0. Notations

x : attribute space/sample space.

$P_{data}(x)$: the target data distribution that we want to find.

$P_G(x;\theta)$: the generated data distribution which is designed to stay close to $P_{data}(x)$. θ is an adjustable parameter (eg: in Gaussian Mixture Model, θ is the mean and variance of the Gaussian)

> GMM: Check it out in Saraj Ravel's video. This YouTuber is worthy of subscription.
>
> The reason why GMM doesn't work on some problems (image classification, speech recognition, etc.) is: the distribution of some complex corpus (images, wideos, speeches, etc.) are not Gaussian (mentioned in Hongyi Li's lecture, he did not give evidence or references).
>
> You can also compare GMM with Fourier Transformation.

$G(x)$: Generator, part of GAN. Its input x samples from a distribution, the output is an attribute/a sample. $P_{normal}(\theta)$: A normal distribution whose mean and variance is denoted by θ.

1. Original GAN Explained

Based on Generative Adversarial Nets.

I like the mathematical modeling method revealed in this paper. It's rare to find in recent journals.

1.1 Basic problem focused

How to find $P_{data}(x)$? That is, how to find a proper $\theta^* = arg\min_\theta Div(P_G(x;\theta), P_{data}(x))$?

> Divergence (statistics) is a function which establishes the "distance" of one probability distribution to the other on a statistical manifold. That is a weaker notion than that of the distance.

1.2 Former method: Maximum Likelihood Estimation

First, we sample $\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$. Assume that m is big enough (line 4), Then several conclusions can be made:

- All samples are in $P_G(x;\theta)$.
- Maximum Likelihood Estimation $\approx$ Minimize KL Divergence.

$$
\begin{aligned}
\theta^* &= arg_\theta^{max} \prod_{i=1}^{m} P_G(x^i;\theta) \\
&= arg_\theta^{max} log(\prod_{i=1}^{m} P_G(x^i;\theta)) \\
&= arg_\theta^{max} \sum_{i=1}^{m} log(P_G(x^i;\theta)) \\
&\approx arg_\theta^{max} E_{x\sim P_{data}} log(P_G(x;\theta)) \\
&= arg_\theta^{max} \int_x P_{data}(x) log(P_G(x;\theta)) \\
&= arg_\theta^{max} \int_x P_{data}(x) log(P_G(x;\theta)) - \int_x P_{data}(x) log(P_{data}(x)) \\
&= -arg_\theta^{max} KL(P_{data}||P_G) \\
&= arg_\theta^{min} KL(P_{data}||P_G)
\end{aligned}
$$

> Kullback-Leibler Divergence (Relative Entrophy) is a measure of how one probability distribution (the latter) is different from a second, reference probability distribution (the former).

1.3 Definition: Generator

It's hard to model $P_G(x;\theta)$. In pre deep-learning era, we use GMM to define $P_G(x;\theta)$, leading a poor quality of generated pictures.

A generator G is a network which defines a probability distribution P_G: assume that $z\sim P_{normal}(\theta)$, then $G(z)\sim P_G$. That is similar to the "domain transfer" concept.

The best generator G^* can be defined as: $G^* = arg_G^{min} Div(P_G, P_{data})$.

We can not know the exact formulae of both P_G and P_{data}, so we need to use "discriminator" to compute the divergence mentioned above. That is the essence of GAN.

1.4 Definition: Discriminator

A good discriminator $D(x)$ can 1) score samples sampled from P_G as low as possible and 2) score samples sampled from P_{data} as high as possible.

The best discriminator D^* can be defined as:

$$
\begin{aligned}
V(G,D) &= E_{x\sim P_{data}} logD(x) + E_{x\sim P_G} log(1-D(x)) \\
D^* &= arg_D^{max} V(G,D)
\end{aligned}
$$

> The objective function $V(G,D)$ is exactly the same as the objective function applied to a binary classifier.

1.5 Finding the best generator by using discriminator

So, how to compute $max(V(G,D))$ or $V(G,D^*)$?

$$
\begin{aligned}
V(G,D) &= E_{x\sim P_{data}} logD(x) + E_{x\sim P_G} log(1-D(x)) \\
&= \int_x P_{data}(x) logD(x) dx + \int_x P_G(x) log(1-D(x)) dx \\
&= \int_x [P_{data}(x) logD(x) + P_G(x) log(1-D(x))] dx
\end{aligned}
$$

Assume that $D(x)$ can be any function, which means: x is fixed and $D(x)$ is not fixed. Define $f(D(x))$ as:

$$
f(D(x)) = P_{data}(x) logD(x) + P_G(x) log(1-D(x))
$$

So, maximizing $V(G,D)$ equals maximizing $f(D(x))$.

$$
\begin{aligned}
\frac{df(D(x))}{dD(x)} &= P_{data}(x) \cdot \frac{1}{D(x)} + P_G(x) \cdot \frac{1}{1-D(x)} \cdot (-1) = 0 \\
D^*(x) &= \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}
\end{aligned}
$$

As a result:

$$
\begin{aligned}
maxV(G,D) &= V(G,D^*) \\
&= E_{x\sim P_{data}} log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} + E_{x\sim P_G} log \frac{P_G(x)}{P_{data}(x) + P_G(x)} \\
&= \int_x P_{data}(x) log \frac{\frac{1}{2} P_{data}(x)}{\frac{1}{2}(P_{data}(x) + P_G(x))} + \int_x P_G(x) log \frac{\frac{1}{2} P_G(x)}{\frac{1}{2}(P_{data}(x) + P_G(x))} \\
&= -2log2 + \int_x P_{data}(x) log \frac{P_{data}(x)}{\frac{1}{2}(P_{data}(x) + P_G(x))} + \int_x P_G(x) log \frac{P_G(x)}{\frac{1}{2}(P_{data}(x) + P_G(x))} \\
&= -2log2 + KL(P_{data} || \frac{P_{data}}{P_{data} + P_G}) + KL(P_G || \frac{P_{data}}{P_{data} + P_G}) \\
&= -2log2 + 2JSD(P_{data} || P_G)
\end{aligned}
$$

> Jensen-Shannon Divergence is also a method of measuring the similarity between two probability distributions. It is based on KL Divergence with some notable differences (symmetric and always finite).

So, $V(G,D^*)$ is one kind of divergence, which can denote $Div(P_{data}, P_G)$.

To conclude:

$$
\begin{aligned}
G^* &= arg_G^{min} Div(P_{data}, P_G) \\
&= arg_G^{min} {}_D^{max} V(G,D)
\end{aligned}
$$

That is the end of hard mathematics.

1.6 How to play the min-max game

It's time to play with algorithms.

1.6.1 Methodology

First, fix the parameters of *G* and try to train *D*; Then, fix the parameters of *D* and try to train *G*. Iterate the procedure.

$$G^* = arg_G^{min} {}_D^{max} V(G,D)$$

Define $L(G) =_D^{max} V(G,D)$, then we need to calculate $\theta_G \leftarrow \theta_G - \eta \frac{\partial L(G)}{\partial \theta_G}$:

- How to differentiate $L(G)$?

Every time when θ_G is upgraded ,check its region.

For several starting iterations:

- Given G_0;
- Find D_0^* maximizing $V(G_0, D)$;
  - $\theta_G \leftarrow \theta_G - \eta \frac{\partial V(G_0, D_0^*)}{\partial \theta_G}$, so we can obtain G_1;
- Find D_1^* maximizing $V(G_1, D)$;
  - $\theta_G \leftarrow \theta_G - \eta \frac{\partial V(G_1, D_1^*)}{\partial \theta_G}$, so we can obtain G_2;

Discussion: Are these iterations really decrease JS Divergence?

> There is a strong assumption in original paper: Assume that $D_0^* \approx D_1^*$.
>
> To make this possible, we should train discriminator more often and train generator less.

1.6.2 General algorithm

For each training iteration:

[learning discriminator]

- sample m examples $(x^1, x^2, \ldots, x^m)$ from data distribution;
- sample m examples $(z^1, z^2, \ldots, z^m)$ from the prior distribution;
- obtain generated data $(\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^n)$, $\hat{x}^i = G(z^i)$;
- Update discriminator parameters θ_D to maximize
  - $\hat{V} = \frac{1}{m} \sum_{i=1}^{m} (logD(x^i) + log(1 - D(\hat{x}^i)))$
  - $\theta_D \leftarrow \theta_D + \eta \nabla \hat{V}(\theta_D)$ [repeat k times]

[learning generator]

- sample another m examples $(z^1, z^2, \ldots, z^m)$ from the prior distribution;
- Update generator parameters θ_G to minimize
  - $\hat{V} = \frac{1}{m} \sum_{i=1}^{m} (logD(x^i) + log(1 - D(\hat{x}^i)))$
  - $\theta_G \leftarrow \theta_G - \eta \nabla \hat{V}(\theta_G)$

[repeat only once]