Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
3. GAN architectures

Drawing? => learning from examples
Review: Auto-encoder

As close as possible

Minimize reconstruction error

Randomly generate a vector as code

Image ?
Review: Auto-encoder
Review: Auto-encoder
Auto-encoder

From a normal distribution $N(0,1)$

Minimize reconstruction error

$\sigma_1 \epsilon_1 + m_1$

$\sigma_2 \epsilon_2 + m_2$

$\sigma_3 \epsilon_3 + m_3$

Problems of AE/VAE

- It does not really try to simulate real images

[Diagram showing the process of encoding, decoding, and output with real and non-realistic examples.]

One pixel difference from the target

Realistic

Non Realistic

As close as possible
Problems of AE/VAE

Game scenario:

Player1, Generator, produces samples
Player2, – Its adversary Discriminator, attempts to distinguish real samples from fake generated ones (produced by Player1)!

Player1 aims at producing Realistic images to fool the Player2
Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
   • GAN Algorithm
Adversarial Nets Framework

Game scenario:

Player1, Generator G
Player2, Discriminator D

\[ V(G, D) = \mathbb{E}_{x \sim P_{data}}[\log D(x)] + \mathbb{E}_{x \sim P_G}[\log (1 - D(x))] \]

\[ G^* = \arg \min_G \max_D V(G, D) \]
GAN Learning – D and G updates

Game scenario:

Player1, **Generator G**, produces samples
Player2, – Its adversary **Discriminator D**, attempts to distinguish real samples from fake generated ones (produced by Player1)!

Player1 aims at producing *Realistic* images to fool the Player2

Fake images: 2899

Real images: 5041
GAN - Discriminator

- Randomly sample a vector
- Generator v1

Real images:
- Discriminator v1

Discriminator Optimization on a batch of images:
Using gradient descent to update the parameters in the discriminator, with a fixed generator
GAN - Generator

Updating the parameters of generator

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

Generator + Discriminator = a network

Optimization:
Using gradient descent to update the parameters in the generator, but fixing the discriminator
GAN Learning – D and G updates

Real images: 5 0 4 1

Game over: Winner==Player1
Generator G producing fully Realistic images that fool the Player2
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 $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
        • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
        • Update the discriminator by ascending its stochastic gradient:
          \[
          \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log \left(1 - D(G(z^{(i)})) \right) \right].
          \]
    end for
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Update the generator by descending its stochastic gradient:
      \[
      \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(G(z^{(i)})) \right).
      \]
end for

GAN algorithm

\[
V = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{x \sim p_G} [\log (1 - D(x))]
\]

\[
G^* = \text{arg min}_{G} \max_{D} V(G, D)
\]
One example GAN

Source of images: https://zhuanlan.zhihu.com/p/24767059
DCGAN: https://github.com/carpedm20/DCGAN-tensorflow
GAN

100 rounds
GAN

1000 rounds
GAN

2000 rounds
GAN

5000 rounds
GAN

10,000 rounds
GAN

20,000 rounds
GAN

50,000 rounds
Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
   • GAN Algorithm
   • KL vs. Jensen Shanon Divergence

\[ V(G, D) = \mathbb{E}_{x \sim P_{data}}[\log D(x)] + \mathbb{E}_{x \sim P_G}[\log (1 - D(x))] \]

\[ G^* = \arg \min_G \max_D V(G, D) \]
Which measure to evaluate how $P_G(x; \theta)$ is close to $P_{data}(x)$ in Maximum Likelihood optimization?

- 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
Which measure to evaluate how $P_G(x; \theta)$ is close to $P_{data}(x)$ in Maximum Likelihood optimization?

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

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

$$\approx \arg \max_{\theta} \mathbb{E}_{x \sim P_{data}} [\log P_G(x; \theta)]$$

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

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

In Maximum Likelihood it is a KLD Kullback Leibler Divergence
If $P_G(x; \theta)$ is a coming with a NN

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

It is difficult to compute the likelihood.

Credits: https://blog.openai.com/generative-models/
Basic Idea of GAN: the 2 players G-D game

- **Generator G**
  - Hard to learn by maximum likelihood
  - 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 (and $P_{\text{prior}}$)

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

- Global objective function $V(G,D)$

$$\theta^* = G^* = \arg \min_G \max_D V(G,D)$$
Basic Idea

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

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

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

• Given G, what is the optimal D* maximizing

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

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

\[
= \int \left[ P_{\text{data}}(x) \log D(x) + P_G(x) \log (1 - D(x)) \right] \, dx
\]

Assume that D(x) can have any value here

• Given x, the optimal D* maximizing

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

- Given \( x \), the optimal \( D^* \) maximizing:
  \[ P_{data}(x) \log D(x) + P_G(x) \log (1 - D(x)) \]
  \[ a \quad D \quad b \quad D \]

- Find \( D^* \) maximizing:
  \[ f(D) = a \log(D) + b \log(1 - D) \]
  \[ \frac{df(D)}{dD} = a \times \frac{1}{D} + b \times \frac{1}{1 - D} \times (-1) = 0 \]

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

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

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

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

“difference” between \( P_{G_1} \) and \( P_{\text{data}} \)
\[
\max_D V(G, D) = V(G, D^*)
\]

\[
= \mathbb{E}_{x \sim P_{data}} \left[ \log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} \right] + \mathbb{E}_{x \sim P_G} \left[ \log \frac{P_G(x)}{P_{data}(x) + P_G(x)} \right]
\]

\[
= \int_x P_{data}(x) \log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} \, dx + \int_x P_G(x) \log \frac{P_G(x)}{P_{data}(x) + P_G(x)} \, dx
\]

\[
=> +2 \log \frac{1}{2} = -2 \log 2
\]

\[
V = \mathbb{E}_{x \sim P_{data}} [\log D(x)] + \mathbb{E}_{x \sim P_G} [\log (1 - D(x))]
\]
\[
\max_D V(G, D) \quad \text{max } V(G, D) = V(G, D^*) \quad D^*(x) = \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_G(x)}
\]

\[
= -2\log 2 + \int \frac{P_{\text{data}}(x) \log \left( \frac{P_{\text{data}}(x)}{\left( P_{\text{data}}(x) + P_G(x) \right)/2} \right)}{\left( P_{\text{data}}(x) + P_G(x) \right)/2} dx
\]

\[
+ \int \frac{P_G(x) \log \left( \frac{P_G(x)}{\left( P_{\text{data}}(x) + P_G(x) \right)/2} \right)}{\left( P_{\text{data}}(x) + P_G(x) \right)/2} dx
\]

\[
= -2\log 2 + \text{KL} \left( P_{\text{data}}(x) \left\| \frac{P_{\text{data}}(x) + P_G(x)}{2} \right\| \right)
\]

\[
+ \text{KL} \left( P_G(x) \left\| \frac{P_{\text{data}}(x) + P_G(x)}{2} \right\| \right)
\]

\[
= -2\log 2 + 2\text{JSD} \left( P_{\text{data}}(x) \left\| P_G(x) \right\| \right) \quad \text{Jensen-Shannon divergence}
\]
In the end ......

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

• Generator G, Discriminator D
• Looking for G* such that  \[ G^* = \arg \min_G \max_D V(G, D) \]
• Given G, \( \max_D V(G, D) = -2 \log 2 + 2 \text{JS}D(P_{data}(x) \| P_G(x)) \)

• What is the optimal G?

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

with/using the JS\((P_G, P_{data})\) Divergence

(In Maximum Likelihood it is a KL Divergence)