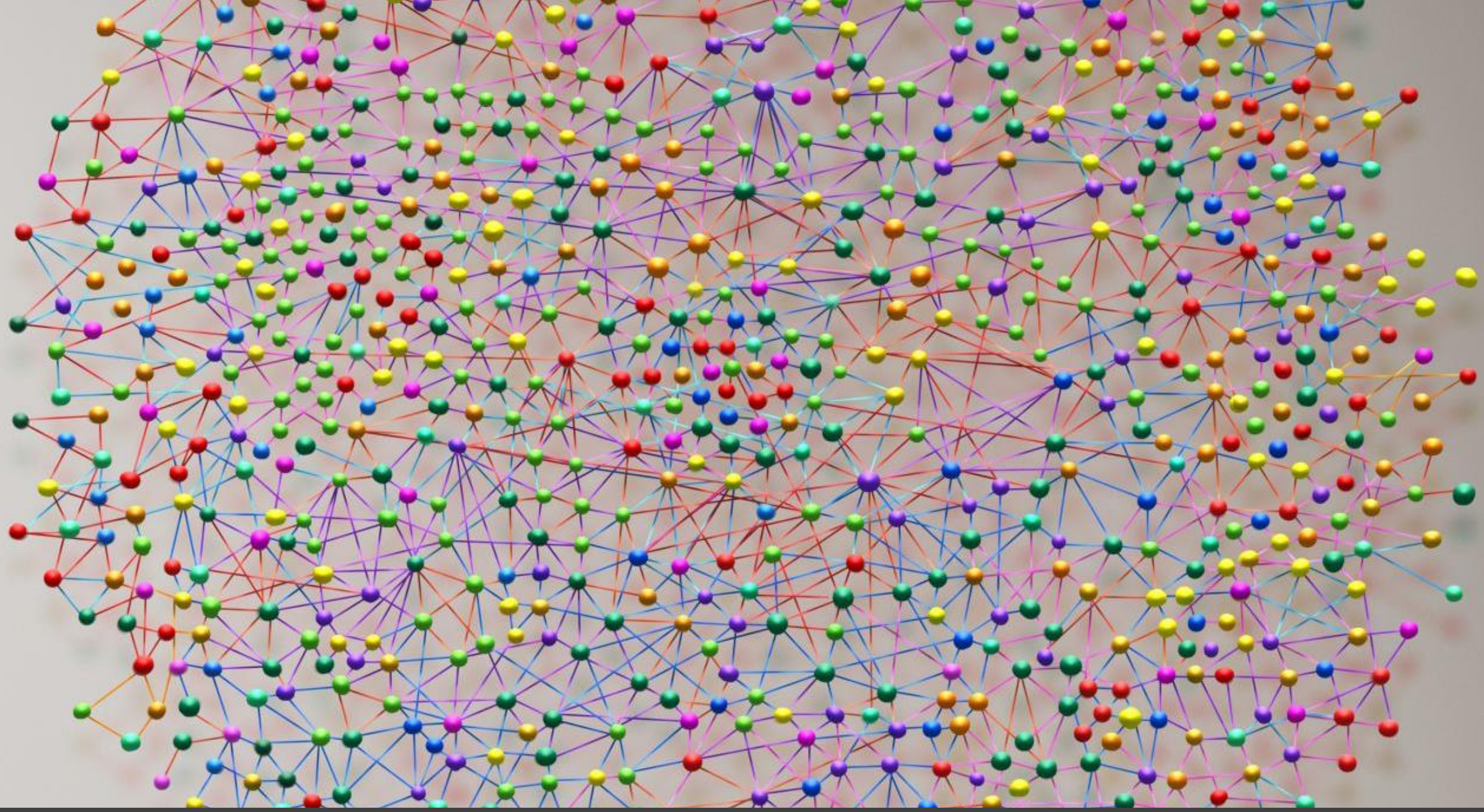


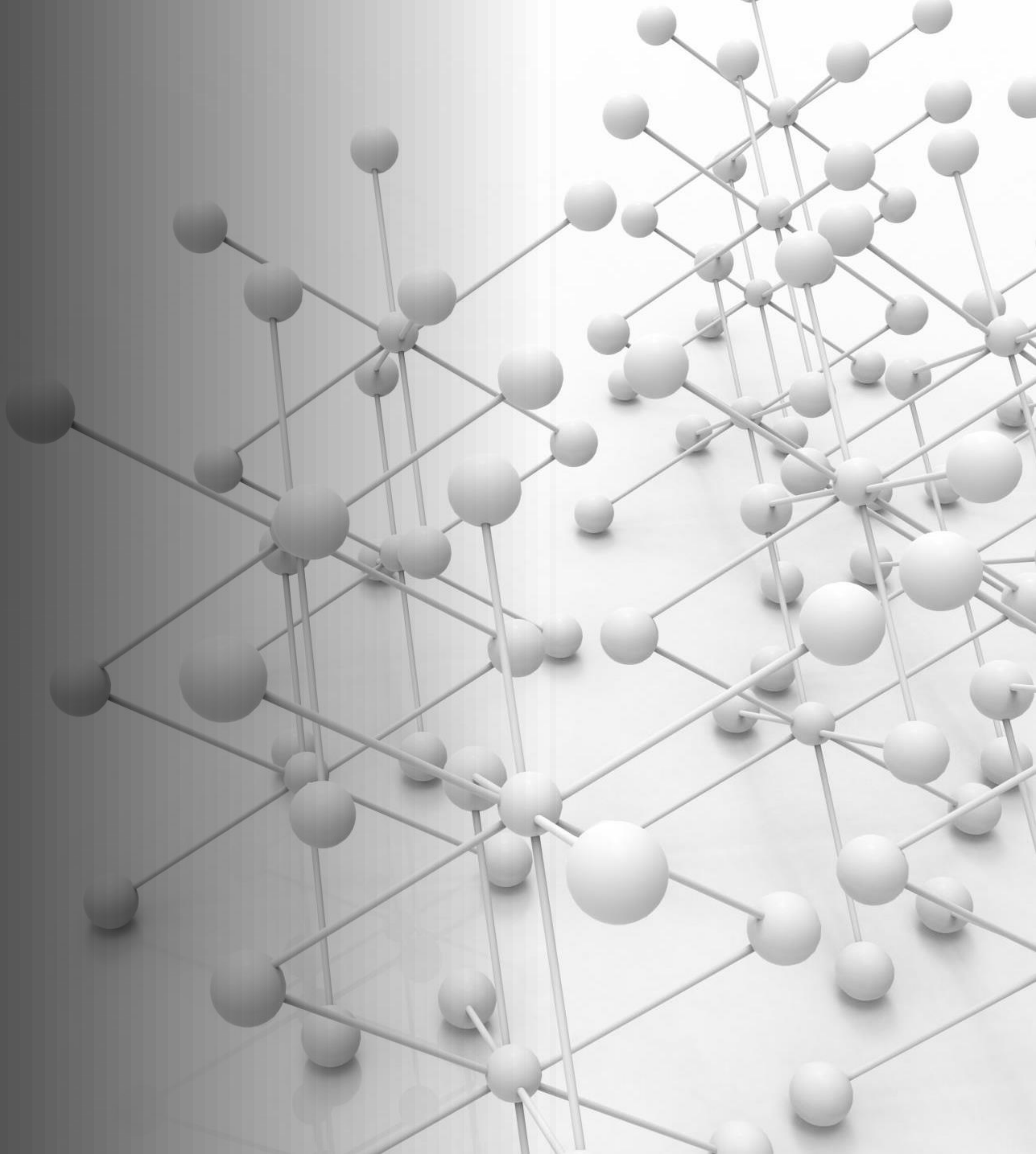
Generative Adversarial Networks

From Kullback-Leibler Divergence to Vanilla Generative Adversarial Networks

List of contents

Acknowledgements: Lilian Weng from OpenAI & François Chollet from Google

What is a Generative Adversarial Network?



Kullback-Lieber Divergence

Kullback-Lieber Divergence (KLD) aims to measure how a probability distribution p diverges from a second expected probability distribution q so that:

$$D_{KL}(p||q) = \int_x p(x) log \frac{p(x)}{q(x)} dx$$

Kullback-Lieber Divergence

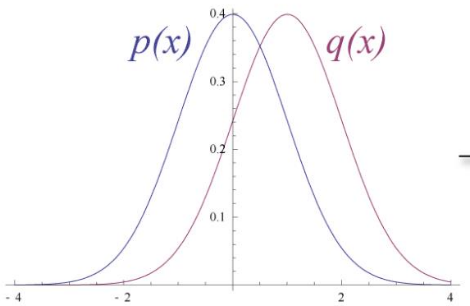
Kullback-Lieber Divergence (KLD) aims to measure how a probability distribution p diverges from a second expected probability distribution q so that:

$$D_{KL}(p||q) = \int_x p(x) log \frac{p(x)}{q(x)} dx$$

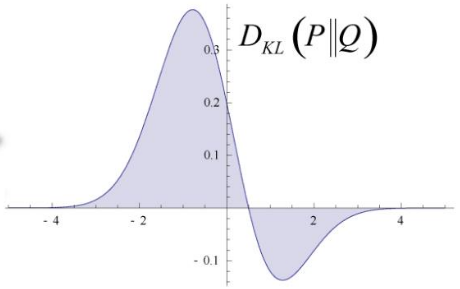
D_{KL} achieves the minimum when $p(x)$ = $q(x)$

Kullback-Lieber Divergence

According to its formulation, KLD is asymmetric by design



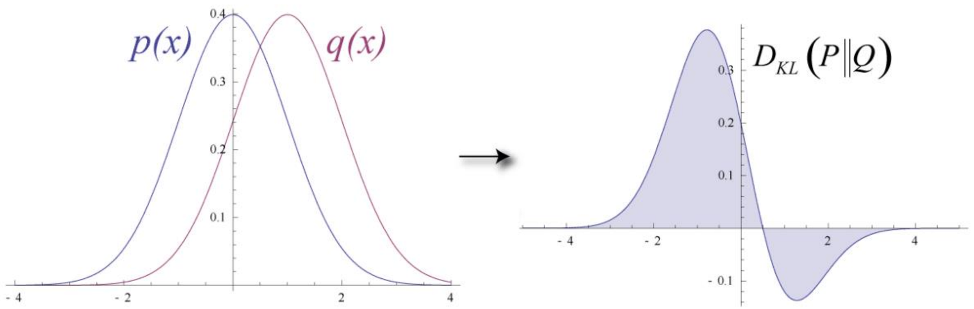
Original Gaussian PDFs



KL Area to be Integrated

Kullback-Lieber Divergence

According to its formulation, KLD is asymmetric by design



Original Gaussian PDFs KL Area to be Integrated

In cases where $p(x)$ is close to zero while $q(x)$ is significantly non-zero, the $q(x)$ effect is ignored. This property may cause undesirable results when we just want to measure the similarity between two equally important distributions

Jensen-Shannon Divergence

Jensen-Shannon Divergence (KLD) aims to measure how a probability distribution p diverges from a second expected probability distribution q so that:

$$D_{JS}(p\|q) = \frac{1}{2} D_{KL}\left(p\| \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q\| \frac{p+q}{2}\right)$$

Jensen-Shannon Divergence

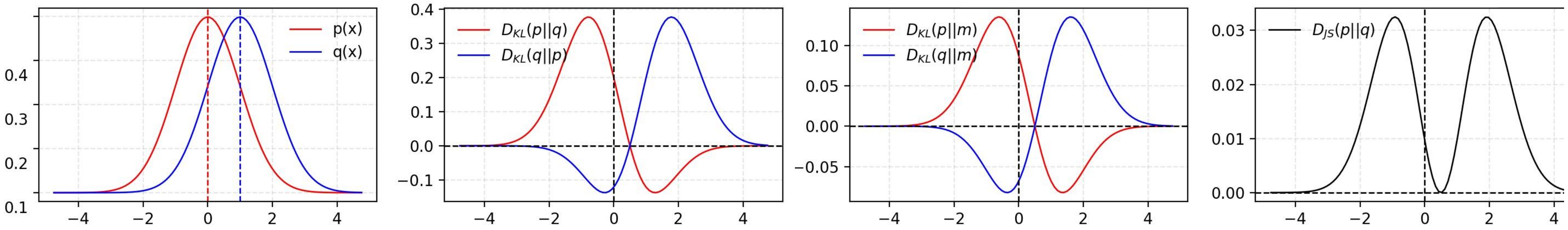
Jensen-Shannon Divergence (KLD) aims to measure how a probability distribution p diverges from a second expected probability distribution q so that:

$$D_{JS}(p \| q) = \frac{1}{2} D_{KL}\left(p \| \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \| \frac{p+q}{2}\right)$$

D_{JS} achieves the minimum when $p(x) = q(x)$

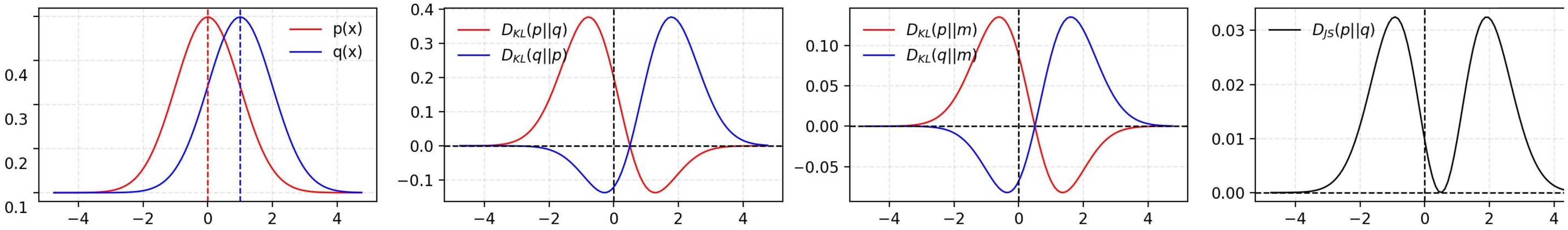
Jensen-Shannon Divergence

According to its formulation, JSD is symmetric by design, is bounded in the [0,1] range, and is smoother than KLD



Jensen-Shannon Divergence

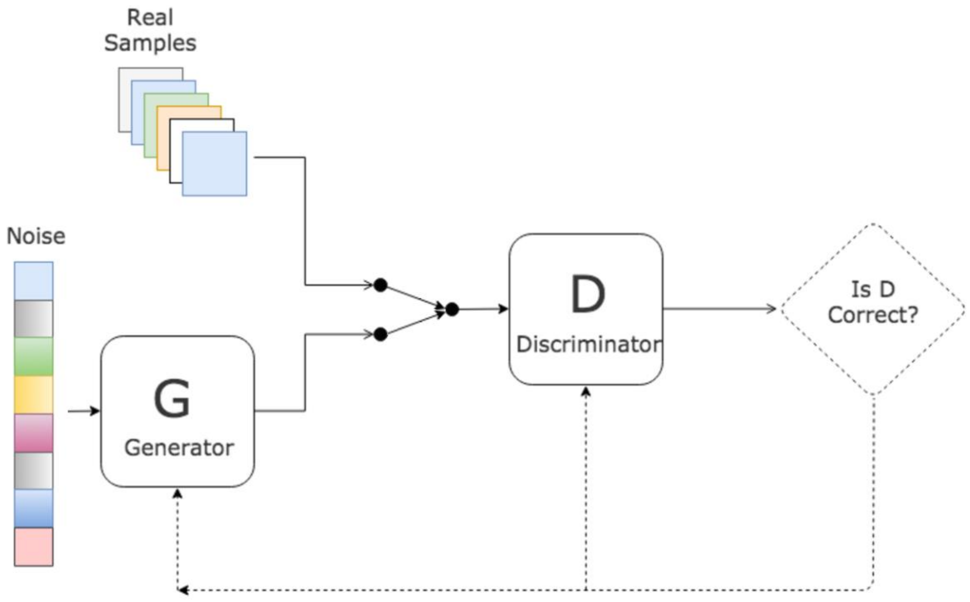
According to its formulation, JSD is symmetric by design, is bounded in the [0,1] range, and is smoother than KLD



One reason behind Generative Adversarial Networks success is switching the loss function from asymmetric KLD in traditional maximum-likelihood approach to symmetric JSD

Generative Adversarial Networks (Introduction)

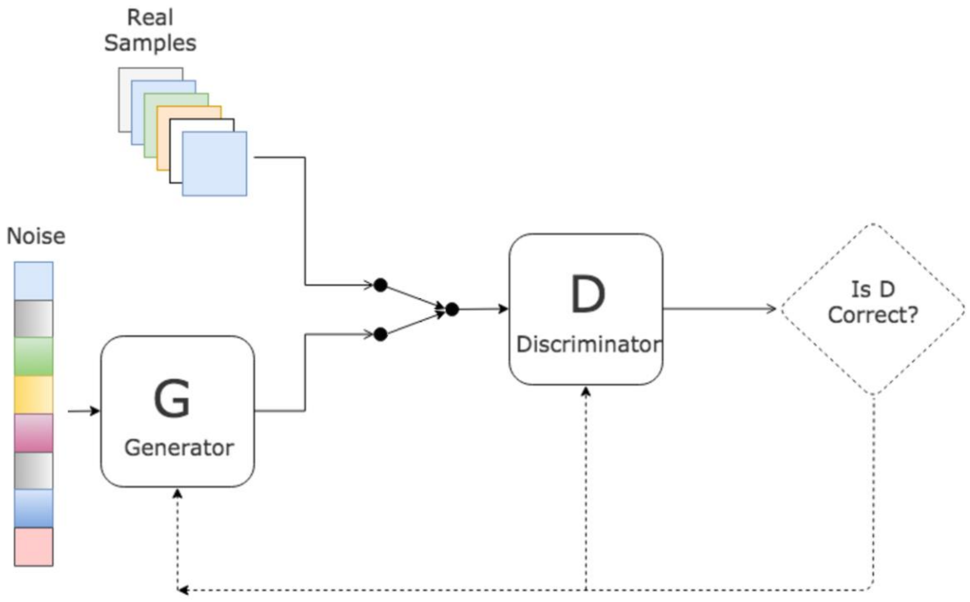
A Generative Adversarial Network (GAN) is an artificial neural network composed of two models:



Generative Adversarial Networks (Introduction)

A Generative Adversarial Network (GAN) is an artificial neural network composed of two models:

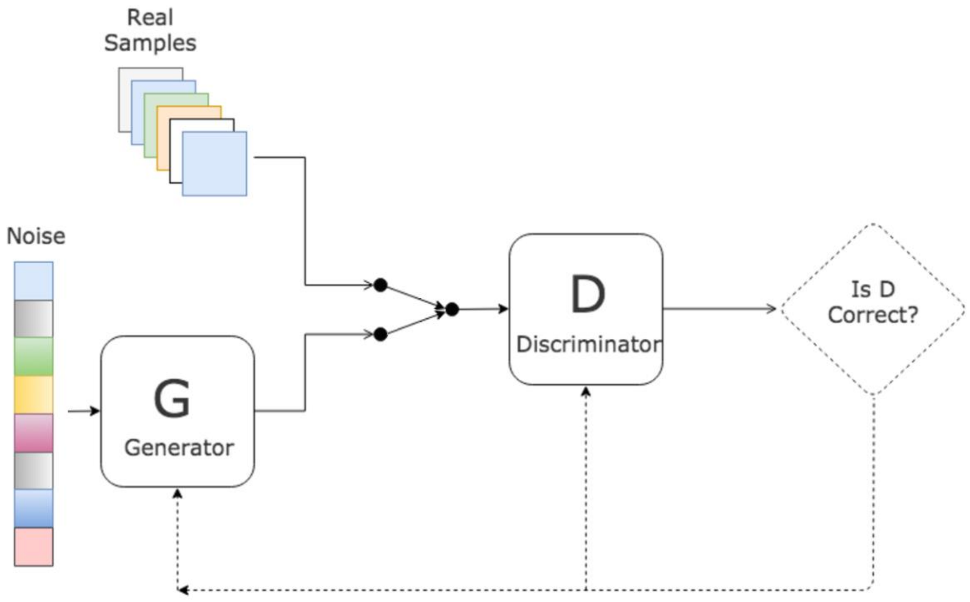
- A discriminator D to estimate the probability of a given sample coming from the real dataset. It works as a critic and is optimized to discriminate the fake samples from the real ones



Generative Adversarial Networks (Introduction)

A Generative Adversarial Network (GAN) is an artificial neural network composed of two models:

- A discriminator D to estimate the probability of a given sample coming from the real dataset. It works as a critic and is optimized to discriminate the fake samples from the real ones
- A generator G to output synthetic samples given a noise variable input z. It is trained to capture the real data distribution so that its generative samples can be as real as possible, or in other words, can trick the discriminator to offer a high probability



Generative Adversarial Networks (Formulation)

Given a noise variable z (usually normally or uniformly distributed) and a dataset x, we define the following distributions:

Generative Adversarial Networks (Formulation)

Given a noise variable z (usually normally or uniformly distributed) and a dataset x, we define the following distributions:

- p_z as the data distribution over noise z

Generative Adversarial Networks (Formulation)

Given a noise variable z (usually normally or uniformly distributed) and a dataset x, we define the following distributions:

- p_z as the data distribution over noise z
- p_g as the generator's distribution over data x

Generative Adversarial Networks (Formulation)

Given a noise variable z (usually normally or uniformly distributed) and a dataset x, we define the following distributions:

- p_z as the data distribution over noise z
- p_g as the generator's distribution over data x
- p_r as the data distribution over real samples x

Generative Adversarial Networks (Formulation)

We want to improve the discriminator D ability to accurately recognize real data coming from distribution $p_r(x)$ maximizing $\mathbb{E}_{x \sim p_r(x)}[logD(x)]$

Generative Adversarial Networks (Formulation)

We want to improve the discriminator D ability to accurately recognize real data coming from distribution $p_r(x)$ maximizing $\mathbb{E}_{x \sim p_r(x)}[logD(x)]$

Moreover, given the fake samples $G(z) \sim p_z(z)$ is expected to output a probability $D\left(G(z)\right)$ close to zero maximizing $\mathbb{E}_{z \sim p_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$

Generative Adversarial Networks (Formulation)

We want to improve the discriminator D ability to accurately recognize real data coming from distribution $p_r(x)$ maximizing $\mathbb{E}_{x \sim p_r(x)}[logD(x)]$

Moreover, given the fake samples $G(z) \sim p_z(z)$ is expected to output a probability $D\big(G(z)\big)$ close to zero maximizing $\mathbb{E}_{z \sim p_z(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$

The generator is instead trained to increase its chances of producing good quality fake examples and with a high probability managing to deceive D. For this purpose, its goal is to minimize $\mathbb{E}_{z \sim p_z(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right] = \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x))]$

Generative Adversarial Networks (Formulation)

D and G are playing a minimax game in which we should optimize the following loss function:

$$\min_G \max_D L(G, D) = \mathbb{E}_{x \sim p_r(x)}\left[\log\left(D(x)\right)\right] + \mathbb{E}_{x \sim p_g(x)}\left[\log\left(1 - D(x)\right)\right]$$

Generative Adversarial Networks (Formulation)

D and G are playing a minimax game in which we should optimize the following loss function:

$$\min_G \max_D L(G, D) = \mathbb{E}_{x \sim p_r(x)}\big[\log\big(D(x)\big)\big] + \mathbb{E}_{x \sim p_g(x)}\big[\log\big(1 - D(x)\big)\big]$$

where $\mathbb{E}_{x \sim p_r(x)}\big[\log\big(D(x)\big)\big]$ has no impact on G during gradient descent updates

Generative Adversarial Networks (Properties)

Given the minmax loss function just defined, the optimal value of $D(x)$ can be found solving the following integral:

$$L(G, D) = \int_x \left(p_r(x) \log\big(D(x)\big) + p_g(x) \log\big(1 - D(x)\big) \right) dx$$

Generative Adversarial Networks (Properties)

Given the minmax loss function just defined, the optimal value of $D(x)$ can be found solving the following integral:

$$L(G, D) = \int_x \left(p_r(x)\log\big(D(x)\big) + p_g(x)\log\big(1 - D(x)\big)\right) dx$$

Let $\tilde{x} = D(x), A = p_r(x), B = p_g(x)$. Since we can safely ignore the integral as x is sampled over all the possible values, the argument of the integral $f(\tilde{x})$ becomes:

$$f(\tilde{x}) = A\log(\tilde{x}) + B\log(1 - \tilde{x})$$

Generative Adversarial Networks (Properties)

By deriving with respect to $\tilde{x}$ we obtain that:

$$\frac{df(\tilde{x})}{d\tilde{x}} = A\frac{1}{ln10}\frac{1}{\tilde{x}} - B\frac{1}{ln10}\frac{1}{1-\tilde{x}} = \frac{1}{ln10}\frac{A-(A+B)\tilde{x}}{\tilde{x}(1-\tilde{x})}$$

Generative Adversarial Networks (Properties)

By deriving with respect to $\tilde{x}$ we obtain that:

$$\frac{df(\tilde{x})}{d\tilde{x}} = A\frac{1}{ln10}\frac{1}{\tilde{x}} - B\frac{1}{ln10}\frac{1}{1-\tilde{x}} = \frac{1}{ln10}\frac{A-(A+B)\tilde{x}}{\tilde{x}(1-\tilde{x})}$$

Thus, by setting $\frac{df(\tilde{x})}{d\tilde{x}} = 0$, we get the best value for the discriminator:

$$D^*(x) = \tilde{x} = \frac{A}{A+B} = \frac{p_r(x)}{p_r(x)+p_g(x)} \in [0,1]$$

Generative Adversarial Networks (Properties)

By deriving with respect to $\tilde{x}$ we obtain that:

$$\frac{df(\tilde{x})}{d\tilde{x}} = A\frac{1}{ln10}\frac{1}{\tilde{x}} - B\frac{1}{ln10}\frac{1}{1-\tilde{x}} = \frac{1}{ln10}\frac{A-(A+B)\tilde{x}}{\tilde{x}(1-\tilde{x})}$$

Thus, by setting $\frac{df(\tilde{x})}{d\tilde{x}} = 0$, we get the best value for the discriminator:

$$D^*(x) = \tilde{x} = \frac{A}{A+B} = \frac{p_r(x)}{p_r(x)+p_g(x)} \in [0,1]$$

Once the generator is trained to its optimal $p_g \approx p_r$ and then $D^*(x) = \frac{1}{2}$

Generative Adversarial Networks (Properties)

When both G and D are at their optimal values, we have $p_g = p_r$ and $D^*(x) = \frac{1}{2}$ and the loss function becomes:

$$L(G^*, D^*) = \int_x \left(p_r(x)\log\big(D^*(x)\big) + p_g(x)\log\big(1 - D^*(x)\big)\right) dx$$

Generative Adversarial Networks (Properties)

When both G and D are at their optimal values, we have $p_g = p_r$ and $D^*(x) = \frac{1}{2}$ and the loss function becomes:

$$L(G^*, D^*) = \int_x \Big(p_r(x)\log\big(D^*(x)\big) + p_g(x)\log\big(1 - D^*(x)\big)\Big)\,dx$$

$$= \log\frac{1}{2}\int_x p_r(x)dx + \log\frac{1}{2}\int_x p_g(x)dx$$

Generative Adversarial Networks (Properties)

When both G and D are at their optimal values, we have $p_g = p_r$ and $D^*(x) = \frac{1}{2}$ and the loss function becomes:

$$L(G^*, D^*) = \int_x \Big(p_r(x)\log\big(D^*(x)\big) + p_g(x)\log\big(1 - D^*(x)\big)\Big)\,dx$$

$$= \log\frac{1}{2}\int_x p_r(x)dx + \log\frac{1}{2}\int_x p_g(x)dx$$

$$= -2\log 2$$

Generative Adversarial Networks (Properties)

According to the previously defined formula, the Jensen-Shannon Divergence between p_r and p_g can be written as:

$$D_{JS}(p_r \| p_g) = \frac{1}{2} D_{KL}\left(p_r \| \frac{p_r+p_g}{2}\right) + \frac{1}{2} D_{KL}\left(p_g \| \frac{p_r+p_g}{2}\right)$$

Generative Adversarial Networks (Properties)

According to the previously defined formula, the Jensen-Shannon Divergence between p_r and p_g can be written as:

$$D_{JS}(p_r \| p_g) = \frac{1}{2} D_{KL}\left(p_r \middle\| \frac{p_r + p_g}{2}\right) + \frac{1}{2} D_{KL}\left(p_g \middle\| \frac{p_r + p_g}{2}\right)$$

$$= \frac{1}{2}\left(\log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} dx\right) + \frac{1}{2}\left(\log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r(x) + p_g(x)} dx\right)$$

Generative Adversarial Networks (Properties)

According to the previously defined formula, the Jensen-Shannon Divergence between p_r and p_g can be written as:

$$D_{JS}(p_r \| p_g) = \frac{1}{2} D_{KL}\left(p_r \middle\| \frac{p_r + p_g}{2}\right) + \frac{1}{2} D_{KL}\left(p_g \middle\| \frac{p_r + p_g}{2}\right)$$

$$= \frac{1}{2}\left(\log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} dx\right) + \frac{1}{2}\left(\log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r(x) + p_g(x)} dx\right)$$

$$= \frac{1}{2}\left(\log 4 + L(G, D^*)\right)$$

Generative Adversarial Networks (Properties)

Essentially the loss function of a GAN quantifies the similarity between the generative data distribution p_g and the real sample distribution p_r by JSD when the discriminator is optimal:

$$L(G, D^*) = 2D_{JS}(p_r \| p_g) - 2\log 2$$

Generative Adversarial Networks (Properties)

Essentially the loss function of a GAN quantifies the similarity between the generative data distribution p_g and the real sample distribution p_r by JSD when the discriminator is optimal:

$$L(G, D^*) = 2D_{JS}\left(p_r \| p_g\right) - 2\log 2$$

The best G^*that replicates the real data distribution leads the loss function to reach the following minimum:

$$L(G^*, D^*) = -2\log 2$$