PART 4

Deep Unsupervised Learing

- So far, **supervised learning**
  - Discriminative methods:

$$\{\mathbf{x}_i, y_i\}_{i=1}^N \rightarrow p(y|\mathbf{x})$$

  - Generative methods:

$$\{\mathbf{x}_i, y_i\}_{i=1}^N \rightarrow P(y), p(\mathbf{x}|y)$$

logistic regression
Softmax regression
FNN
CNN

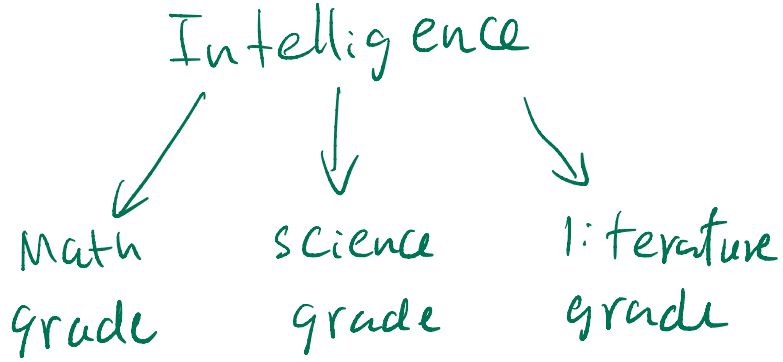
class sizes

class conditional

* Latent / Hidden Variables:

* Observed Variables:

* Latent variable models:



Market Sentiment

unlabelled data

Gaussian Mixture Model

cluster sizes

cluster characteristics

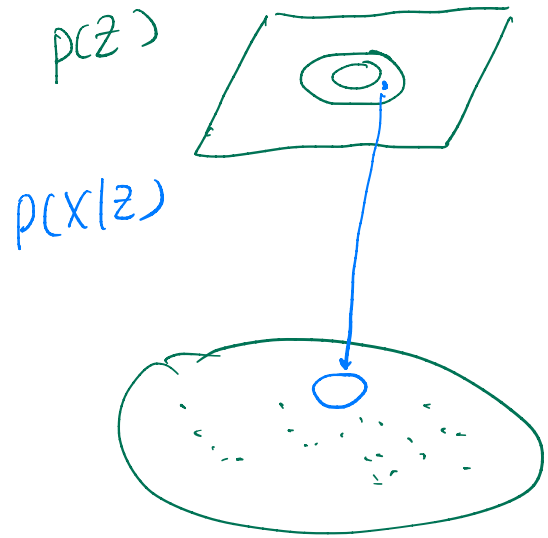
- Next, **unsupervised learning**:
  - *Finite mixture models* for clustering [Skipped]

    $\{\mathbf{x}_i\}_{i=1}^N \to P(z), p(\mathbf{x}|z)$
  - *Varitional autoencoder* for data generation and representation learning

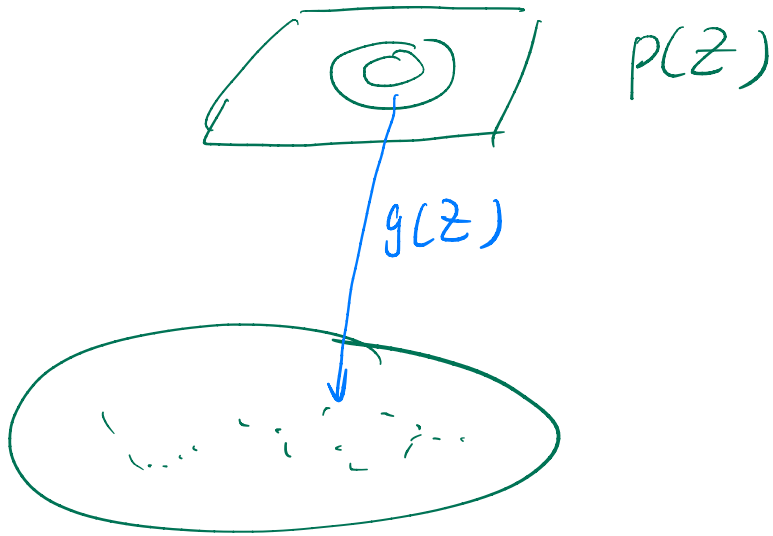
    $\{\mathbf{x}_i\}_{i=1}^N, p(\mathbf{z}) \to p(\mathbf{x}|\mathbf{z})$ $q(\mathbf{z}|\mathbf{x})$ used in inference
  - *Generative adversarial networks* for data generation

    $\{\mathbf{x}_i\}_{i=1}^N, p(\mathbf{z}) \to \mathbf{x} = g(\mathbf{z})$

VAE



GAN



L01-2 : Page 18

Unsupervised Learning

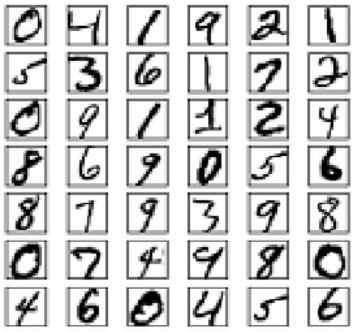
- Unknown true distribution $P(\mathbf{x})$.

$$P(\mathbf{x}) \xrightarrow{\text{sampling}} \mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N} \xrightarrow{\text{learning}} Q(\mathbf{x})$$

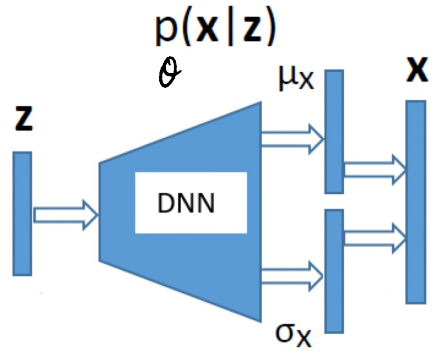
- Objective:
  - **Minimizing KL**: $KL(P||Q)$
  - Same as **minimizing cross entropy**: $H(P, Q)$
  - Approximating the cross entropy using data:

$$\begin{aligned} H(P, Q) &= -\int P(\mathbf{x}) \log Q(\mathbf{x}) d\mathbf{x} \\ &\approx -\frac{1}{N} \sum_{i=1}^{N} \log Q(\mathbf{x}_i) \\ &= -\frac{1}{N} \log Q(\mathcal{D}) \end{aligned}$$

  - Same as **maximizing likelihood**: $\log Q(\mathcal{D})$.



$\cdot\ p(z) \Rightarrow$



X

$$\log p_\theta(x) = \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x^i)$$

$$p(z),\ p(x|z) \Rightarrow p(x) = \int p(z)\, p(x|z)\, dz$$

$$\log p_\theta(x^i) = \int p(z)\, p(x^i|z)\, dz \approx \frac{1}{L} \sum_{j=1}^{L} p(x^i|z^j)$$

$$z^1 \ldots z^L \sim p(z)$$

$x^i = (x_{i1}, x_{i2}, \cdots x_{iD})$

$D = 256 \times 256$

$= 65,536$

$$P_\theta(x^i | z^l) = C \prod_{j=1}^{D} e^{-\frac{(x_{ij} - \mu_{ij}^l)^2}{2\sigma^2}}$$

$$= C \times \underbrace{0.99 \times 0.99 \times \cdots \times 0.99}_{65K} \approx 0$$

Analogy:

X : Image of person flying small airplane

z : income

$z^\ell \sim p(z)$

$p(x|z)$ { negligible ✓ / Not small

$x \longrightarrow q(z|x)$

Distribution of income among those flying small airplane

$z^\ell \sim q(z|x)$

$p(x|z^\ell)$ { Negligible / not small ✓

Want: $z^l \sim p(z)$ $\quad \frac{1}{L} \sum_{l=1}^{L} p(x^i \mid z^l)$

Do: $z^l \sim q(z \mid x)$ $\quad \frac{1}{L} \sum_{l=1}^{L} p(x^i \mid z^l)$

Bias

$$E_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right] = E_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right]$$

$$= \int q(z|x) \log \frac{p(x,z)}{q(z|x)} dz = E_{q(z|x)}\left[\log \frac{p(z)p(x|z)}{q(z|x)}\right]$$

$$\leq \log \int q(z|x) \frac{p(x,z)}{q(z|x)} dz = E_{q(z|x)}[\log p(x|z)]$$

Jensen's inequality

$$- E_{q(z|x)}\left[\log \frac{q(z|x)}{p(z)}\right]$$

$$KL(q(z|x) \,||\, p(z))$$

$$= \log p(x)$$

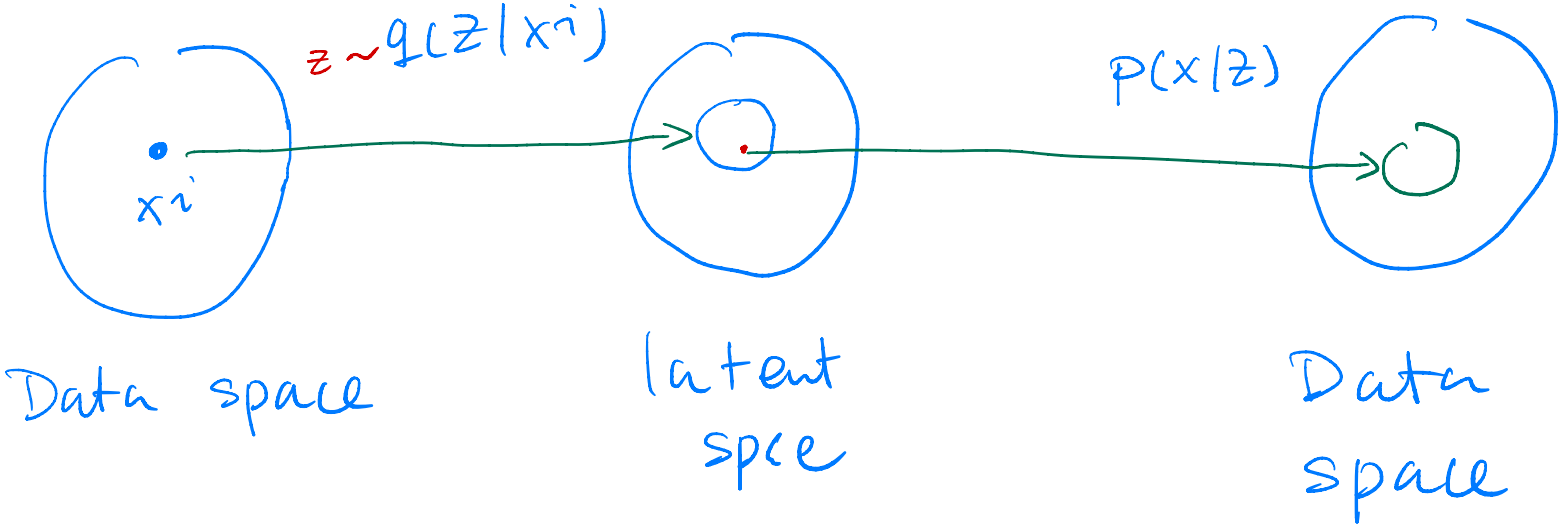
$$\log p(x) \geq E_{q(z|x)}[\log p(x|z)] - KL(q(z|x)|p(z))$$

Variational lower bound

Evidence lower Bound (ELBO)

reconstruction error

$$\mathcal{L}(\mathbf{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}) = E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] - \mathcal{D}_{KL}[q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})]$$



* How well p(x|z) match x^i : $\log p(x^i|z)$

* How well x^i is reconstructed from $q(z|x^i)$.

Representation learning

NLP: BERT, Word2Vec, ... ELMo

Images: AE

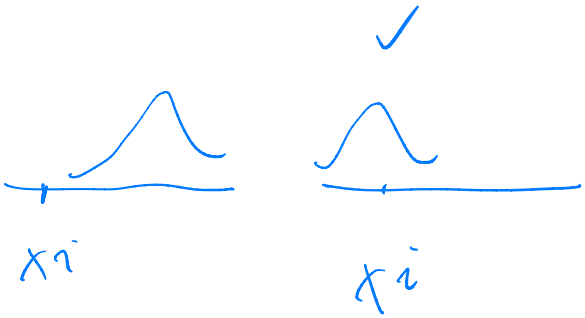
DAE

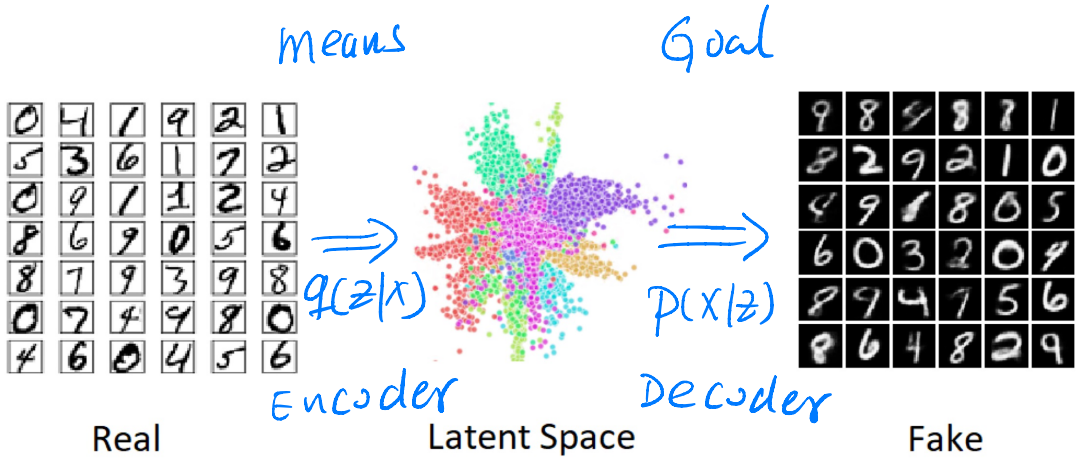
VAE

Contrastive learning = SimCLR

Self-Supervised learning : DINO

VAE





$$\mathcal{L}(\mathbf{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}) = E_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] - \mathcal{D}_{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})]$$

How to Compute (1) (2) ?

Depends on ϕ — Depends on ϕ? No

$$\mathcal{L}_1 = E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] \approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$$

where $\mathbf{z}^{(i,l)} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$.

Reparameterization : scalar x, z

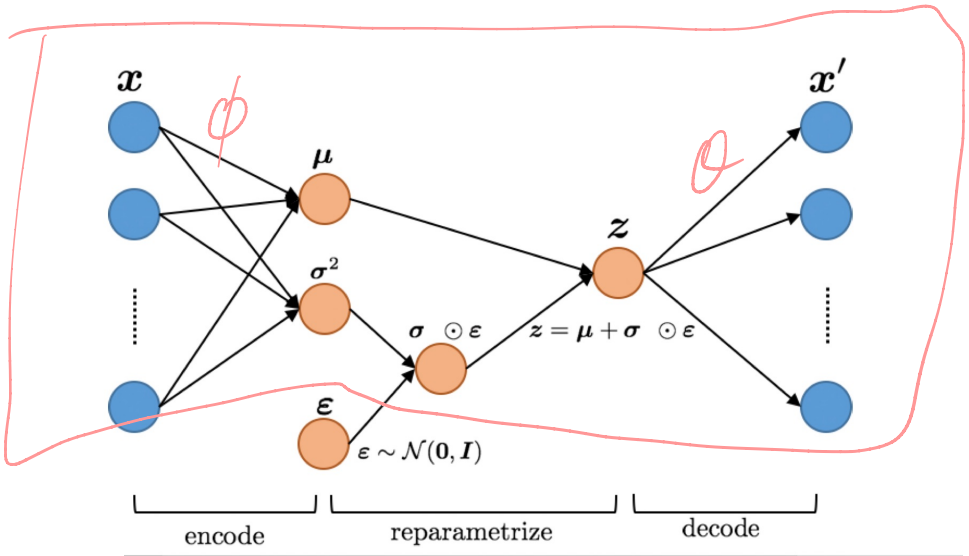
$$q(z|x) = \mathcal{N}(\underbrace{x^2+2}_{\mu}, \underbrace{x^4}_{\sigma^2}) = \frac{1}{\sqrt{2\pi}x} e^{-\frac{(z-(x^2+2))^2}{2x^4}}$$

$$\frac{dz}{dx} = ? \qquad \frac{dq}{dx} = \text{..ok}$$

$$z = \underbrace{x^2+2}_{\mu} + \underbrace{x^2}_{\sigma}\,\varepsilon \qquad \varepsilon \sim N(0,1)$$

$$\frac{dz}{dx} = 2x + 2x\varepsilon$$

$$\mathbf{z} = \mu_z(\mathbf{x}, \phi) + \sigma_z(\mathbf{x}, \phi) \odot \epsilon, \ \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



$$\mathcal{L}_1 = E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

$$= \frac{1}{L}\sum_{l=1}^{L} \log P_\theta(x^i \mid z^{(i,l)})$$

θ ϕ

$\varepsilon^l \sim N(0, I)$

$l = 1, \dots L$

$z^{(i,l)} = \mu_z(x^i, \phi) + \sigma_z(x^i, \phi) \odot \varepsilon^l$

$$\mathcal{L}_2 = -\mathcal{D}_{KL}[q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})].$$

max $1 + \log \sigma^2 - \sigma^2$

$$= \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right)$$

$q_\phi(z|x^i)$ $\qquad$ $p_\theta(z)$

mean $\begin{bmatrix} \mu_1^i \\ \vdots \\ \mu_J^i \end{bmatrix}$ $\xrightarrow{\max L_2}$ $\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

Σ $\begin{bmatrix} \sigma_1^i & & 0 \\ & \ddots & \\ 0 & & \sigma_J^i \end{bmatrix}$ $\xrightarrow{\max L_2}$ $\begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$

L11: GAN

Train D

Real: $\{x^i\}_{i=1}^m$ $\quad Y_i = 1$

Fake: $\{g(z^i)\}_{i=1}^m$ $\quad Y_i = 0$

Binary Xentropy

$$\begin{cases} -\log P(Y_i=1 \mid x^i) \\ = -\log D(x^i) \end{cases}$$

$$\begin{cases} -\log P(Y_i=0 \mid g(z^i)) \\ = -\log [1 - P(Y_i=1 \mid g(z^i))] \\ = -\log [1 - D(g(z^i))] \end{cases}$$

D minimize:

$$J = -\frac{1}{m}\sum_{i=1}^{m} \log D(x^i) - \frac{1}{m}\sum_{i=1}^{m} \log (1 - D(g(z^i)))$$

D maximize

$$V(\theta_g, \theta_d) = \frac{1}{m}\sum_{i=1}^{m} \log D(x^i) + \frac{1}{m}\sum_{i=1}^{m} \log (1 - D(g(z^i)))$$

D maximize

$$V(\theta_g, \theta_d) = \frac{1}{m}\sum_{i=1}^{m} \log D(x^i) + \frac{1}{m}\sum_{i=1}^{m} \log(1 - D(G(z^i)))$$

Depends on G : ~~Yes~~ No

$V(G, D)$

$$V(G) = \max_D V(G, D)$$

min max Same

$$G: \min_G V(G) = \min_G \max_D V(G, D)$$

$$\min_{\theta_g} \frac{1}{m}\sum_{i=1}^{m} \log(1 - D(G(z^i)))$$ Theory

$$\min_{\theta_g} \frac{1}{m}\sum_{i=1}^{m} -\log D(G(z^i))$$

prob of fake { max ~~min~~ } Practice

The GAN training algorithm (Goodfellow et al 2014)

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

* How is GAN training related to minimizing $JS(P_r | P_g)$?

* Why cannot we train the discriminator to optimal?

* Why use the following loss for the generator?

$$\min_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} -\log D(G(z^i))$$

D maximizes: $\frac{1}{m}\sum_{i=1}^{m}\log D(\mathbf{x}^{(i)}) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g(\mathbf{z}^{(i)})))$

$$V(G,D) = E_{x\sim P_r}[\log D(x)] + E_{x\sim P_g}[\log(1-D(x))]$$

$$= \int [P_r(x)\log D(x) + P_g(x)\log(1-D(x))]\,dx$$

Fixed Fixed

optimal D: $D^*(x) = \frac{P_r(x)}{P_r(x)+P_g(x)}$

Thumbtack: $\theta = P(H)$ Data: 3H, 7T

$$\ell(\theta|\text{Data}) = 0.3\log\theta + 0.7\log(1-\theta)$$

MLE: $\theta^* = \frac{0.3}{0.3+0.7}$

$$V(G,D) = E_{x\sim p_r}[\log D(x)] + E_{x\sim p_g}[\log(1-D(x))]$$

$$= \int [\underbrace{p_r(x)}_{\text{Fixed}} \log D(x) + \underbrace{p_g(x)}_{\text{Fixed}} \log(1-D(x))]\, dx$$

$z \longrightarrow g(z) = \bar{x}$

optimal D: $D^*(x) = \dfrac{p_r(x)}{p_r(x)+p_g(x)}$

$$G: \min_G \max_D V(G,D) = \min_G V(G,D^*)$$

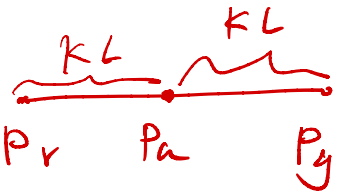
$p_a = \frac{1}{2}(p_r + p_g)$

$$V(G,D^*) = E_{x\sim p_r}\left[\log \frac{p_r(x)}{p_r(x)+p_g(x)}\right] + E_{x\sim p_g}\left[\log \frac{p_g(x)}{p_r(x)+p_g(x)}\right]$$

$$= E_{x\sim p_r}\left[\log \frac{p_r(x)}{p_a(x)}\right] + E_{x\sim p_g}\left[\log \frac{p_g(x)}{p_a(x)}\right] - 2\log 2$$

$$= 2\,\underbrace{\frac{1}{2}[KL(p_r\|p_a) + KL(p_g\|p_a)]}_{JS(p_r\|p_g)} - 2\log 2$$

G: minimize $JS(p_r\|p_g)$



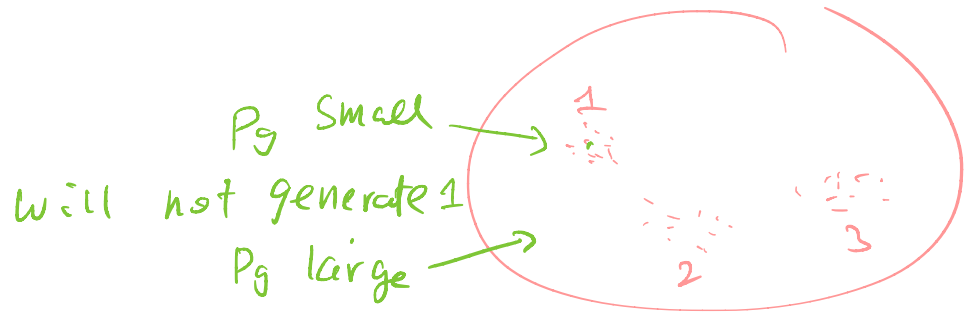
VAE : minimizes

$$KL(p_r||p_g) = \int p_r(\mathbf{x}) \log \frac{p_r(\mathbf{x})}{p_g(\mathbf{x})} d\mathbf{x}$$

JS

Different regions in the data space

	P_r	P_g	$\log \frac{P_r(x)}{P_g(x)}$	$\log \frac{P_r(x)}{P_a(x)}$	$\frac{P_g(x)}{P_a(x)}$	$\log \frac{P_g(x)}{P_r(x)}$
Mode collapse ∈	large	Small	large (unlikely)			
Unrealistic images ∈	Small	large	Small (likely)			



Loss for G:

$$\min \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D(G(z^i))\right)$$

$$\nabla_{\theta_g} E_{x \sim P_g}\left[\log(1 - D(x)\right] = \nabla_{\theta_g} \int P_g(x) \log(1 - D(x)\, dx$$

$$= \int \log(1 - D(x)) \nabla_{\theta_g} P_g(x)\, dx$$

↳ small

Slow training

Depends on θ_g { ~~yes~~ no

At beginning:

$x \sim P_g(x)$: $D(x)$ { 1 / 0

$$\min \frac{1}{m} \sum_{i=1}^{m} -\log D(G(z^i))$$

$$\min \frac{1}{m} \sum_{i=1}^{m} -\log D(G(z^i))$$

$$\nabla_{\theta_g} E_{x \sim g}[-\log D(x)] = -\nabla_{\theta_g} \int P_g(x) \log D(x)\, dx$$

$$= -\int \log D(x)\, \nabla_{\theta_g} P_g(x)\, dx$$

If D is close optimal $\hookrightarrow$ { large, ~~small~~

$\Rightarrow$ unstable training

Do not train D to optimum.

Difficulty with JS

$$JS(P_r \| P_g) = \frac{1}{2} KL(P_r \| P_a) + \frac{1}{2} KL(P_g \| P_a)$$

$$Supp(P_r) = \{x \mid P_r(x) > 0\}$$

≈ 0

$x \in \mathbb{R}^{d_r}$

$d_r = 256 \times 256$

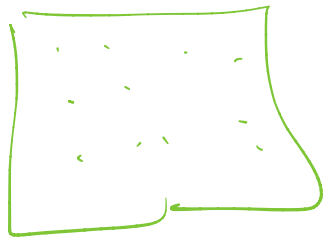
$= 56K$

$$Supp(P_g) = \{x = g(z) \mid z \in \mathbb{R}^{d_z}\}$$

≈ 0

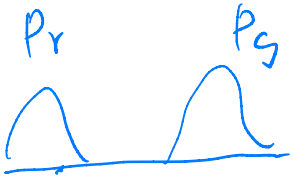
$d_z \ll d_r$

100 $\quad$ 56K

$P($  look real $)$

Volume of 2D surface in 3D space $= 0$

≈ 0



$$\text{Supp}(P_r) \cap \text{Supp}(P_g) = \phi$$

$$JS(P_r \| P_g) = \frac{1}{2} KL(P_r \| P_a) + \frac{1}{2} KL(P_g \| P_a)$$

$$= \log 2$$

$$KL(P_r \| P_a) = \int P_r(x) \log \frac{P_r(x)}{P_a(x)} dx$$

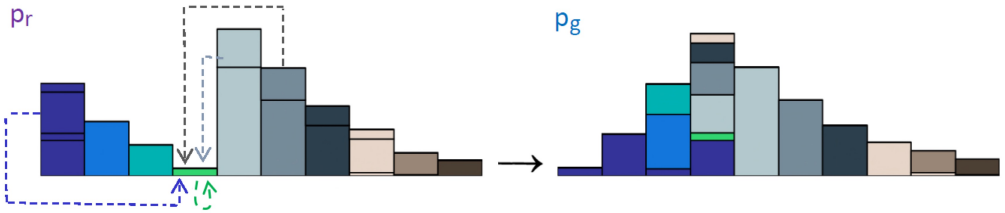
$P_g(x) = 0 \leftarrow x \in \text{Supp}(P_r)$

$$= \int_{x \in \text{Supp}(P_r)} \frac{P_r(x)}{P_a(x)} \log 2 \, dx$$

$$\geq \log 2$$

$$P_a(x) = \frac{1}{2}(P_r(x) + P_g(x))$$

Earth Mover's Distance / Wasserstein Distance



$\gamma(x, y)$: amount of dirt from x to y

① $\gamma(x, y) \geq 0$

② $\sum_y \gamma(x, y) = P_r(x)$

③ $\sum_x \gamma(x, y) = P_g(y)$

$\sum_{x,y} \gamma(x, y) = 1$

amount of dirt at x at start

prob dist

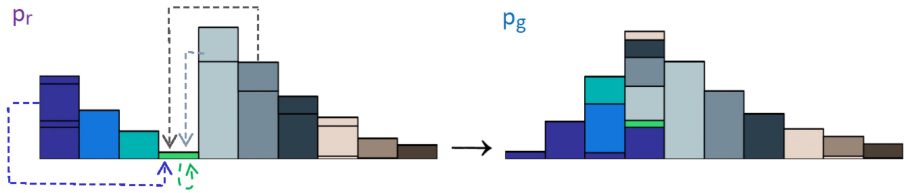
Follow $\gamma(x, y)$, the total amount of work

$$\sum_{x,y} \gamma(x, y) \|x - y\| = E_{(x,y)\sim\gamma} \|x - y\|$$

γ: Prob Dist

$$EMD(P_r, P_g) = \min_{\gamma} E_{(x,y)\sim\gamma} \|x - y\|$$

Earth Mover's Distance or Wasserstein Distance



$$EMD(p_r, p_g) = \min_{\gamma} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} ||\mathbf{x} - \mathbf{y}||$$

$\gamma(x, y)$: Amount to transport from x to Y

optimal transport problem

Wasserstein GAN (WGAN)

$$\arg\min_{\boldsymbol{\theta}} EMD(p_r, p_{\boldsymbol{\theta}})$$

unable to optimize directly

$$EMD(p_r, p_g) = \min_{\gamma} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}||\mathbf{x} - \mathbf{y}||$$

Kantorovich-Rubinstein Duality

High scores for real

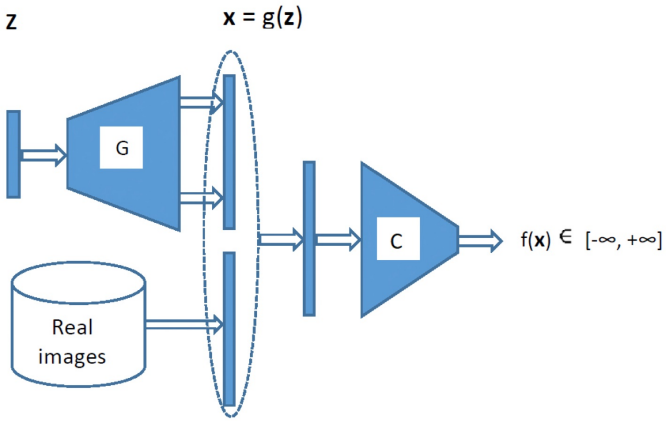
Low scores for fake

$$EMD(p_r, p_\theta) = \max_{f \in \mathcal{F}_{1-L}} (\mathbb{E}_{\mathbf{x}\sim p_r}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_\theta}[f(\mathbf{x})])$$

$\downarrow$

$\mathcal{F}_{k-L}$

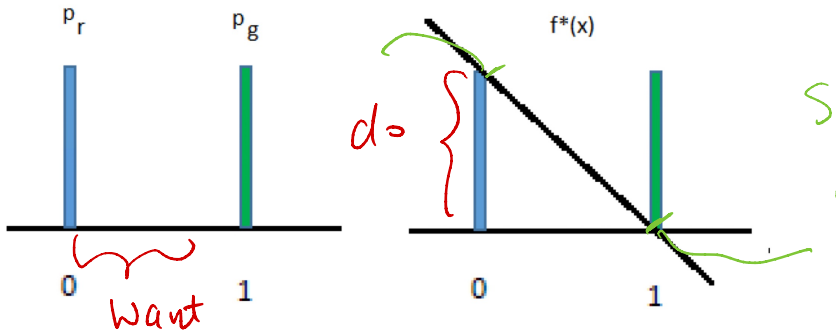
$EMD_k = k\,EMD$



$$L_c = \mathbb{E}_{\mathbf{x}\sim p_r}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_\theta}[f(\mathbf{x})]$$

restrict each weight to be from an interval $[-c, c]$.

to make f Lipschitz



Distance ← Height

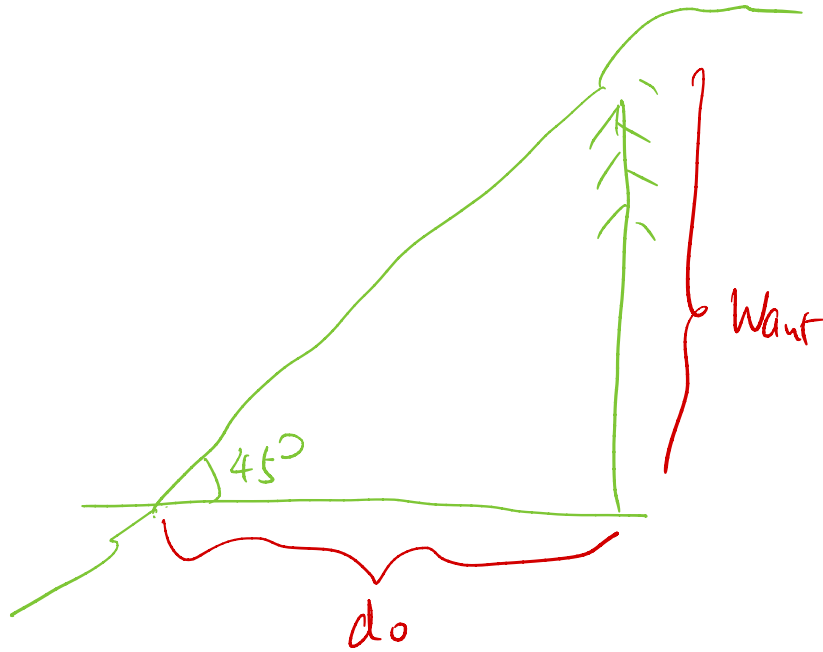
Slope ≤ 1 everywhere?

↳ in [0, 1]

$EMD(p_r, p_\theta) = \max_{f \in \mathcal{F}_{1-L}} (\mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)])$

$\geq \max_{f(x)=ax+b, |a| \leq 1} (b - (a+b))$

$= \max_{|a| \leq 1} -a = 1$



Height ← Distance