LEC-4 Deep Generative Modeling

Unsupervised Learning

x data, no labels → Learn hidden/underlying structure

Generative modeling

Goal: Take as input training samples from some distribution & learn a model that represents that distribution

Density estimation

Sample generation

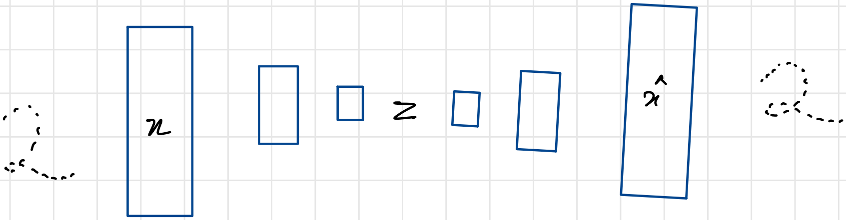
Outliner detection

Latent Variable models

- Autoencoders & Variational encoders
- Generative Adversial Networks (GANs)

Autoencoders

Encoders learns mapping from the data x to a low dimensional latent space z

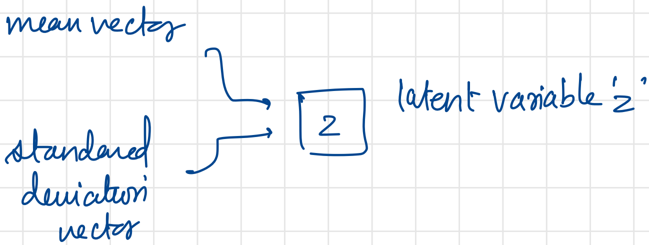


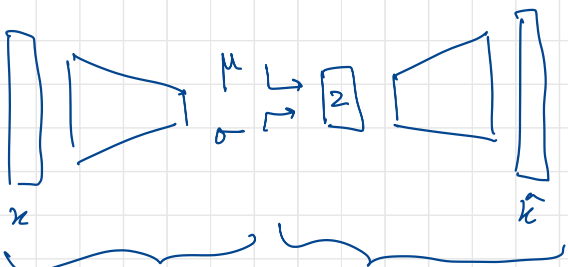
$$\mathcal{L}(x,\hat{x}) = \|x-\hat{x}\|^2$$

Bottleneck hidden layer forces network to learn a compressed latent representation

Reconstruction loss forces latent representation to capture / encode as much information as possible.

Variational Autoencoder → probabilistic twist on traditional autoencoder
→ non-deterministic





encoder: $q_\phi(z, x)$ Decoder $p_\theta(x|z)$

$\mathcal{L}(\phi, \theta, x) = $ (reconstruction loss) + (regularization loss)

↑ encoder ↑ decoder

$D(q_\phi(z|x) \,\|\, p(z))$

difference between probability distributions

$$D\left(q_\phi(z|x) \,\|\, p(z)\right)$$

Inferred latent distribution

fixed prior on latent distribution

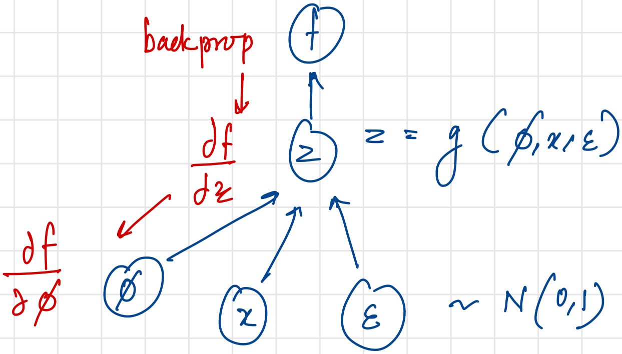
K-L divergence

$$= -\frac{1}{2} \sum_{j=0}^{k-1} \left(\sigma_j + \mu_j^2 - 1 - \log \sigma_j\right)$$

Reparameterizing the sampling layer

* fixed μ vector
* fixed σ vector scaled by random constants drawn from the prior distribution

$$\Rightarrow \quad z = \mu + \sigma \odot \varepsilon$$

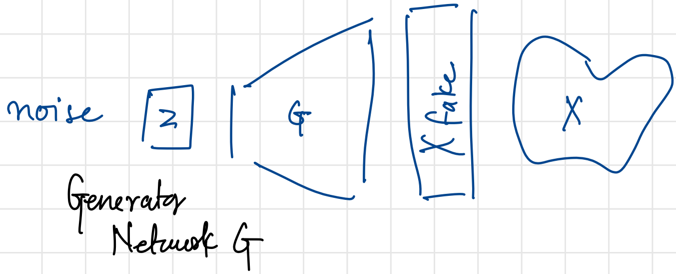


$$\mathcal{L}(\theta, \phi; x, z, \beta) = \underbrace{E_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right]}_{\text{reconstruction}} - \underbrace{\beta D_{KL}\left(q_\phi(z|x) \,\|\, p(z)\right)}_{\text{regularization}}$$

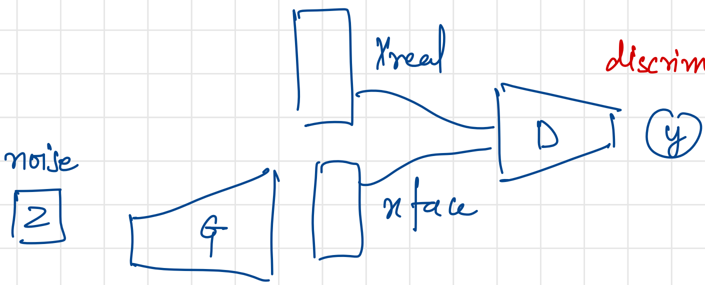
Generative Adversarial Networks

Idea: do not explicitly model density — use samples for new instances

Solution: sample from something simple (noise) learn a transformation to the data distribution



fake sample from learned representation of data distribution



discriminator — identify real data from fake

generator turns noise into an imitation of data to trick discriminator

GAN loss function

$$\arg\max_{D} E_{z,x}\left[\overbrace{\log D(G(z))}^{\text{Fake}} + \overbrace{\log(1-D(x))}^{\text{Real}}\right]$$

Discriminator tries to identify synthesized images

Generator

$$\arg\min_{G} E_{z,x}\left[\log D(G(z)) + \log(1-D(x))\right]$$

final loss function

$$\arg\max_{D}\min_{G} E_{z,x}\left[\log D(G(z)) + \log(1-D(x))\right]$$