Objective: Develop a Mathematical Foundation for Gen AI.

Family of DGMs we'll look at:

1. Generative Adversarial Network (GAN)
2. Variational Auto-Encoder (VAE)
3. Denoising Diffusion probabilistic Models (DDPM)
4. Auto-Regressive Models (AR)
   Large Language Models (LLM)
5. State-space Models (SSM)
   S4, Mamba.
6. RL-Based Alignment for LLMs.
   RLHF, PPO, DPO.

Generative Models:

Examples: ChatGPT, Claude, Gemini, etc...
Conditional Text Generators.

DallE, Stable Diffusion.
Conditional Image Generators.

Speech Generators : text $\longrightarrow$ wav

Mathematical Formulation of the Problem:

Starting point : Data.

Data $D = \{x_1, x_2, \dots, x_n\} \overset{iid}{\sim} \mathbb{P}_x$ (Unknown dist.)

$x_i \in \mathbb{R}^d$, d-dimensionality of the data.

X : Random Variable with a distribution $\mathbb{P}_x$.

Suppose $D = \{x_1, x_2, \dots x_n\}_{n=1000}$, then

x_i, x_j are statistically independent and are sampled from the same distribution. $\longrightarrow$ Assumptions.

$X_i \perp\!\!\!\perp X_j$, $X_i \sim \mathbb{P}_x$

Why?? For Mathematical ease.

X_i : Instances of a Vector-Valued Random Variable of size 'd'.

Goal: Estimate $\mathbb{P}_x$ & learn to sample from it.

Not all models explicitly figure out $\mathbb{P}_x$. They may generate $\mathbb{P}_x$ in implicit manner.

But almost all of them learn to sample from the underlying distribution.

General principle of Gen Models:

Density function.

i) Assume a parametric family on P_x, denoted by P_θ.

P_θ: Represented using Deep NNs. (Model)

(ii) Define & Compute a divergence metric between the P_θ & P_x → We dunno P_x tho??

(iii) Solve an optimization problem over the parameters of P_θ, to minimize the above divergence metric.

The task of Generative modelling is 2-fold.

a) Estimate the dist.

b) sample from it.

Using the above 3-step process, we accomplished

ⓐ. what about ⓑ ??

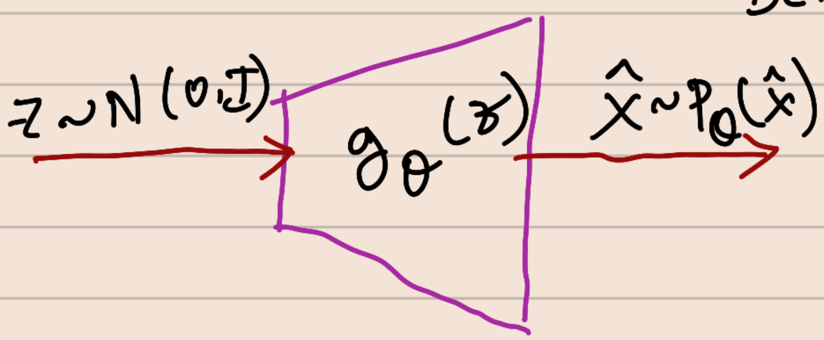
Example:

some arbitrary but known distribution $z \sim N(0, I)$

suppose $g_\theta(z) : z \longrightarrow X$, then

$\tilde{x} = g_\theta(z)$ has a different distribution than that of z & the distribution of $\tilde{x}$ depends on the function $g_\theta()$.

Suppose $g_\theta(z)$ is a nueral network



Denote the density of $\tilde{x} = g_\theta(z)$ using $P_\theta(\tilde{x})$

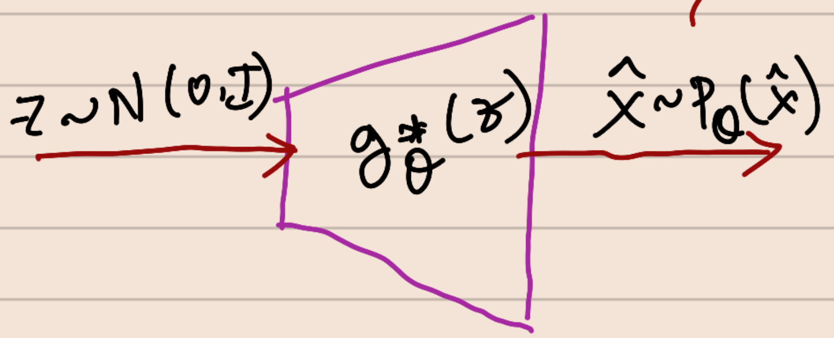
Suppose $D(P_x \| P_\theta)$ denote a divergence measure between P_x & P_θ,

Next step: solve the following optimization Problem

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \; D(P_x \| P_\theta)$$

Upon solving the above optimization Problem, the distribution P_x is implicitly estimated by $g_\theta(z)$ & one can sample from P_x using $g_\theta(z)$.

Push-forward Methods



A sample from $z \sim N(0, I)$, passed through $g_\theta(z)$ would produce a sample from $P^*_\theta(\hat{x})$, which is close to P_x, we end up sampling from P_x.

General principle followed by all Generative Models

The Questions:

1. How to compute the divergence metric without knowing P_θ & P_x.

2. What should be the choice of the divergence metric $\mathcal{D}$?

(3) How to choose the $g_\theta(z)$, in turn P_θ?

(4) How to solve the optimization problem of minimizing the divergence metric?

Variational Divergence Minimization

Define divergence metrics between distributions

f-divergence:

Given a probability distribution functions with the corresponding density functions denoted by P_x & P_θ, the f-divergence between them is defined as follows:

$$D_f(P_x \| P_\theta) = \int_{\mathcal{X}} P_\theta(x) f\left(\frac{P_x(x)}{P_\theta(x)}\right) dx.$$

$f(u): \mathbb{R}^+ \to \mathbb{R}$, convex, left-semi continuous

$f(1) = 0$

$\mathcal{X}$: space on which the P_x & P_θ are supported

Properties of f-Divergence:

(i) $D_f(\,) \geq 0$ for any choice of $f(\,)$

(ii) $D_f(P_x \| P_\theta) = 0$, if $P_x = P_\theta$

Examples of f-divergence:

(a) $f(u) = u\log u$: KL- Divergence.

$$D_f(P_x \| P_\theta) = \int_x \cancel{P_\theta(x)} \frac{P_x(x)}{\cancel{P_\theta(x)}} \cdot \log\left(\frac{P_x(x)}{P_\theta(x)}\right) \cdot dx$$

$$= D_{KL}$$

$$\underbrace{D_{KL}(P_x \| P_\theta)}_{\text{Forward KL}} \neq \underbrace{D_{KL}(P_\theta \| P_x)}_{\text{Reverse KL}}$$

KL Divergence is not symmetric.

Different choices of 'f' functions result in different divergence metrics with their own properties, which neccessitates the need to look at swathes of divergence metrics.

(b) $f(u) = \frac{1}{2}\left(u\log u - (u+1)\log\left(\frac{u+1}{2}\right)\right)$: JS-Divergence.

$\hookrightarrow$ Used famously in GANs

(c) $f(u) = \frac{1}{2}|u-1|$: Total Variational Distance.

Algorithm for F-Divergence Minimization.

Objective: Algorithm to minimize D_f b/w P_X & P_θ, without knowing both, but having samples from both.

Samples from P_X : dataset D

samples from P_θ : Outputs of $g_\theta(z)$ for different z.

Without knowing $P_\theta(x)$ & $P_X(x)$, solving a high-dimensional integral is infeasible.

Key Idea: Integrals involving density functi-ons can be approxi-mated using samples from the distribution.

Suppose we want to compute

$$I = \int_X h(x) \cdot P_X(x) \cdot dx$$

where $h(x)$ is a function & $P_X(x)$ the density fn.

We have samples drawn iid from P_X

$$x_1, x_2, \ldots, x_n \overset{iid}{\sim} P_X$$

$$I = \mathbb{E}_{P_X}[h(x)]$$ [From the Law of Unconscious Statistician]

Law of Large Numbers:

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} h(x_i) \approx \mathbb{E}_{P_X}[h(x)]$$

$$x_i \sim \text{iid } P_X$$

If the f-divergence metric can be expressed in terms of expectation of functions wrt P_X & P_θ, then one can compute & optimize them.

Expressing D_f in terms of Expectations over P_X & P_θ.

$$D_f(P_X \| P_\theta) = \int_x P_\theta(x) f\left(\frac{P_X(x)}{P_\theta(x)}\right) dx$$

$$\downarrow$$

Even though this expression looks eerily similar to the $\int P_\theta(x) h(x) \cdot dx$, you cannot directly rewrite that expression in terms of expectation precisely because of the arguments of f. They aren't simple "x" but instead are the ratio of prob density functions.

So, we somehow have to decouple the ratio of PDFs from f.

SLIGHT DETOUR

Conjugate function for a Convex fn:

If $f(u)$ is a convex function, then there exists a conjugate function

$$\underbrace{f^*(t) := \sup_{u \in \mathrm{dom}(f)}}_{\text{Point-wise defn}} \left\{ ut - f(u) \right\}.$$

$\hookrightarrow$ Basically, we are lowerbounding $f(u)$ at every point 't'.

Properties of Conjugate:

i) $f^*(t)$ is also convex.

(ii) $[f^*(t)]^* = f(u) \Rightarrow$

$$f(u) = \sup_{t \in \mathrm{dom}(f^*)} \left\{ tu - f^*(t) \right\}.$$

DETOUR DONE.

$$D_f(P_X \| P_\theta) = \int_X P_\theta(x) f\Big(\underbrace{\frac{P_X(x)}{P_\theta(x)}}_{u}\Big) \cdot dx$$

$$= \int_X P_\theta(x) f(u) \cdot dx.$$

$$f(u) = \sup_t \left\{ tu - f^*(t) \right\}$$

$$D_f(P_X \| P_\theta) = \int_X P_\theta(x) \cdot \sup_t \left(tu - f^*(t) \right) \cdot dx$$

$$= \int_X P_\theta(x) \sup_t \left\{ t \left[\frac{P_X(x)}{P_\theta(x)} \right] - f^*(t) \right\} \cdot dx$$

↪ cannot directly pull out the sup.!!

$$= \sup_{T(x) \in \mathcal{T}} \int_X P_\theta(x) \left\{ T(x) \frac{P_X(x)}{P_\theta(x)} - f^*(T(x)) \right\} \cdot dx$$

∵ The inner optimization problem involves x & the solution for it is dependent (a fn of x)

$\mathcal{T} : X \longrightarrow \mathrm{dom} f^*$.

$T(x) \in \mathcal{T}$

[Space of functions containing solutions for the inner optimization problem

But,

$$D_f \geq \sup_{T(x) \in \mathcal{T}} \int_X P_\theta(x) \left\{ T(x) \frac{P_X(x)}{P_\theta(x)} - f^*(T(x)) \right\} \cdot dx$$

because the space of functions $\mathcal{T}$, that we are optimizing over may not contain the optimal $T^*(x)$, that is the soln for the inner optimization problem.

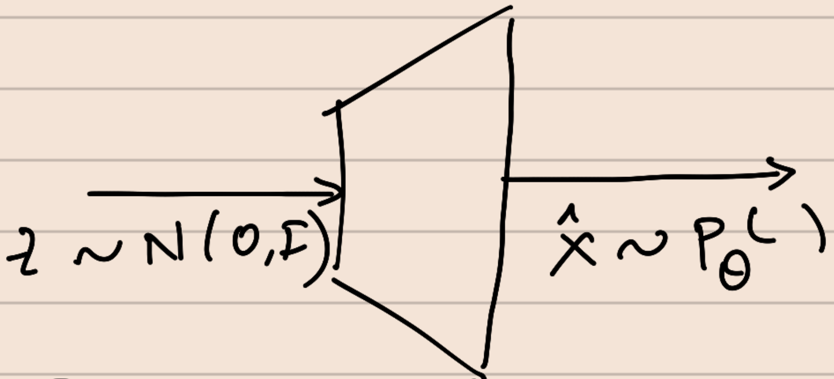
$$\geq \sup_{T(x)} \left[\int_X p_X(x)\, T(x) \cdot dx - \int_X p_\theta(x)\, f^*(T(x)) \cdot dx \right]$$

$$\geq \sup_{T(x)} \left[\mathbb{E}_{p_X} T(x) - \mathbb{E}_{p_\theta} f^*(T(x)) \right]$$

18/05/25, W2L5

Realization of VDM

Given Data $D = \{x_1, x_2, \ldots, x_n\} \overset{iid}{\sim} P_X$



Goal:

$$\theta^* = \arg\min_\theta D_f(P_X \| P_\theta)$$

$$D_f \geq \sup_{T(x) \in \mathcal{T}} \left[\mathbb{E}_{P_X} T(x) - \mathbb{E}_{P_\theta} f^*(T(x)) \right]$$

We set out to solve — $\theta^* = \arg\min_\theta D_f(P_X \| P_\theta)$

but we are delegated to solve $\theta^* \approx \arg\min_\theta \{\text{lower bound on } D_f\}$ and that's the best we can do.

$$= \arg\min_\theta \left[\sup_{T(x)} \left(\mathbb{E}_{P_X} T(x) - \mathbb{E}_{P_\theta} f^*(T(x)) \right) \right]$$

w.r.t the parameters of $g_\theta(z)$ → w.r.t a class of functions $T(x) \in \mathcal{T}$

This cannot be done analy-tically?

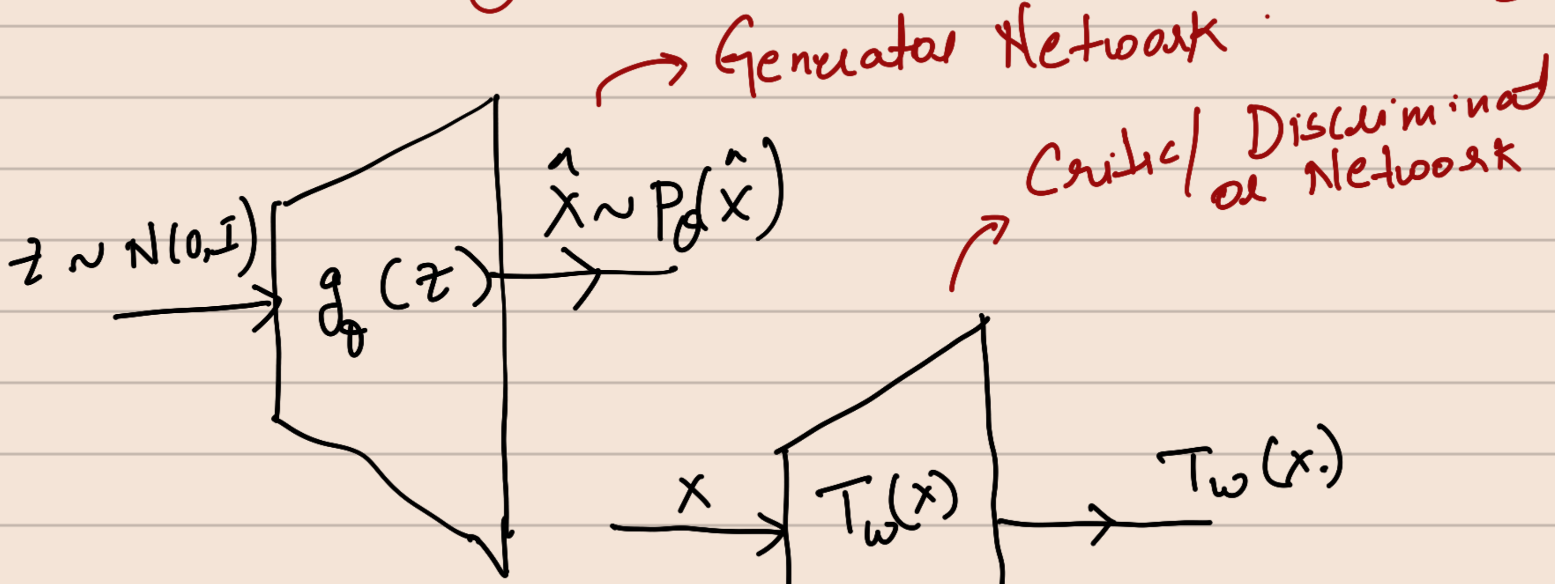
optimizing over space of non-trivial functions.

In practice, we represent $\mathcal{T}$ via neural networks: $T_w(x)$, where 'w' are the parameters of the neural network.

With this, the objective will become:

$$\theta^*, w^* = \arg\min_{\theta} \max_{w} \left[\mathbb{E}_{P_x} T_w(x) - \mathbb{E}_{P_\theta} f^*(T_w(X)) \right]$$

Implementing VDM for Generative Sampling.



$$J(\theta, w) = \left[E_{P_x} T_w(x) - E_{P_\theta} f^*(T_w(x)) \right]$$

$$\theta^*, \omega^* = \arg \min_{\theta} \max_{\omega} J(\theta, \omega)$$

$\hookrightarrow$ saddle-point optimization

Problem where we have alternate minimization & maximization.

$\underbrace{\hspace{10em}}$

Blueprint for adversial training

Any saddle-point optimization problem is also a adversial optimization problem.

19 May 2025, W2L6:

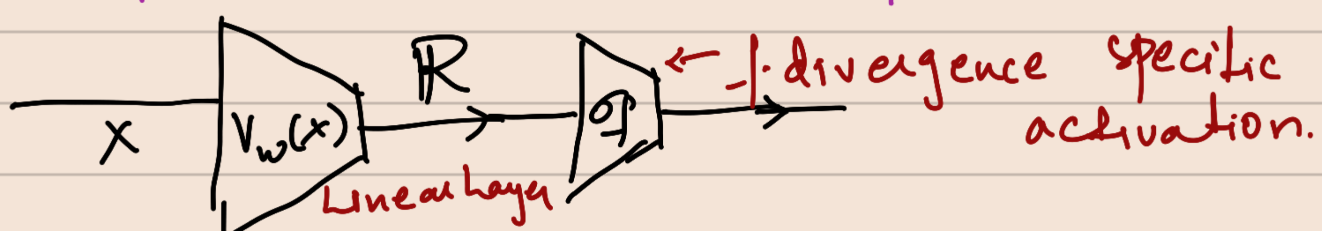
We know, by construction, $T(\,): \mathcal{X} \longrightarrow \text{dom} f^*$

In practice, we do this by: $T_\omega(x)$ is represented as: $\sigma_f(V_\omega(x))$, $\leftarrow$ Depending on the choice of f^*, we need to tweak the 'T' network so that the range of T network corresponds to domain of f^*.

where σ_f is a f-divergence specific activation

$$V_\omega(x): \mathcal{X} \longrightarrow \mathbb{R}, \quad \sigma_f(v): \mathbb{R} \rightarrow \text{dom} f^*$$

This means the T network that we'll approximate using a neural network is represented as a composition of 2 f^ns.



$$J(\theta,\omega) = \mathbb{E}_{P_x}\left[\sigma_f(V_\omega(x))\right] - \mathbb{E}_{P_\theta}\left[f^*(\sigma_f(V_\omega(x)))\right]$$

Generative Adversial Networks

A special case of VDM algorithms:

The choice of f-divergence: $\underbrace{u\log u-(u+1)\log(u+1)}_{\text{Awfully similar to JS Div.}}$

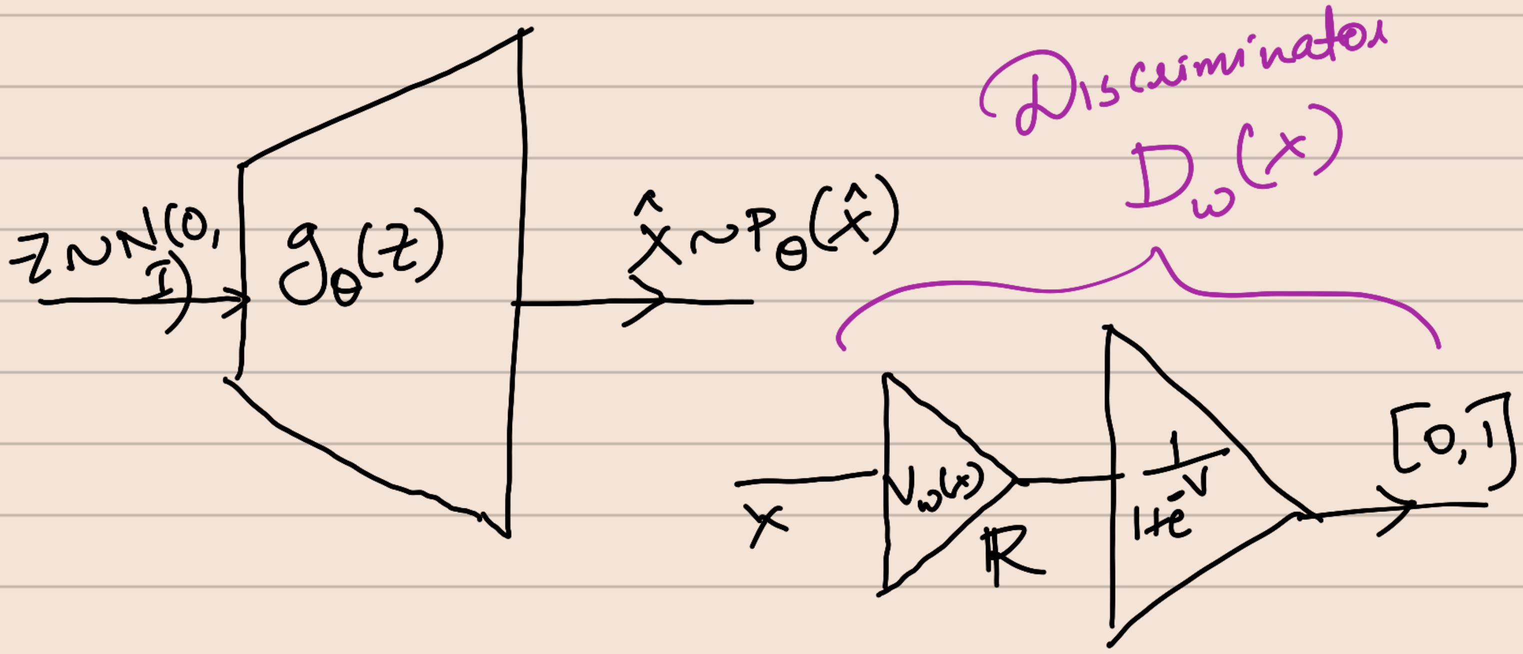
$$f^*(t) = -\log(1-e^t),$$

$$\operatorname{dom} f^* = \mathbb{R}^-$$

$$\sigma_f(v) = -\log(1+e^{-v})$$

$$J_{Gan}(\theta,\omega) = \mathbb{E}_{P_x}\left[\log D_\omega(x)\right] + \mathbb{E}_{P_\theta}\left[\log(1-D_\omega(x))\right]$$

where $D_\omega(x) = \dfrac{1}{1+e^{-V_\omega(x)}}$



$$J_{GAN}(\theta, \omega) = \underset{P_X}{E}\, T_\omega(x) - E_{P_\theta} f^*(T_\omega(x))$$

$$= E_{P_X}\, g_f(V_\omega(x)) - E_{P_\theta} f^*(g_f(V_\omega(x)))$$

$$= E_{P_X} -\log(1+e^{-V_\omega(x)}) - E_{P_\theta} f^*(-\log(1+e^{-V_\omega(x)}))$$

$$= E_{P_X} \log\left(\frac{1}{1+e^{-V_\omega x}}\right) - E_{P_\theta} f^*\left(\log\left(\frac{1}{1+e^{-V_\omega x}}\right)\right)$$

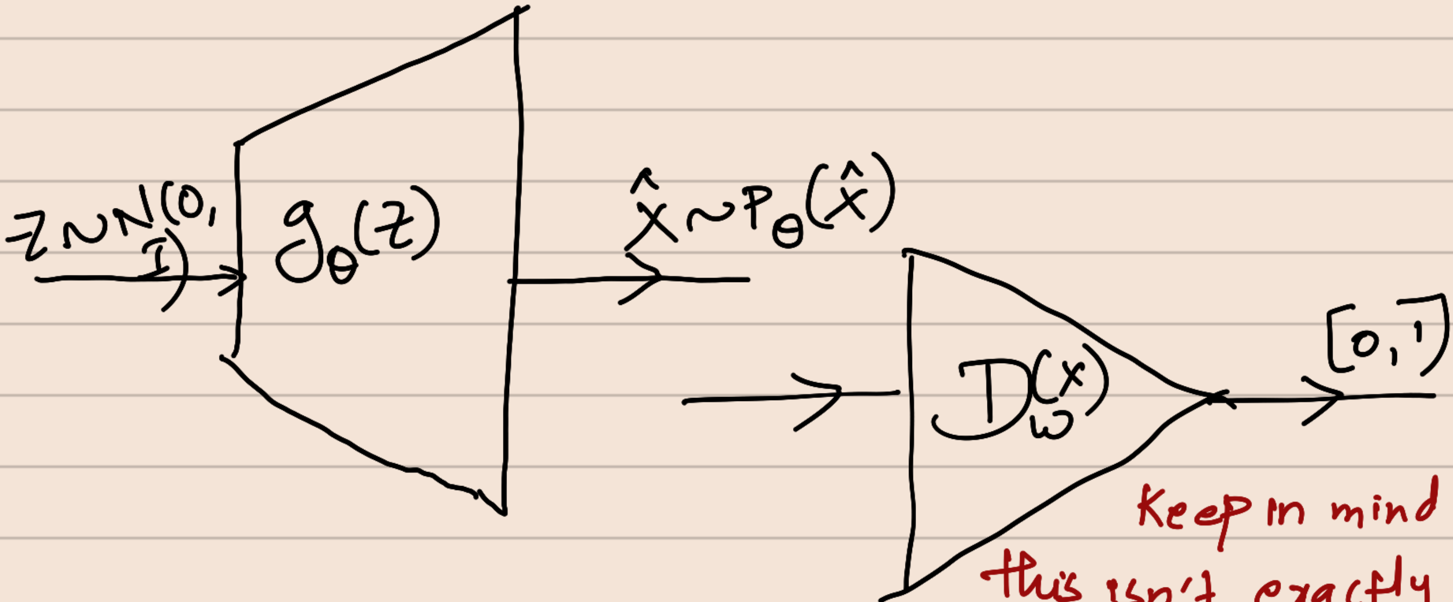
$$= E_{P_X} \log\left(\frac{1}{1+e^{-V_\omega(x)}}\right) + E_{P_\theta} \log\left(1+e^{-\log\left(\frac{1}{1+e^{-V_\omega x}}\right)}\right)$$

$$= E_{P_X} \log\left(\frac{1}{1+e^{-V_\omega x}}\right) + E_{P_\theta} \log\left(1 - \frac{1}{1+e^{-V_\omega x}}\right)$$

$$= E_{P_X} \log D_\omega(x) + E_{P_\theta} \log(1 - D_\omega(x))$$

Derivation of J_{GAN} for $f := u\log u - (u+1)\log(u+1)$

GAN ARCHITECTURE



Keep in mind this isn't exactly the T function but a deterministic composition of the T Network.

$$J_{GAN}(\theta, w) = \mathbb{E}_{x \sim p_x} \log D_w(x) + \mathbb{E}_{\hat{x} \sim p_\theta} \log(1 - D_w(\hat{x}))$$

21 May 2025

IMPLEMENTATION OF GAN IN PRACTICE.

Input: $D = \{x_1, x_2, \ldots, x_n\} \overset{iid}{\sim} p_x$

$$w^* = \arg\max_w \left[\mathbb{E}_{p_x}\left(\log D_w(x)\right) + \mathbb{E}_{p_\theta}\left(\log(1 - D_w(\hat{x}))\right) \right]$$

$$\approx \arg\max_{w} \left[\frac{1}{B_1} \sum_{j=1}^{B_1} \log D_w(x_i) + \frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(\hat{x}_j)) \right]$$

$$x_1 \dots, x_{B_1} \sim P_x$$
$$\hat{x}_1, \dots, \hat{x}_{B_2} \sim P_\theta$$

$$w^{t+1} \longleftarrow w^t + \alpha_1 \nabla J_{GAN}(\theta, w)$$: One gradient step thru Discriminator. θ is kept a constant.

$$\theta^* = \arg\min_{\theta} J_{GAN}(\theta, w)$$

Independent of θ.

$$\approx \arg\min_{\theta} \left[\frac{1}{B_1} \sum_{j=1}^{B_1} \log D_w(x_i) + \frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(\hat{x}_j)) \right]$$

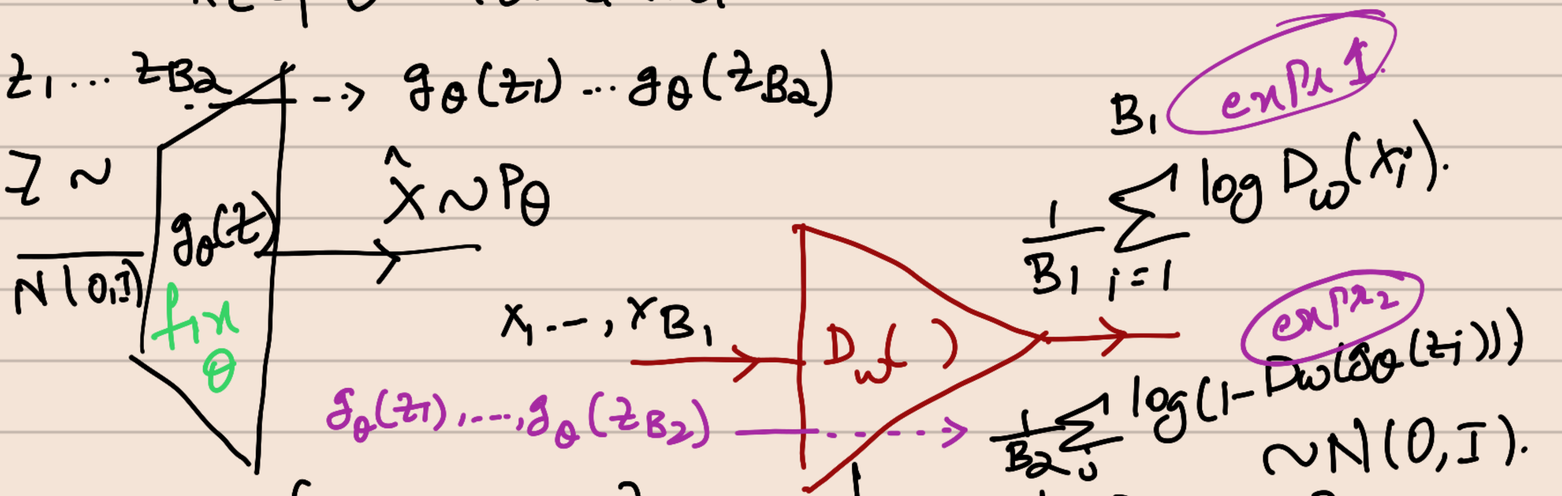
$$\approx \arg\min_{\theta} \left[\frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j))) \right]$$

$$\theta^{t+1} \longleftarrow \theta^t - \alpha_2 \nabla J_{GAN}(\theta, w)$$: 1 GD step thru the generator.

w is kept a constant

To Train the Discriminator:

keep θ constant



$B_1 = \{x_1, \dots, x_{B_1}\} \subset D$ | sample $z_1, \dots, z_{B_2}$
Pass $z_1, \dots, z_{B_2}$ thru $g_\theta(\ )$ with fixed θ.

$$J_{GAN} = \frac{1}{B_1}\sum_{j=1}^{B_1} \log D_w(x_i) + \frac{1}{B_2}\sum_{j=1}^{B_2} \log(1 - D_w(\underbrace{\hat{x}_j}_{g_\theta(z_j)}))$$

Once we have exp1 & exp2, let us calculate the loss and backpropagate the gradients all the way to the inputs of the discriminator.

If $P_\theta(x) = P_x(x)$ | $\mathop{E}_{P_X} \log(D(x)) + \mathop{E}_{P_\theta} \log(1 - D(x)) = J_{GAN}$.

The optimization over $D(x)$ is treated independently for each 'x', allowing us to find optimal $D^*(x)$ by maximizing point-wise loss.

$$f(D(x)) = P_x(x)\log(D(x)) + P_\theta(x)\log(1-D(x))$$

$$\frac{df(D(x))}{dD(x)} = \frac{P_x(x)}{D(x)} + \frac{P_\theta(x)}{1-D(x)}(-1) = 0$$

$$\frac{P_x(x)}{D(x)} = \frac{P_\theta(x)}{1-D(x)}$$

&

$$1-D(x) = D(x)$$

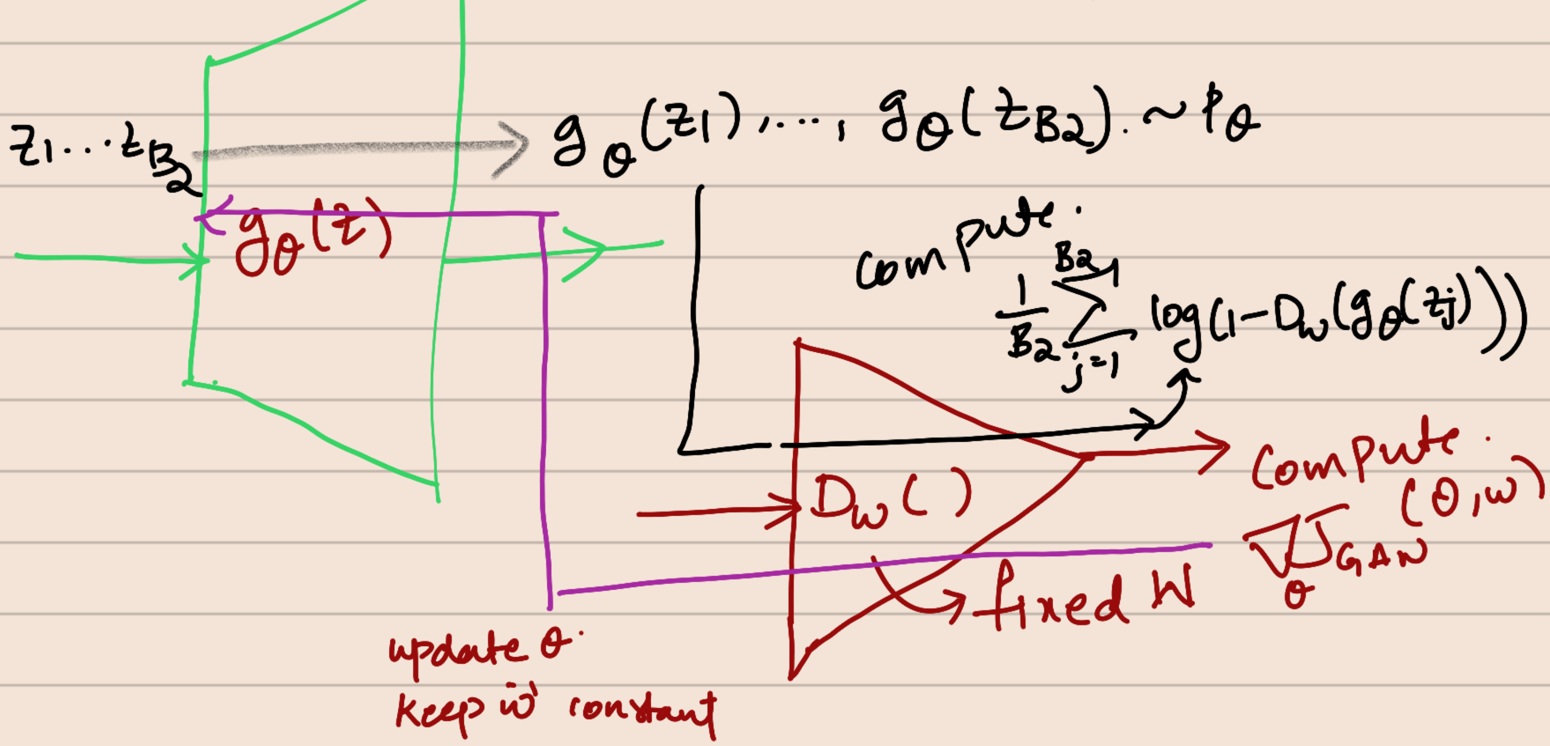
$$2D(x) = 1$$

$$D(x) = 1/2.$$

Think more deeply about why optimization over $D(x)$ is treated independently for each x.

TO TRAIN THE GENERATOR:

sample $z_1 \dots z_{B_2} \sim N(0, I)$



$$\left[\frac{1}{B_2}\sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j))) \right] = J_{GAN}.$$

To update the params of generator, we donot need the 1st term (we donot need the samples from the data).

$$\theta^{t+1} \leftarrow \theta^t - \alpha_2 \nabla_\theta J_{GAN}(\theta, w)$$

Typically,
Alternate between 1 step of Generator and 1 step of discriminator whilst training

Stopping Criterion?? When the quality metrics of the Generator are reached.

—— Training VDM or GANs done ——

Next we'll look at inference of GAN, 'improvisations [different f-functions' result in different kind of VDMs]

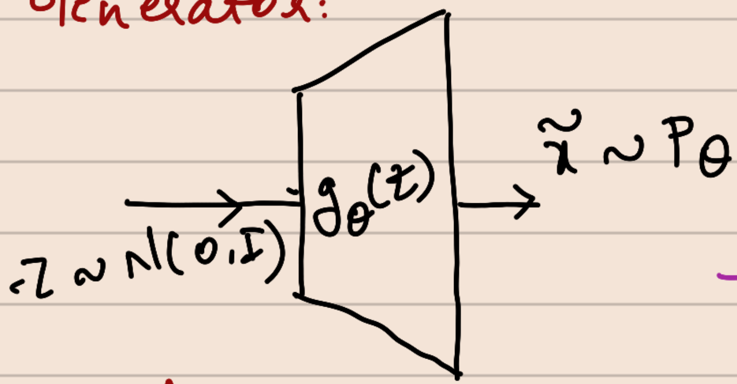
17:53, 25 May, Sun

Vanilla GAN:

Input: $D = \{x_1, x_2, \dots, x_n\} \overset{iid}{\sim} p_x$

Data: MNIST $_{28 \times 28}$

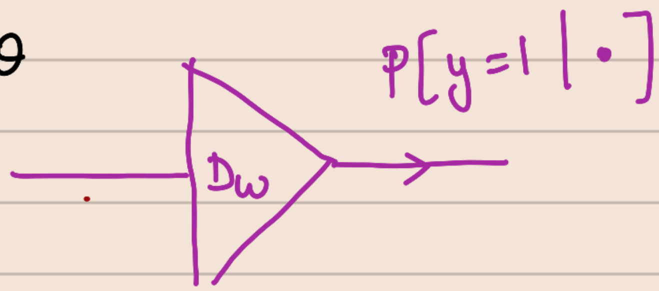
Generator:



Decide on z-dim: 100

Output: 784, then reshape to 28×28

Discriminator



Input: 784

Output: $\mathbb{P}[y=1 | \cdot]$

Loss function:

$$J(\theta, w) = \mathbb{E}_{P_X}\left[\log D_w(x)\right] + \mathbb{E}_{P_\theta}\left[1 - \log D_w(x)\right]$$

Approximate expectation using sample mean (WGAN) and use Batches.

$$J(\theta, w) = \frac{1}{B_1}\sum_{i=1}^{B_1} \log D_w(x_i) + \frac{1}{B_2}\sum_{j=1}^{B_2} \log(1 - D_w(\hat{x}_j))$$

$$x_1, x_2, x_3, \ldots, x_{B_1} \overset{iid}{\sim} P_X$$
$$\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{B_2} \overset{iid}{\sim} P_\theta$$

$$J(\theta, w) = \frac{1}{B_1}\sum_{i=1}^{B_1} \log D_w(x_i) + \frac{1}{B_2}\sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j)))$$

$$z_1, z_2, \ldots, z_{B_2} \overset{iid}{\sim} N(0, I)$$

Discriminator N/W:

$$w^* = \underset{w}{\arg\max} \frac{1}{B_1}\sum_{i=1}^{B_1} \log(D_w(x_i)) + \frac{1}{B_2}\sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j)))$$

$$w^{t+1} \leftarrow w^t + \alpha_1 \nabla_w J(\theta, w)$$

while training the discriminator the parameters of the generator network are kept constant.

*

Sample $z_1, z_2, \dots, z_{B_2} \overset{iid}{\sim} N(0, I)$
pass thru $g_\theta(z)$

$g_\theta(z_1), g_\theta(z_2), \dots, g_\theta(z_{B_2})$

Pass these thought $D_w(\cdot)$ & obtain 2nd term in loss.

*

$x_1, x_2, \dots, x_{B_1} \sim P_x$

Pass through $D_w(\cdot)$ & get the 1st term.

Total loss = term 1 + term 2

This is how the gradient ascent is implemented for the discriminator network.

Generator N/w:

$$\theta^* = \arg\min_\theta J(\theta, w)$$

$$= \arg\min_\theta \frac{1}{B_1} \sum_{i=1}^{B_1} \log(D_w(x_i)) + \frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j)))$$

Independent of θ.

$$\theta^{t+1} \longleftarrow \theta^t - \alpha_2 \nabla_\theta J(\theta, w)$$

while training the generator keep the params of the discriminator constant.