriminator also evolves, so the first generation of discriminator will evolve into the second generation of discriminator $D2$
- For example, the second-generation discriminator $D2$ will evaluate if a picture is a cartoon character by determining whether or not it has hair and mouth

5.Then the third-generation generator $G2$ will try it best to updates the parameters to adjust the new target, with the aim of fooling the second-generation discriminator $D2$

How generator learn from a picture and the copy?



6. Looping through the above, the discriminator and generator will evolve together until the discriminator can't tell if it's a fake cartoon character.

The purpose of the generator and discriminator is exactly opposite, one can discriminate well and one wants to make the other discriminate badly, so it is what we call adversarial.

Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Global Optimum

Proposition For G fixed, the optimal discriminator D is

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity V(G, D)

$$\begin{aligned} V(G, D) &= \int_x p_{data}(x) log(D(x))dx + \int_z p_z(z) log(1 - D(g(z)))dz \\ &= \int_x (p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)))dx \end{aligned}$$

Global Optimum

$$V(G, D) = \int_x p_{data}(x) log(D(x))dx + \int_z p_z(z) log(1 - D(g(z)))dz$$
$$= \int_x (p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)))dx$$

$$f(y) = alog(y) + b(1 - log(y))$$

$f(y)$ is a concave function, so it has a maximum when $y = a/(a + b)$

So for any x, $D^*_G(x) = p_{data}(x)/(p_{data}(x) + p_g(x))$

Global Optimum

$$
\begin{aligned}
C(G) &= max_D V(G, D) \\
&= \mathbb{E}_{x\sim p_{data}}[log D_G^*(x)] + \mathbb{E}_{z\sim p_z}[log(1 - D_G^*(G(z)))] \\
&= \mathbb{E}_{x\sim p_{data}}[log D_G^*(x)] + \mathbb{E}_{x\sim p_g}[log(1 - D_G^*(x))] \\
&= \mathbb{E}_{x\sim p_{data}}[log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}] + \mathbb{E}_{x\sim p_g}[log \frac{p_g(x)}{p_{data}(x) + p_g(x)}]
\end{aligned}
$$

Theorem The global minimum of the virtual training criterion C(G) is achieved if and only if $p_g = p_{data}$. At that point, C(G) achieves the value $-log4$.

Global Optimum

the Kullback–Leibler divergence (also called relative entropy) is a type of statistical distance: a measure of how one probability distribution P is different from a second, reference probability distribution Q.

$$KL(p \parallel q) = \mathbb{E}_{x \sim p} log \frac{p(x)}{q(x)}$$

Property KL divergence is always non-negative, $KL(p \parallel q) = 0$ if and only if $p = q$ as measures.

Global Optimum

Proof of Thm

$$C(G) = \mathbb{E}_{x \sim p_{data}}[log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}] + \mathbb{E}_{x \sim p_g}[log \frac{p_g(x)}{p_{data}(x) + p_g(x)}]$$

$$= \mathbb{E}_{x \sim p_{data}}[log \frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x))}{2}}] + log \frac{1}{2} + \mathbb{E}_{x \sim p_g}[log \frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}}] + log \frac{1}{2}$$

$$= -log(4) + KL(p_{data} \parallel \frac{p_{data} + p_g}{2}) + KL(p_g \parallel \frac{p_{data} + p_g}{2})$$

So the minimum of C(G) is $-log4$ if and only if $p_{data} = \frac{p_{data}+p_g}{2}$ and $p_g = \frac{p_{data}+p_g}{2}$,

i.e. $p_{data} = p_g$ is the only solution.

Convergence

Proposition If G and D have enough capacity, and at each step of Algorithm, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion

$$\mathbb{E}_{x\sim p_{data}}[logD_G^*(x)] + E_{x\sim p_g}[log(1 - D_G^*(x))]$$

then p_g converges to p_{data}

Proof. Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $U(p_g, D)$ is convex in p_g. $sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof.