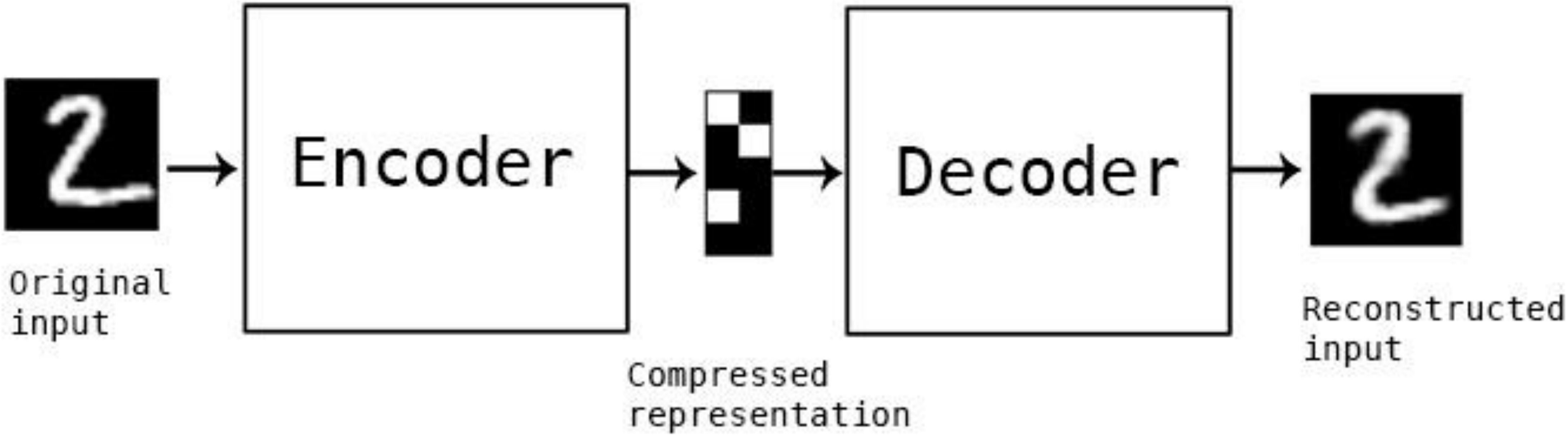
Autoencoders and Variational Autoencoders

Gautam Kamath

Autoencoder

- A type of “compression”
- Input: $x \in \mathbf{R}^d$
- Encoder: $f: \mathbf{R}^d \to \mathbf{R}^m$, Decoder: $g: \mathbf{R}^m \to \mathbf{R}^d$
- Goal: $g\big(f(x)\big) = x$
- Trivial if $m = d$: just let $f(x) = x$ and $g(x) = x$
- Interesting when $m \ll d$ (e.g., $d = 1000, m = 10$)

Autoencoder



Linear Autoencoder

- (Draw simple autoencoder, label weights W_f and W_g and bottleneck)
- Output: $W_g W_f x$
- How to optimize? Use objective

$$\min_{W_f, W_g} \sum_i \frac{1}{2} \left\| W_g W_f x_i - x_i \right\|_2^2$$

- $W_f x$ is a compression of x
- With linear autoencoder, similar to principal component analysis (PCA) (draw)

Nonlinear Autoencoder

- f and g are non-linear (draw non-linear auto-encoder, label W_f, W_g)
- $\min_{W_f, W_g} \sum_i \frac{1}{2} \|g(f(x_i)) - x_i\|_2^2$
- Deep autoencoder (draw)

Other Autoencoders

- Sparse autoencoders
  - Encourage a sparse encoding of input
  - May have wider bottleneck layer (draw)
  - $\min_{W_f, W_g} \sum_i \frac{1}{2} \|g(f(x_i)) - x_i\|_2^2 + \lambda \|f(x_i)\|_1$
- Denoising autoencoders
  - Given noised input $\tilde{x}$, produce denoised x as output (draw)
  - $\min_{W_f, W_g} \sum_i \frac{1}{2} \|g(f(\tilde{x}_i)) - x_i\|_2^2$

Uses of Autoencoders

- Can "detach" input and output, use separately
- Can compress data to a smaller dimension
- Can find interesting representations of data
- Generally, finds some underlying structure of the dataset
- However, is not useful to understand *distribution* of dataset
  - In particular, can't necessarily generate new images

Generative Modelling

- Given $X_1, \ldots, X_n \sim D$, can we generate $X_{n+1}, X_{n+2}, \ldots$?
  - Ideally from D, but actually from something *close* to D
- D may be more complex than a GMM
  - E.g., the distribution of all handwritten numbers, or ImageNet (draw)
- Solution: use a neural network to do the work
- Draw a sample from $N(0, I)$, use an NN to map it to a sample from D
- (Draw NN version, where low d Gaussian mapped to high d output)
- Actually: use variational autoencoder (VAE)

Variational Autoencoder

- (Draw encoder, from $x \in \mathbf{R}^d$ to $\mu(x), \sigma(x) \in \mathbf{R}^m$, decoder from $z \sim N\left(\mu(x), \mathrm{diag}(\sigma(x))\right) \in R^m$ to $\tilde{x} \in \mathbf{R}^d$)

Variational Autoencoder (VAE)

- Some notation: x's live in the *data* space (in $\mathbf{R}^d$), while z's live in the *latent* space (in $\mathbf{R}^m$). p_θ is the decoder network's distribution, q_ϕ is the encoder network's distribution
- E.g., $p_\theta(x)$ is density of decoder network's outputs. $p_\theta(x|z)$ is density of decoder network's outputs, *conditioned on* some latent vector input z. $p_\theta(z)$ is density of decoder network's latent vector input. $q_\phi(z|x)$ is distribution of encoder network's outputs, *conditioned on* some data input x
  - $p_\theta(z)$ generally chosen to be $N(0, I)$
  - Why does $p_\theta(x|z)$ have a distribution? Isn't it deterministic? For loss calculation, we assume the output of the network is fed into a Gaussian sampler. Will revisit shortly.
  - (Draw mapping from data space to latent space and back)

VAE Goals

- Ensure that input image distribution maps to latent distribution $N(0, I)$ (draw)
  - Minimize $KL\left(q_\phi(z|x)||p_\theta(z)\right) = KL\left(q_\phi(z|x)||N(0,I)\right)$ (draw lines)
- Similar to autoencoder, ensure that an input gets encoded and mapped back to itself
  - Maximize $E_{z\sim q_\phi(\cdot|x)}[\log p_\theta(x|z)]$
- Claim: $\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL\left(q_\phi(z|x)||N(0,I)\right)$
  - Similar to the inequality when doing EM
  - Bigger picture: variational inference

Optimizing: Minimize KL divergence

- $\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - \boldsymbol{KL}\left(\boldsymbol{q_\phi}(\boldsymbol{z}|\boldsymbol{x})||\boldsymbol{N}(\boldsymbol{0},\boldsymbol{I})\right)$
- $KL\left(q_\phi(z|x)||N(0,I)\right) = KL\left(N(\mu_\phi(x), \mathrm{diag}(\sigma_\phi^2(x)))||N(0,I)\right)$
- For two Gaussians, this KL divergence has a simple expression

$$= \frac{1}{2}\left(\left\|\mu_\phi(x)\right\|_2^2 - m + \sum_{j=1}^{m}\left(\sigma_\phi^2(x)_j - \log\left(\sigma_\phi^2(x)_j\right)\right)\right)$$

- Sanity check: what if $\mu_\phi(x) = 0$ and $\sigma_\phi^2(x)_j = 1$ for all j?

Optimizing: Autoencoding points

- $\log p_\theta(x) \geq \boldsymbol{E_{z\sim q_\phi(z|x)}}[\mathbf{log}\, \boldsymbol{p_\theta(x|z)}] - KL\big(q_\phi(z|x)||N(0,I)\big)$
- We imagine the density $p_\theta(x|z)$ is that of $N(\mu_\theta(z), I)$ where μ_θ is the decoder network
  - When sampling, can instead just output $\mu_\theta(z)$ rather than additional sampling
    - Analogy: when we run softmax on outputs of an NN, we output the max index, we don't sample from it
- $E_{z\sim q_\phi(z|x)}[||x - \mu_\theta(z)||_2^2]$ ~~$- d\log\sqrt{2\pi}$~~ (essentially same as AE)
- Given sampling capability, can draw $z \sim q_\phi(z|x)$ to optimize
- Reparameterization trick (Draw how to sample $Z \sim N(\mu, \sigma^2)$ as $\mu + \sigma G$ where $G \sim N(0,1)$)

Summary

- Solve generative modelling
- Use neural network to map Gaussian samples to data distribution
- Do it by using variational autoencoder: tries to map original distribution to a Gaussian, and also maps back to original distribution. Each is encoded in the loss function.

Samples from a VAE

(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Interpolation using VAEs (Explain how)