

Variational Autoencoders



What are Generative Models

Form of unsupervised learning

Goal: generate data indistinguishable from actual data

Take some input z \implies Output a “generated” example e.g an image

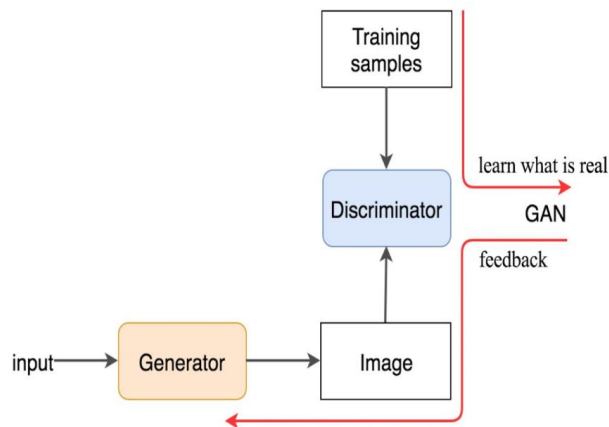
z : sample from the latent space where the sample represents a distribution

Latent space: unseen representation of data

Approaches to Generative Modeling

GANs

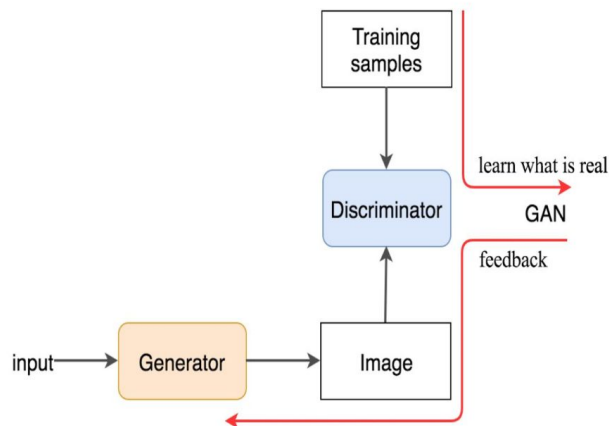
- Sample from latent space $P(z)$
 - Latent space itself has no meaning in reference to data distribution
 - $z \sim N(0, 1)$



Approaches to Generative Modeling

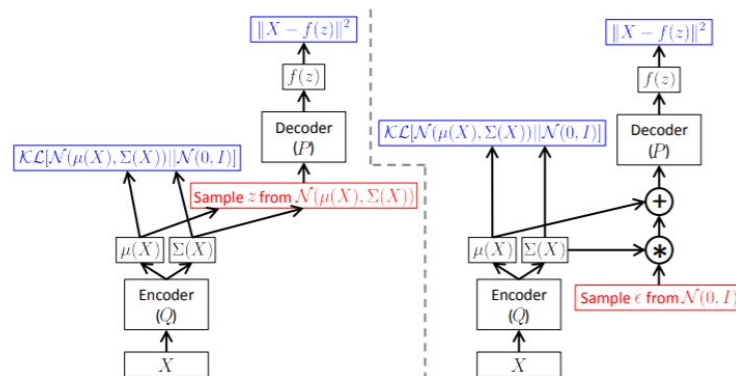
GANs

- Sample from latent space $P(z)$
 - Latent space itself has no meaning in reference to data distribution
 - $z \sim N(0, 1)$



VAEs

- Sample from latent space $P(z)$
 - Latent space is a learned representation of true data distribution $P(X)$
 - $z \sim N(\mu(X), \Sigma(X))$



Main Differences

GANs

Two networks “fight” to generate realistic images

Most of the heavy lifting is done by the generator and discriminator

More difficult to train and takes longer, but can produce better/sharper images

Main Differences

GANs

Two networks “fight” to generate realistic images

Most of the heavy lifting is done by the generator and discriminator

More difficult to train and takes longer, but can produce better/sharper images

VAEs

Train to improve the latent space to generate realistic images

Creates a semi-supervised learning environment by minimizing loss of a generated image

Takes less time to train, but images can be quite fuzzy

Exploring the Latent Space

GANs

Latent space is a random sample from same distribution

Samples from the latent space are interpreted by the generator

Exploring the Latent Space

GANs

Latent space is a random sample from same distribution

Samples from the latent space are interpreted by the generator

VAEs

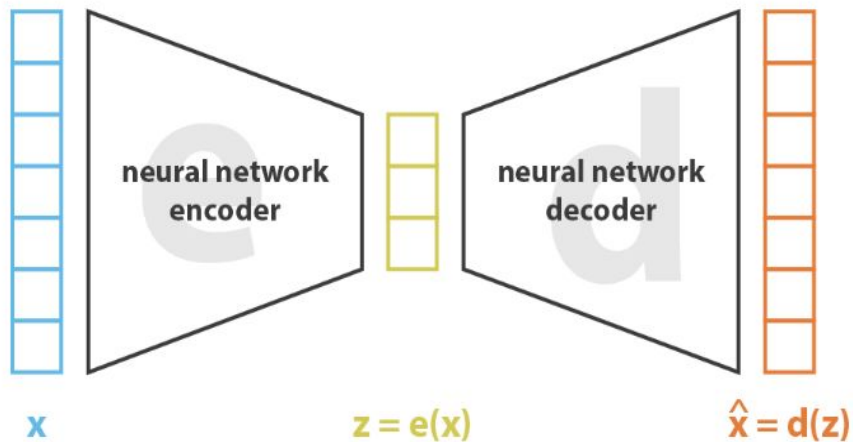
Can think of the latent space as attributes/variables that are characteristic to the data

For example, what image to generate, the angle in which to generate the image, other abstract stylistic properties

Important that the latent space be regularized as to generate images that have meaning

Autoencoders vs Variational Autoencoders

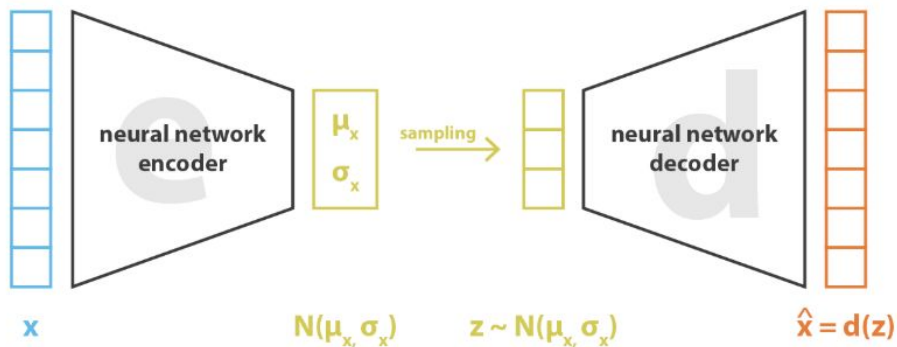
- Autoencoder: Learn a compressed representation of the data



$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$

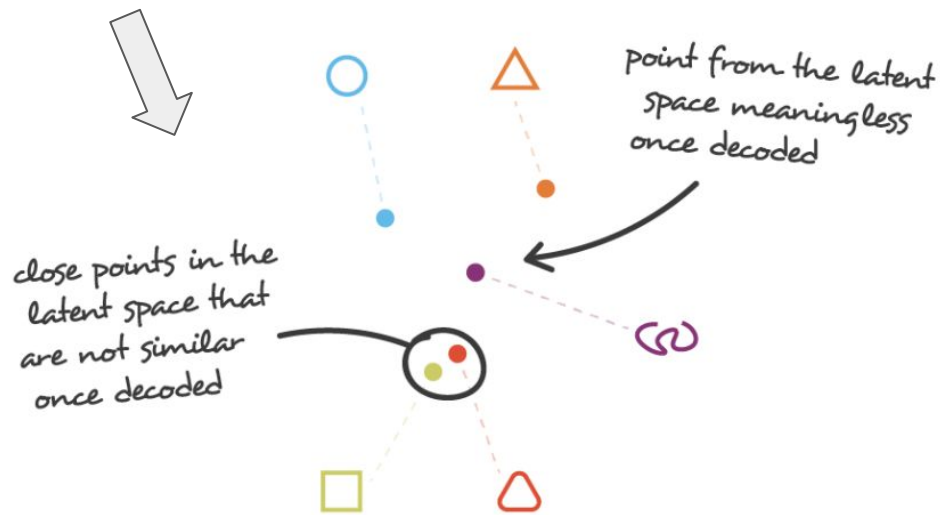
Autoencoders vs Variational Autoencoders

- Variational Autoencoder: Learn a compressed representation of the data subject to some constraint



$$\text{loss} = ||x - \hat{x}||^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = ||x - d(z)||^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

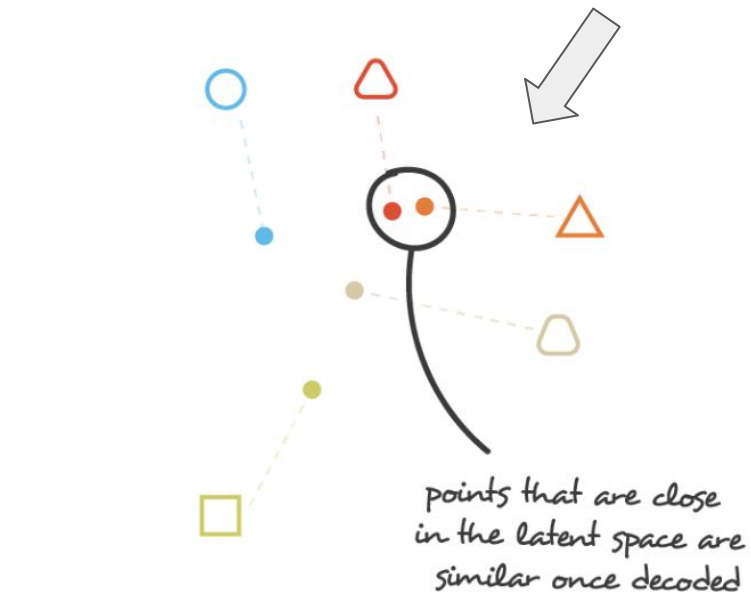
Autoencoder



irregular latent space



Variational Autoencoder



regular latent space



Difference between a "regular" and an "irregular" latent space.

VAEs and their assumptions

Maximize this equation:

$$P(X) = \int P(X|z; \theta) P(z) dz$$

VAEs and their assumptions

Maximize this equation:

$$P(X) = \int P(X|z; \theta) P(z) dz$$



Any datapoint
in our
distribution

VAEs and their assumptions

Maximize this equation:

$$P(X) = \int P(X|z; \theta) P(z) dz$$



Any datapoint
in our
distribution

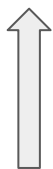


$P(X)$, given that we observed z ,
and given model parameters

VAEs and their assumptions

Maximize this equation:

$$P(X) = \int P(X|z; \theta) P(z) dz$$



Any datapoint
in our
distribution



$P(X)$, given that we observed z ,
and given model parameters



Probability of a latent
representation, z

How do we maximize the probability of X ?

- First we need a form of this equation that we can backpropagate through...

$$P(X) = \int P(X|z; \theta) P(z) dz$$

How do we maximize the probability of X ?

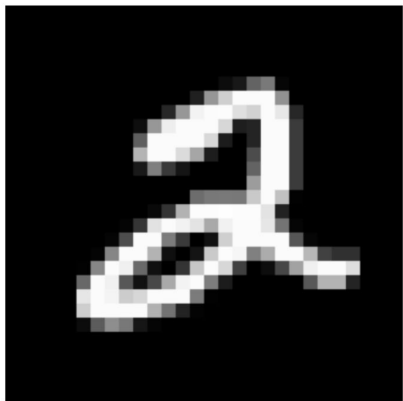
- First we need a form of this equation that we can backpropagate through...

$$P(X) = \int P(X|z; \theta) P(z) dz$$

$$P(X) \approx \frac{1}{n} \sum_i P(X|z_i)$$

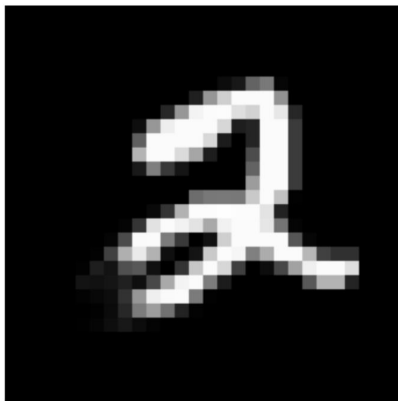
Recall that we assumed $P(X|z)$ to be gaussian

$$P(X) \approx \frac{1}{n} \sum_i P(X|z_i)$$



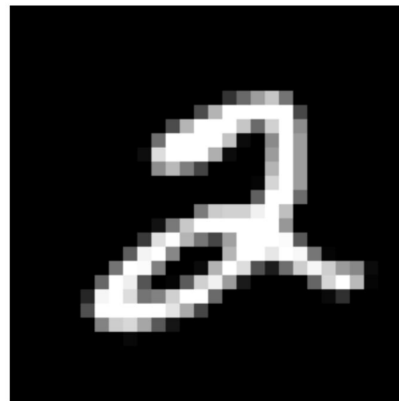
(a)

Distance. to (a):
0



(b)

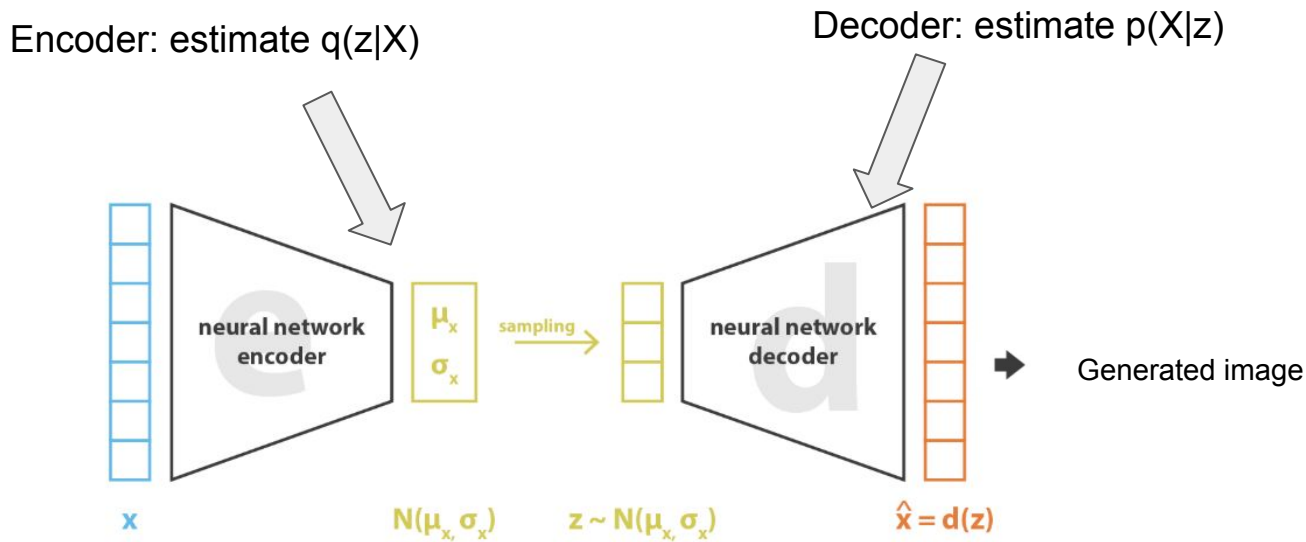
Distance to (a):
0.04



(c)

Distance to (a)
0.27

Instead of sampling over all latent variables...



$$\text{loss} = ||x - \hat{x}||^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = ||x - d(z)||^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

OBJECTIVE FUNCTION

For an Image X_i The law of total probability states

$$P(X_i) = \int_z P(X_i|z)P(z)dz$$

$$P(X_i) = \int_z P(X_i|z_1, z_2, z_3, \dots)P(z_1, z_2, z_3, \dots)dz_1 dz_2 dz_3 \dots$$

OBJECTIVE FUNCTION

For an Image X_i The law of total probability states

$$P(X_i) = \int_z P(X_i|z)P(z)dz$$

$$P(X_i) = \int_z P(X_i|z_1, z_2, z_3, \dots)P(z_1, z_2, z_3, \dots)dz_1 dz_2 dz_3 \dots$$

Now we want to maximize the log likelihood L.

$$L = \sum_{X_i \in X} \log(P(X_i))$$

We can Approximate integrals with sums.

We can Approximate integrals with sums.

$$P(X_i) = \int_z P(X_i|z)P(z)dz \approx \frac{1}{n} \sum_{i=1}^n P(X|z_i)$$

We can Approximate integrals with sums.


$$P(X_i) = \int_z P(X_i|z)P(z)dz \approx \frac{1}{n} \sum_{i=1}^n P(X|z_i)$$

If z lives in a large space like \mathbf{R}^{100} we need to know where to look.

So we introduce our encoder Q to approximate $P(z|X_i)$

$Q(z|X_i)$ is taken to be a multivariate normal distribution.

In the paper the sigma vector was taken to be the log of the variance.

$$Q(z|X_i) = N(z|\mu(X_i), \Sigma(X_i))$$


where μ, Σ are outputs of a neural network

KL DIVERGENCE

If we want Q to approximate $P(z|X)$ we need a measure of similarity between two probability distributions.

One such measure is Kullback-Leibler(KL) divergence.

KL DIVERGENCE

If we want Q to approximate $P(z|X)$ we need a measure of similarity between two probability distributions.

One such measure is Kullback-Leibler(KL) divergence.

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

For continuous distributions

$$D_{KL}(P||Q) = \int_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right) dx$$

Properties of KL divergence

$$D_{KL}(P||Q) \geq 0$$

$$D_{KL}(P||Q) = 0 \iff Q = P$$

$$D_{KL}(P||Q) \neq D_{KL}(Q||P)$$

There exists a closed form for the KL divergence between two multivariate normal distributions.

$$D_{KL}(\mathcal{N}_0 \parallel \mathcal{N}_1) = \frac{1}{2} \left(\text{tr}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1} (\mu_1 - \mu_0) - k + \ln \left(\frac{\det \Sigma_1}{\det \Sigma_0} \right) \right).$$

$$D_{KL}(Q(z|X_i)||P(z|X_i)) = E_{z \sim Q}[\log(Q(z|X_i)) - \log(P(z|X_i))]$$

Recall Bayes Theorem

$$P(z|X_i) = \frac{P(X_i)P(X_i|z)}{P(z)}$$

$$D_{KL}(Q(z|X_i)||P(z|X_i)) = E_{z \sim Q}[\log(Q(z|X_i)) - \log(P(X_i|z)) + \log(P(X_i)) - \log(P(z))]$$

Do some algebra and we arrive at,

$$\log(P(X_i)) - D_{KL}(Q(z|X_i)||P(z|X_i)) = E_{z \sim Q}[\log(P(X_i|z))] - D_{KL}(Q(z|X_i)||P(z))$$

Job is to make
our data as likely
as possible

Job is to give us
z's likely to
produce X

LHS: Not computable

RHS: computable

Remember the prior distribution for z .

$$z \sim N(0, I)$$

$$\log(P(X_i)) - D_{KL}(Q(z|X_i)||P(z|X_i)) = E_{z \sim Q}[\log(P(X_i|z))] - D_{KL}(Q(z|X_i)||N(0, I))$$

let $M \subset X$ be our mini-batch

Then our objective function is

$$E_{X_i \in M} E_{z \sim Q}[\log(P(X_i|z))] - D_{KL}(Q(z|X_i)||N(0, I)) = \frac{1}{|M|} \sum_{X_i \in M} (E_{z \sim Q}[\log(P(X_i|z))] - D_{KL}(Q(z|X_i)||N(0, I)))$$

$$E_{z \sim Q} [\log(P(X_i|z))] - D_{KL}(Q(z|X_i) || N(0, I))$$

Can't be computed exactly so we sample many z 's from Q and estimate the expected value with the average.

$$E_{z \sim Q} [\log(P(X_i|z))] - D_{KL}(Q(z|X_i) || N(0, I)) \approx \frac{1}{n} \sum_{j=1}^n [\log(P(X_i|z_j))] - D_{KL}(Q(z|X_i) || N(0, I))$$

Luckily, the original authors state that n can be set to 1.

$$E_{z \sim Q} [\log(P(X_i|z))] - D_{KL}(Q(z|X_i) || N(0, I))$$

Can't be computed exactly so we sample many z 's from Q and estimate the expected value with the average.

$$E_{z \sim Q} [\log(P(X_i|z))] - D_{KL}(Q(z|X_i) || N(0, I)) \approx \frac{1}{n} \sum_{j=1}^n [\log(P(X_i|z_j))] - D_{KL}(Q(z|X_i) || N(0, I))$$

Luckily, the original authors state that n can be set to 1.

Can you spot a problem that arises when going from the expectation to the average?

$$E_{z \sim Q} [\log(P(X_i|z))] - D_{KL}(Q(z|X_i) || N(0, I))$$

Can't be computed exactly so we sample many z's from Q and estimate the expected value with the average.

$$E_{z \sim Q} [\log(P(X_i|z))] - D_{KL}(Q(z|X_i) || N(0, I)) \approx \frac{1}{n} \sum_{j=1}^n \log(P(X_i|z_j)) - D_{KL}(Q(z|X_i) || N(0, I))$$

Can you spot a problem that arises when going from the expectation to the average?

$$E_{z \sim Q} [\log(P(X_i|z))]$$

Depends on Q and thus the parameters in the encoders network.

$$\frac{1}{n} \sum_{j=1}^n [\log(P(X_i|z_j))]$$

This sum does not depend on Q. Once the z's are sampled they are just constants. We can't differentiate the random sampling process without a trick.

Reparameterization Trick

$$Q \sim N(\mu(X_i), \Sigma(X_i))$$

From a simple property of normal distributions.

$$Q = \mu(X_i) + \Sigma(X_i) * \epsilon \quad \epsilon \sim N(0, I)$$

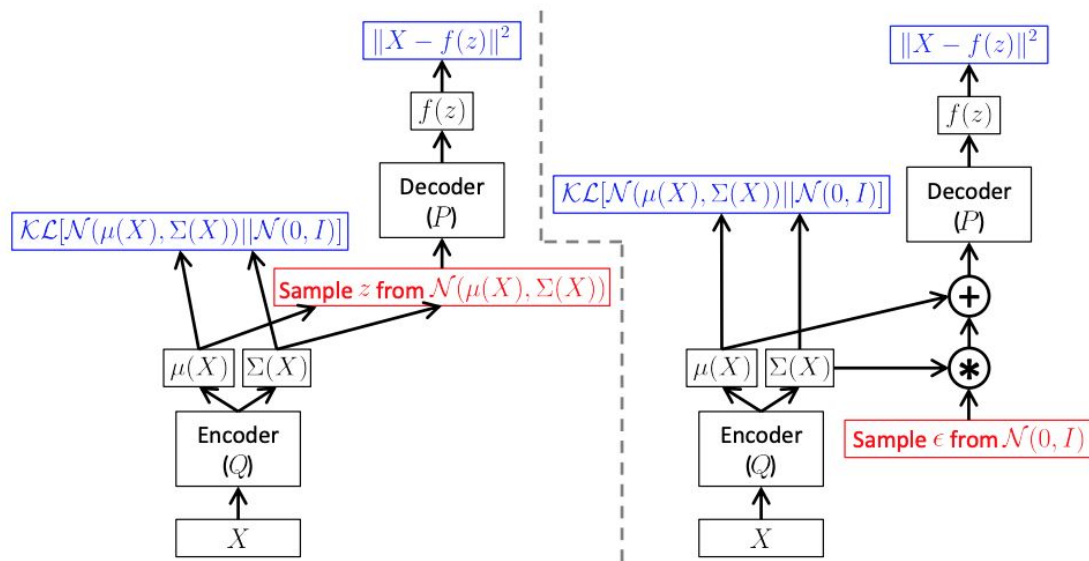
Now we sample from epsilon and calculate the z's.

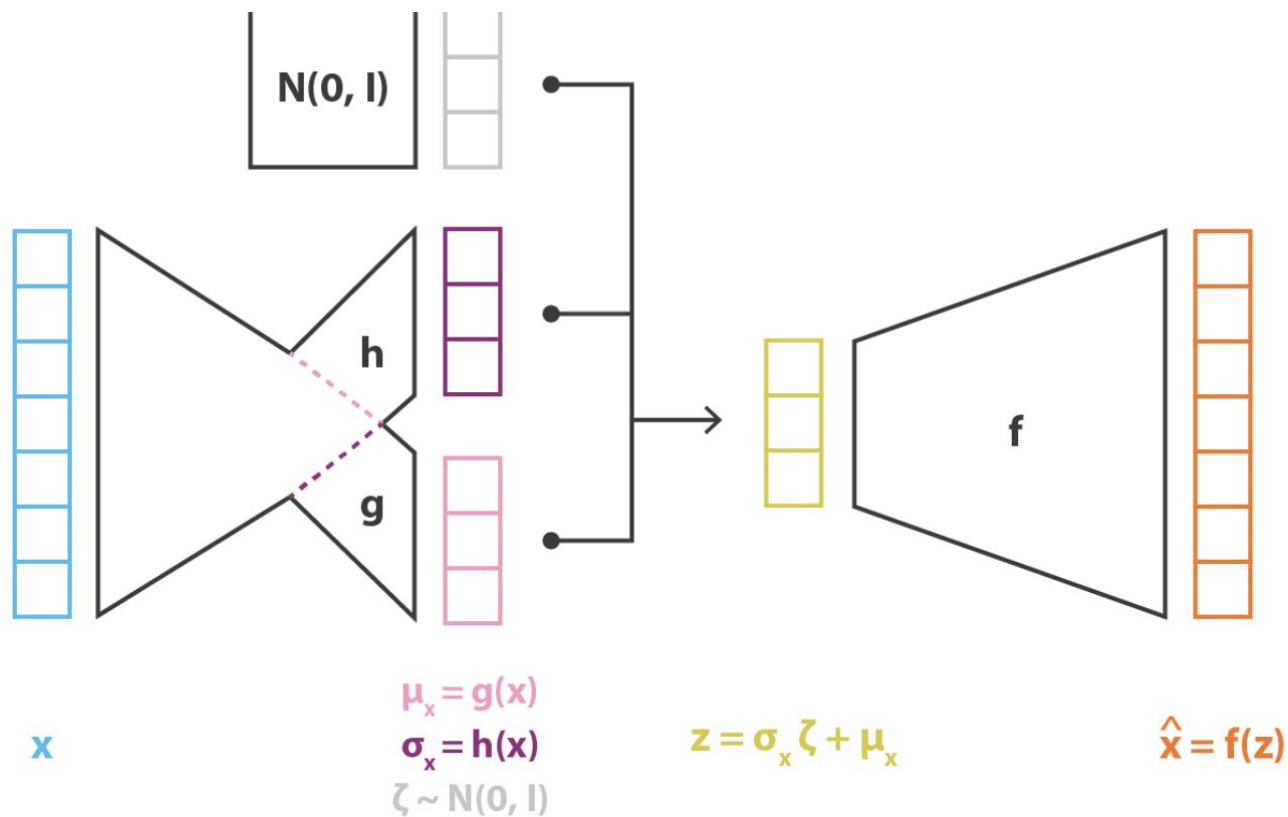
$$z_j = \mu(X_i) + \Sigma(X_i) * \epsilon_j$$

$$z_j = \mu(X_i) + \Sigma(X_i) * \epsilon_j$$

Why is the problem now fixed?

The sampling process is now differentiable with respect to the encoders parameters since the encoder can be treated as constants.





$$\text{loss} = C \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = C \|x - f(z)\|^2 + \text{KL}[N(g(x), h(x)), N(0, I)]$$


Parameter Update

Let m be the size of the mini-batches, n be the size of the sample from Q .

$$\nabla Loss = -\frac{1}{m} \sum_{i=1}^m \nabla (\log(P(X_i|z = \mu(X_i) + \sum(X_i) * \epsilon_j)) - D_{KL}[Q(z|X_i)||N(0, I)])$$

$$\log(P(X_i|z)) \propto \log(e^{\frac{1}{2}(X_i - f(z; \theta))^T (X_i - f(z; \theta))}) = \frac{1}{2}(X_i - f(z; \theta))^T (X_i - f(z; \theta))$$

Sum Squared Error



$$\nabla Loss = \frac{1}{m} \sum_{i=1}^m \nabla (D_{KL}[Q(z|X_i)||N(0, I)] - \frac{1}{2}(X_i - f(z; \theta))^T (X_i - f(z; \theta)))$$

The rest is up to auto diff and the choice of optimizer.

Questions

- Remember face algebra from GANs? You can do the same thing with VAEs.

Do you think VAEs can learn to “count” objects within images and perform algebra on their latent representations?

- What would happen if we put a scalar on the KL term in the loss?

Sources

<https://indabaxmorocco.github.io/materials/posters/El-Kaddoury1.pdf>

<https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>

<https://arxiv.org/pdf/1606.05908.pdf>