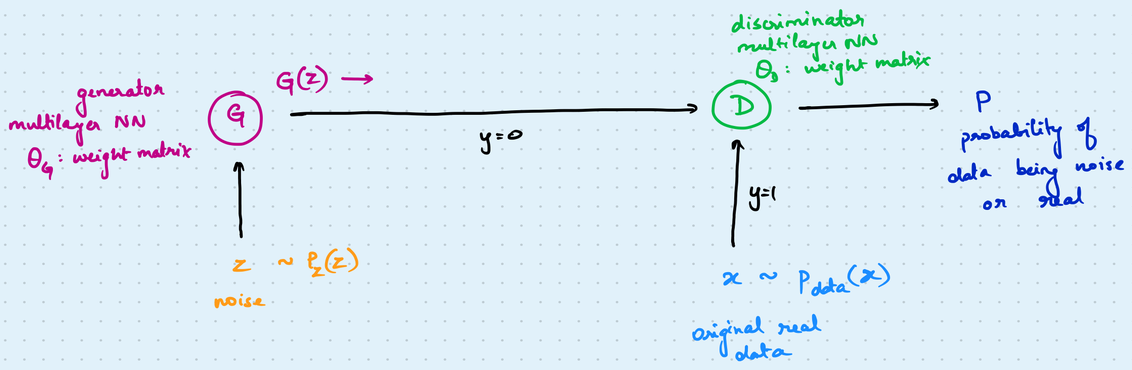
Generative Adversarial Networks

GAN
- Generator: tries to make noise look like meaningful data
- Discriminator: tries to distinguish between real inputs and noise

(adversarial inputs)



Binary cross entropy:

$$L = -\Sigma\left[y \log \hat{y} + (1-y)\log(1-\hat{y})\right]$$

real data: $y=1$, $\hat{y} = D(x)$ $\Rightarrow L = \ln(D(x))$

noise: $y=0$, $\hat{y} = D(G(z))$ $\Rightarrow L = \ln(1-D(G(z)))$

Thus, for any given sample:

$$L = \ln(D(x)) + \ln(1-D(G(z)))$$

For all samples, we take expectation

$$E(L) = E(\ln D(x)) + E(\ln(1-D(G(z)))) = \int P_{data}(x)\cdot \ln(D(x))\,dx + \int P_z(z)\cdot \ln(1-D(G(z)))\,dz$$

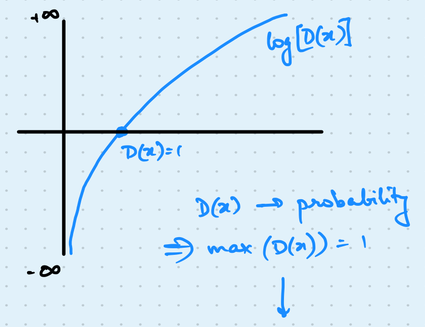
$$V(G,D) = \underset{x\sim p_{data}}{E}\left(\ln(D(x))\right) + \underset{z\sim p_z}{E}\left(1-D(G(z))\right) \quad \text{— (1)}$$

The Discriminator

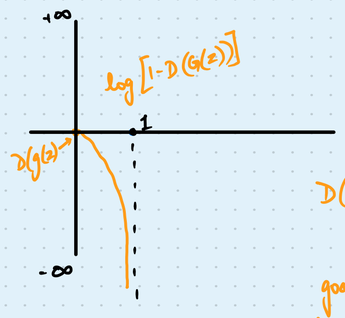
Objective: classify real vs. fake data

$D(x)$ = probability of a label to be associated w/ real data x → (associated w/) → $\ln(D(x))$

$D(G(z))$ = " " " " " " " fake data z → (associated w/) → $\ln(1-D(G(z)))$



D(x) → probability
$\Rightarrow \max(D(x)) = 1$
↓
Discriminator's goal for real data



D(G(z)) = 0
↓
goal of discriminator for fake data

Finally we have the goal of the discriminator:

$$D = \max\left\{\ln(D(x)) + \ln(1-D(G(z)))\right\} \quad \text{— (2)}$$

The Generator

Objective: fool the discriminator

Make $G(z)$ such that $D(G(z))$ becomes 1!

Discriminator: I'm going to be so perfect @ differentiating real & fake
real gets 1.
fake gets 0.

Generator: haha I'm going to fool the discriminator.
I'm going to make G(z).
When I pass to discriminator, D(G(z)) = 1.
She'll think it's real data!

Now, the generator will achieve the target at $-\infty$

From the graph, we see that we need to minimise $\ln(1-D(G(z)))$. Thus:

$$G = \min\left\{\ln(D(x)) + \ln(1-D(G(z)))\right\} \quad \text{— (3)}$$

- $\ln(D(x))$ plays no role here
- $\ln(1-D(G(z)))$: we try to get $D(G(z)) \approx 1$ $\Rightarrow$ produce sample close to real data

Putting Everything Together

From (2), (3):

$$\min_G \max_D \left\{\ln[D(x)] + \ln[1-D(G(z))]\right\}$$ } single sample

From (1):

$$V(G,D) = \min_G \max_D \left\{\underset{x\sim P_{data}}{E}\left[\ln(D(x))\right] + \underset{z\sim P_z}{E}\left[1-D(G(z))\right]\right\}$$

Standardise variables

$$V(D,G) = \min_G \max_D \left\{\underset{x\sim P_{data}}{E}\left[\ln(D(x))\right] + \underset{z\sim P_z}{E}\left[\ln(1-D(G(z)))\right]\right\}$$

$$= \int P_{data}(x)\cdot \ln[D(x)]\,dx + \int P_z(z)\cdot \ln(1-D(G(z)))\,dz \quad \text{— (1)}$$

Say we have random var X with distri $P_x(x)$,
and $Y = G(x)$

$\Rightarrow$ PDF of $Y = P_y(y) = P_x[G^{-1}(y)] \cdot \frac{d}{dy}[G^{-1}(y)]$

- $P_x(x)$
- rate of change of x wrt y

Using the above result, we see that here:

$z \xrightarrow[P_z(z)]{} [G] \longrightarrow x = G(z)$

because we are trying to make a generator that makes data as close to input as possible

$$P_g(x) = P_z[G^{-1}(x)] \cdot \frac{d}{dx}[G^{-1}(x)] \quad \text{— (2)}$$

Replacing $z = G^{-1}(x)$ in (1):

$$\int_x P_z[G^{-1}(x)] \cdot \log(1-D(G(G^{-1}(x)))) \, d[G^{-1}(x)] = \int P_z[G^{-1}(x)] \cdot \log(1-D(x))\, d[G^{-1}(x)]$$

From (2), we have

$$P_z[G^{-1}(x)] \cdot d[G^{-1}(x)] = P_g(x)\cdot dx$$

Substituting, we get:

$$\int P_g(x)\cdot \log(1-D(x))\cdot dx$$

Optimal Discriminator

We now have:

$$V(D,G) = \int P_{data}(x)\log[D(x)]\,dx + \int P_g(x)\cdot\log(1-D(x))\,dx$$

(1) To find optimal D_G^*, fix G and differentiate wrt $D(x)$, set to zero

$$\frac{d}{d[D(x)]}\left[P_{data}(x)\log[D(x)] + P_g(x)\log(1-D(x))\right] = 0$$

$$\frac{P_{data}(x)}{D(x)} - \frac{P_g(x)}{1-D(x)} = 0 \Longrightarrow P_{data}(x) - D(x)P_{data}(x) - D(x)P_g(x) = 0$$

$$P_{data} = D(x)\cdot P_{data}(x) + D(x)\cdot P_g(x)$$

$$D_G^*(x) = \frac{P_{data}(x)}{P_{data}(x)+P_g(x)}$$

(2) Remember we need to maximise D. Use second derivative

$$-\frac{P_{data}(x)}{[D(x)]^2} - \frac{P_g(x)}{[1-D(x)]^2} < 0$$

→ concave function: critical point is a maximum

$P_{data}(x), P_g(x) \longrightarrow >=0$ [probability]

$D(x), 1-D(x) \longrightarrow >=0$ [D(x) = probability]

$\Rightarrow$ with -ve sign, all terms become -ve

Optimal generator

We have found optimal discriminator D_G^* for some fixed G. Now, we need to find the optimal generator G^* after substituting D_G^*

$$\Rightarrow G^* = \arg\min_G\left\{V(D_G^*, G)\right\}$$

$$= \arg\min_G\left\{\int\left[P_{data}(x)\log[D_G^*(x)] + P_g(x)\log[1-D_G^*(x)]\right]dx\right\}$$

$$= \arg\min_G\left\{\int\left[P_{data}(x)\log\left[\frac{P_{data}(x)}{P_{data}(x)+P_g(x)}\right] + P_g(x)\log\left[\frac{P_g(x)}{P_{data}(x)+P_g(x)}\right]\right]dx\right\}$$

Note: Jenson-Shannon Divergence

$$JS(P_1\|P_2) = \frac{1}{2}\underset{x\sim P_1}{E}\left[\ln\left(\frac{P_1}{P_1+P_2/2}\right)\right] + \frac{1}{2}\underset{x\sim P_2}{E}\left[\ln\left(\frac{P_2}{P_1+P_2/2}\right)\right]$$

The previous integral simplifies to:

$$G^* = \arg\min_G\left\{-\log 4 + 2\cdot JS(P_{data}\|P_g)\right\}$$

$$G^* = \arg\min_G\left\{JS(P_{data}\|P_g)\right\}$$ } ignoring constants

Reasoning:

- JS divergence = 0 if two distributions exactly the same
- If $P_{data} = P_g$, then generator G is perfectly replicating data
- Thus you find the value of G that minimises JS divergence $\longrightarrow G^*$

Note: Nash equilibrium

If $P_g = P_{data}$:

$D = \frac{1}{2}$ $\forall x$

$\Rightarrow$ generator creates perfect fakes, discriminator cannot tell the difference

Note: GAN Training Tips

- Different learning rates for D, G
- Sometimes train D more than G
- Soft labels (0.9 instead of 1.0)
  ↳ prevent D from being overconfident
- Match feature statistics between real and fake data

	VAE	GAN
Training Stability	More stable	Can be unstable
Sample Quality	Often blurry	Often sharper
Latent Space	Well-structured	Less structured
Mode Coverage	Better coverage	Mode collapse risk
Theoretical Foundation	Strong (variational inference)	Game theory based
Computational Cost	Moderate	Higher (two networks)

- **Use VAE:** Need structured latent space, want stability, care about mode coverage
- **Use GAN:** Need high-quality samples, have computational resources, can handle training complexity