Autoencoders

INNOPOLIS UNIVERSITY

Contents

Course Structure

- **Lectures** (Offline) – Resources and links will be provided before the class)
- Labs (**Offline** in-class)
- **One Project** (Mostly programming, presentations and some theoretical questions during checkpoints) - Released after mid-term
- Midterm and Final (**Written and Oral Exam**)

Grading

- Mid-term Oral Exam (20%)
- Project (in teams): 30%
- Final Written Exam + Oral Exam (30% + 10%)
- Lab Tasks (10%)

NO BONUS TASKS or ALTERNATIVE

Grade Scale

➔ Scale Rounded

- A: >= 90%
- B: >= 80%
- C: >= 65%
- D: < 65%

Acknowledgement

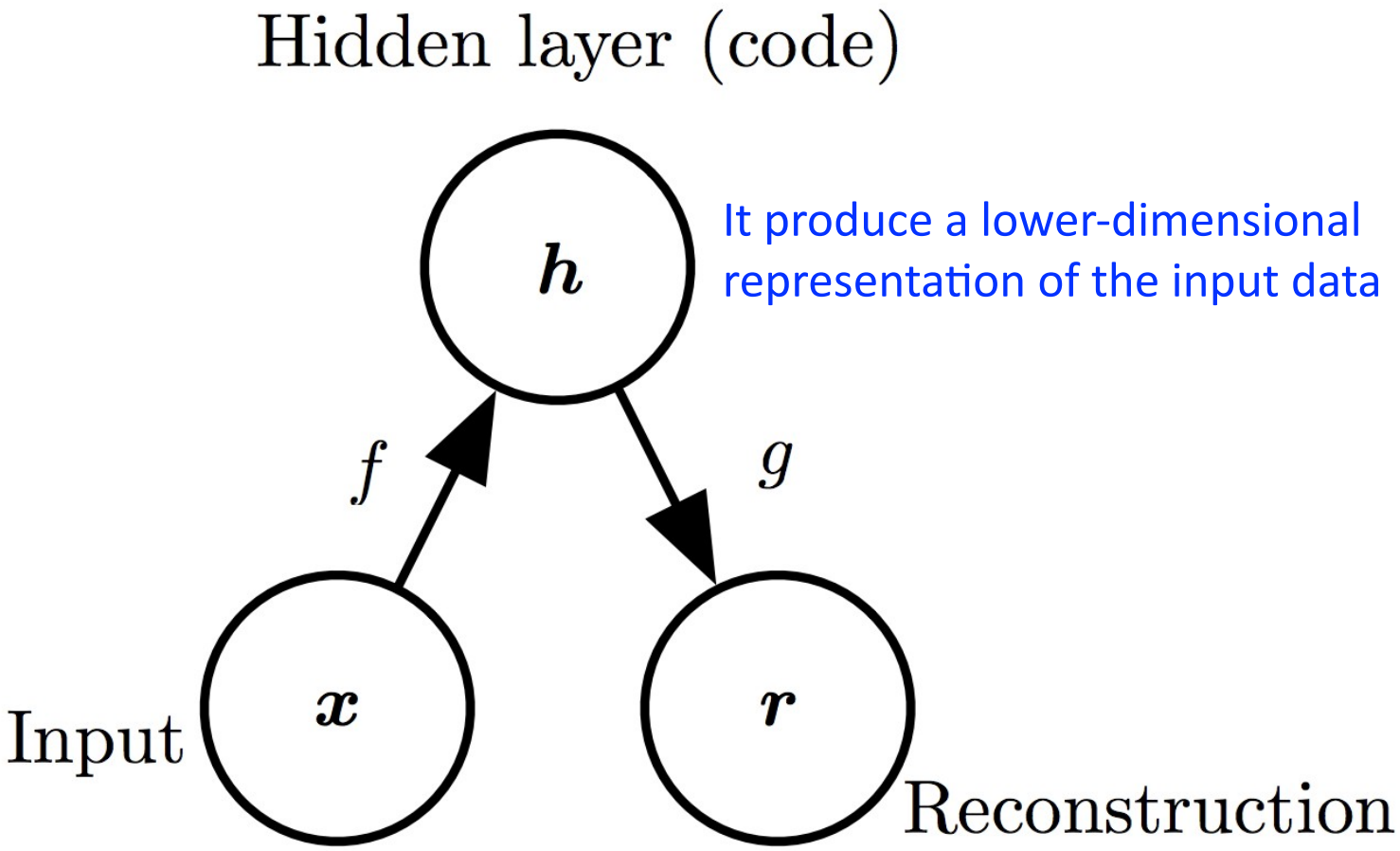
- This lecture is based on the following material and some other resources over the internet
  - http://www.deeplearningbook.org/contents/autoencoders.html
  - https://seas.ucla.edu/~kao/nndl/lectures/vae.pdf
  - http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf
  - https://www.jeremyjordan.me/autoencoders/
  - https://arxiv.org/pdf/1601.00670.pdf
  - https://www.jeremyjordan.me/variational-autoencoders/
  - https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/
  - Variational Autoencoder by Ahlad Kumar

Autoencoder

- An Autoencoder is a feedforward neural network that is trained to attempt to copy its input to its output.

$$input = output$$

Autoencoder – Architecture (Presentation 1)



It produce a lower-dimensional representation of the input data

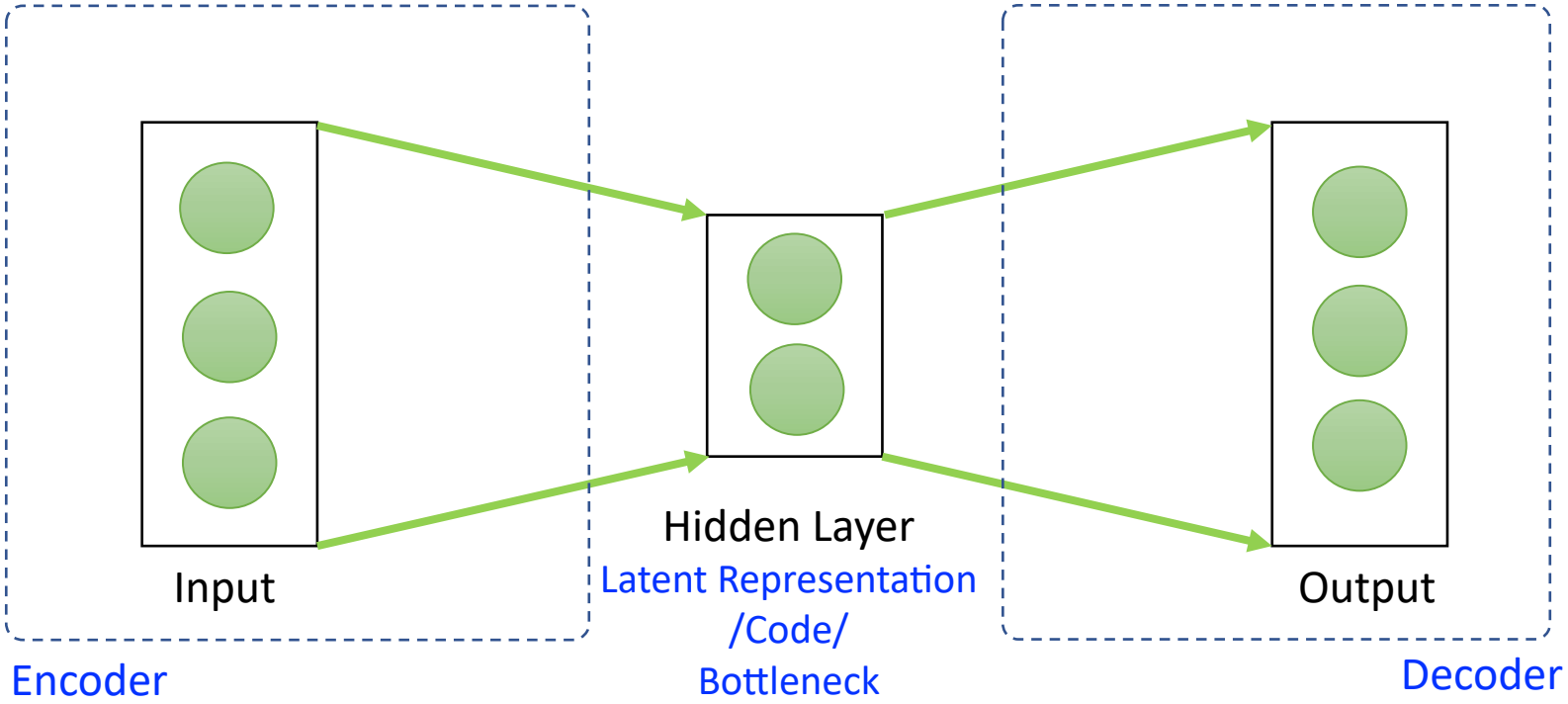
$$\mathbf{x} \in \mathbb{R}^n$$

$$\mathbf{h} \in \mathbb{R}^d$$

where $d < n$

h is designed to contain most of the important features of x to reconstruct it.

Autoencoder – Architecture (Presentation 2)



- The autoencoder has two components: the encoder and the decoder
- An encoder function is to create a hidden layer (or multiple layers) which contains a code to describe the input.
- There is then a decoder which creates a reconstruction of the input from the hidden layer.

Autoencoder – Learning and Activation Function

- It tries to learn an approximation of an **"identity"** function

- Neural Network training strategies work here!!
  - Backpropagation
  - Loss Function
  - Activation Functions
  - Regularization

- We can obtain optimum weights for this by starting with random weights and calculating the gradient

Autoencoder – Mathematically

- **Encoder:** Perform a dimensionality reduction step on the data, $x \in \mathbb{R}^n$ to obtain features $h \in \mathbb{R}^d$.
- **Decoder:** Map the features $h \in \mathbb{R}^d$ to closely reproduce the input, $\hat{x} \in \mathbb{R}^n$.
- Thus, the autoencoder implements the following problem:

Let $\mathbf{x} \in \mathbb{R}^n$, $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^d$ and $g(\cdot) : \mathbb{R}^d \to \mathbb{R}^n$. Let

$$\hat{\mathbf{x}} = g(f(\mathbf{x}))$$

- Define a loss function, $\mathcal{L}(x, \hat{x})$ and minimize $\mathcal{L}$ with respect to the parameters of $f(\cdot)$ and $g(\cdot)$.
- There are different loss functions that you could consider, but a common one is the squared loss:

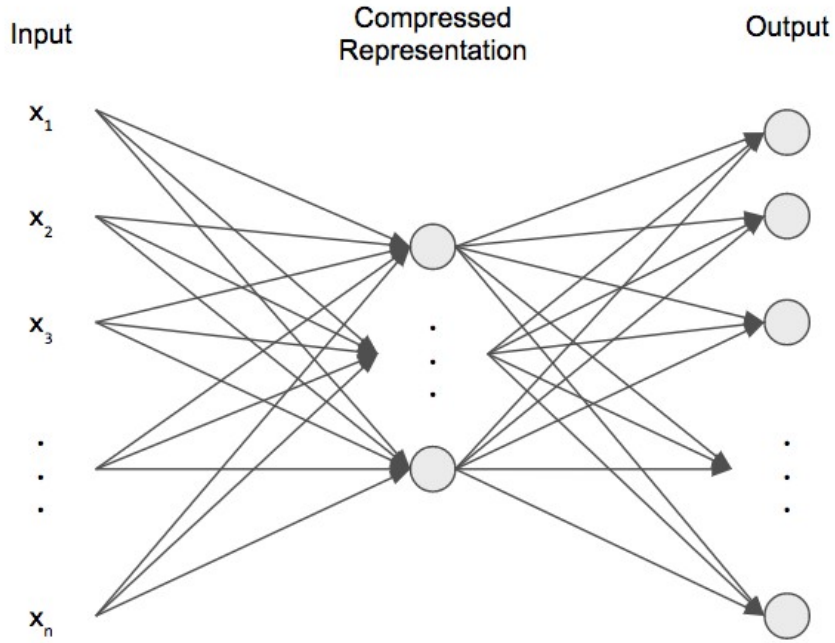
$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

Variant of Autoencoders

- Undercomplete Autoencoders
- Regularized Autoencoders
  - Sparse Autoencoders
  - Denoising Autoencoders
- Variational Autoencoders
- Many others…

Undercomplete Autoencoders

- An autoencoder whose compressed dimension is less than the input dimension is called undercomplete
- Learning an undercomplete representation forces the autoencoder to capture the most salient features of the training data
- The learning process is described simply as minimizing a loss function



Undercomplete Autoencoders

- Undercomplete autoencoder do the dimensionality reduction in some sense as compared to Principal Component Analysis (PCA).

- PCA is a dimensionality reduction technique to present the important features of the input spaces.

- PCA vs. Undercomplete autoencoders
  - Autoencoders are much flexible than PCA.
  - Neural Network activation functions introduce "**non-linearities**" in encoding, but PCA **only** linear transformation.

Regularized Autoencoders

- Regularized autoencoders use a loss function that encourages the model to have other properties besides the ability to copy its input to its output.

- Types of regularized autoencoder (In Practice)
  - Sparse autoencoder
  - Denoising autoencoder.

Sparse Autoencoder

- Autoencoders normally discover useful structures by having a small number of hidden units

- An architecture can have a large number of hidden units
  - By doing so, the autoencoder enlarges the given input's representation

- Typically, It is used to learn features for another task, such as classification

Sparse Autoencoder – Learning

- It impose a constraint in its loss by adding a regularization term in the loss function.

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda \sum_i |h_i|$$

- **Regularization Form**
  - It can be L1 regularization
  - Any other kinds of penalties are possible

For detailed Information:
https://arxiv.org/pdf/1505.05561.pdf

Denoising Autoencoders

- **Idea:** a special autoencoder that is robust to noise.

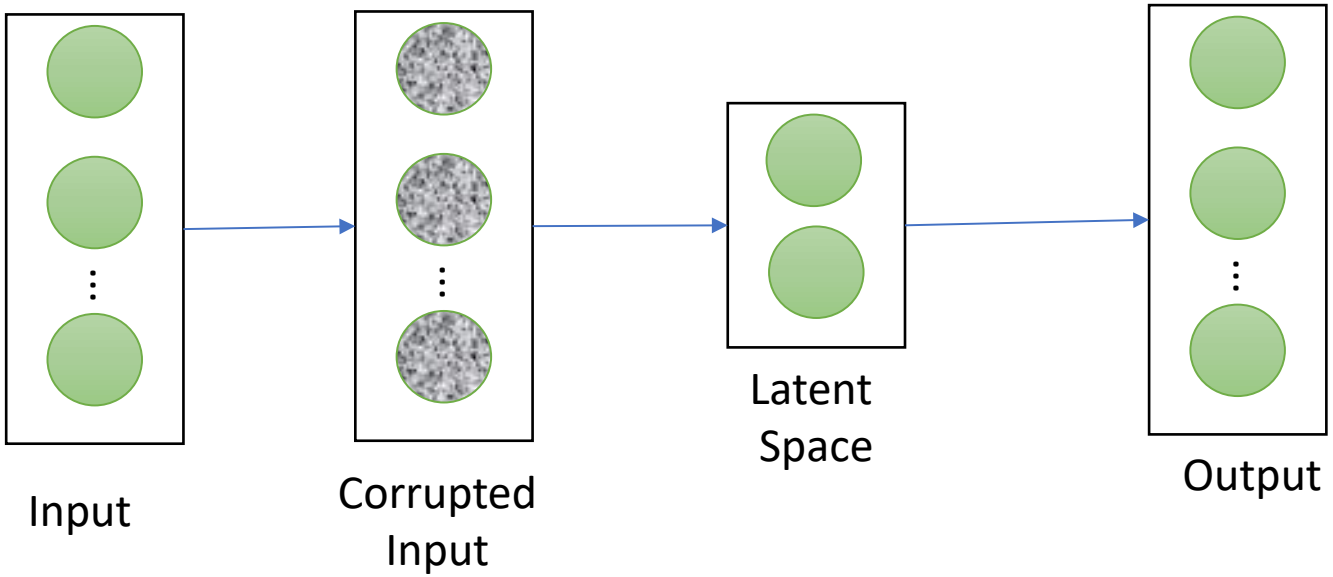
- After giving the autoencoder the corrupted data, we force the hidden layer to learn only the more robust features, rather than just an identity.

- One could generate noise, ε, and add it to the input x, so that $\tilde{x} = x + \varepsilon$. Then, the loss function

$$\mathcal{L}(\mathbf{x}, g(f(\tilde{\mathbf{x}})))$$

Denoising Autoencoders

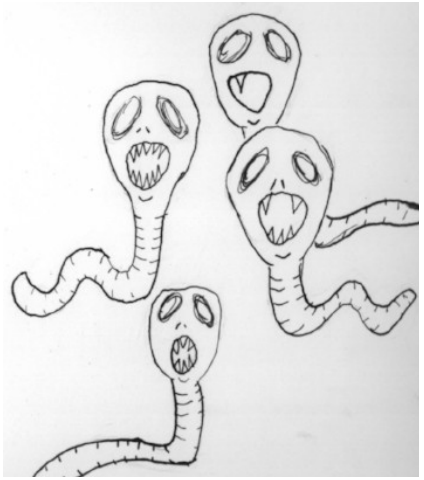
- By adding stochastic noise, it can force Autoencoder to learn more robust features.



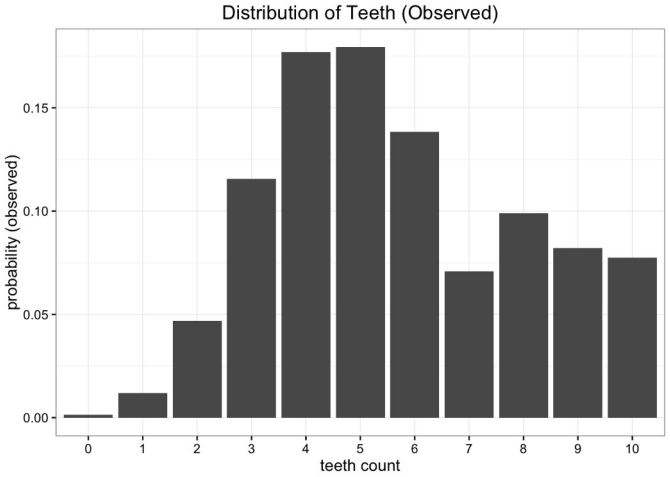
Variational Autoencoder – Preliminary

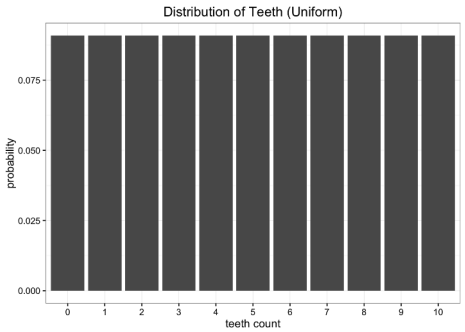
1. Kullback-Leibler Divergence (KL Divergence)
and
2. Generative Models

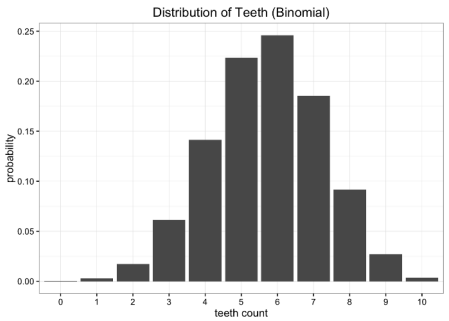
Kullback-Leibler Divergence (KL Divergence)

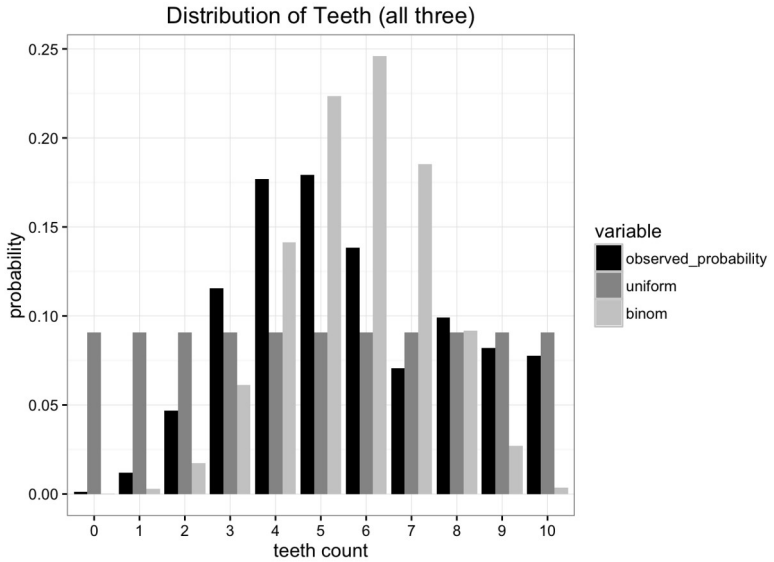


- We are at new planet, and we've discovered a species of biting worms that we'd like to study
- We've found that these worms have 10 teeth, but because of all the chomping away, many of them end up missing teeth
- While this data is great, we have a bit of a problem. We're far from Earth and sending data back home is expensive. What we want to do is reduce this data to a simple model with just one or two parameters





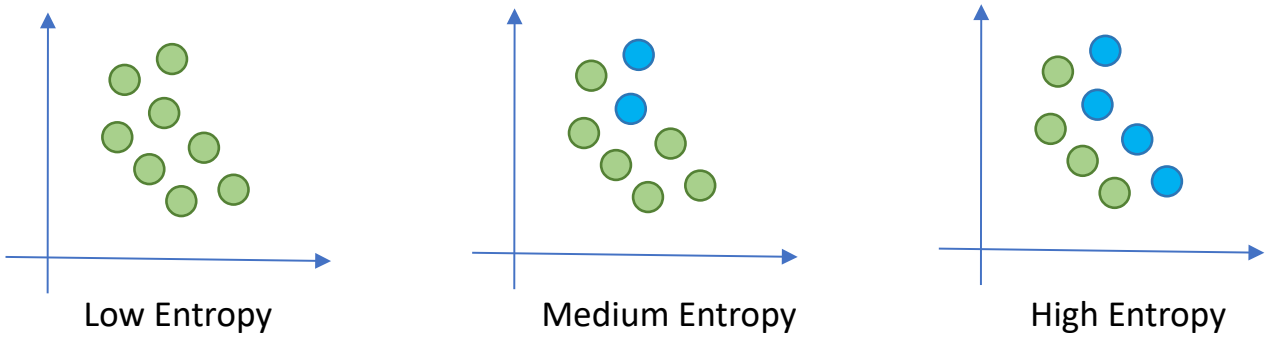




- The best test of which is better is to ask which distribution preserves the most information from our original data source. This is where Kullback-Leibler Divergence comes in.

Entropy

- Entropy measures the uncertainty



$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Note: Original Paper: “A Mathematical Theory of Communication” by Claude Shannon (1948)

It is one of the most successful and impactful mathematical theories known

KL Divergence

- KL Divergence has its origins in information theory.

$$H = -\sum_{i=1}^{N} p(x_i) \cdot \log p(x_i)$$

- KL Divergence is just a slightly modification of our formula for entropy. Rather than just having our probability distribution p we add in our approximating distribution q. Then we look at the difference of the log values for each:

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot (\log p(x_i) - \log q(x_i)) \qquad D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot log \frac{p(x_i)}{q(x_i)}$$

- Essentially, what we're looking at with the KL divergence is the expectation of the log difference between the probability of data in the original distribution with the approximating distribution.

$$D_{KL}(p||q) = E[\log p(x) - \log q(x)]$$

$$D_{kl}(\text{Observed} \,||\, \text{Uniform}) = 0.338 \qquad D_{kl}(\text{Observed} \,||\, \text{Binomial}) = 0.477$$

Source: https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained

KL Divergence Properties

- In general, KL divergence is not commutative

$$KL(q||p) \neq KL(p||q)$$

- $KL(q||p) \geq 0$ for any q, p.

- $KL(q||p) = 0$ if and only if $q(z) = p(z)$ for all z,
  (i.e., q and p are the same distribution)

Variational Autoencoder – Generative Models

- Variational autoencoders are in a class of models called generative models.

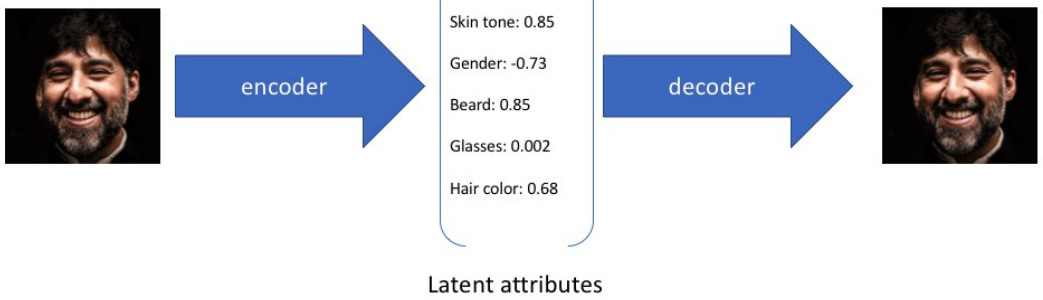
- These models do exactly what their name suggests: they can be used to generate examples of input data by learning their statistics.

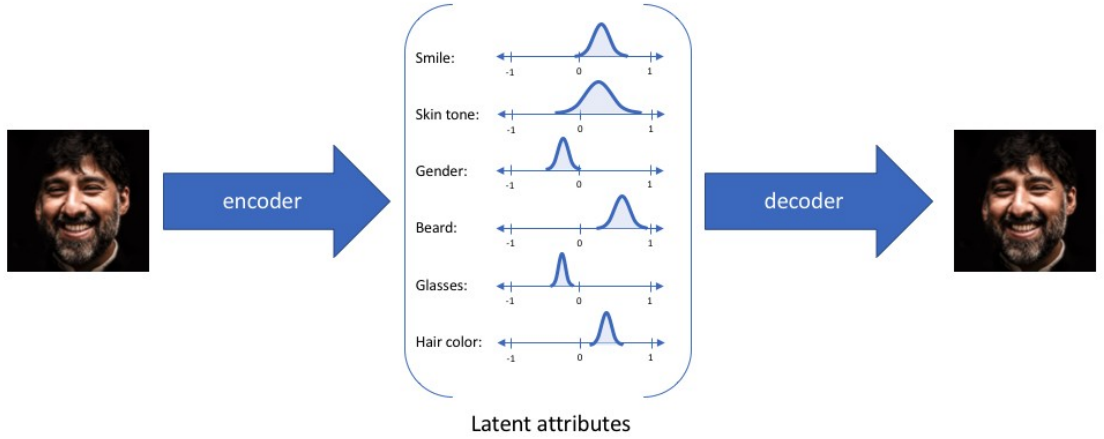
Variational Autoencoder – How they differ?

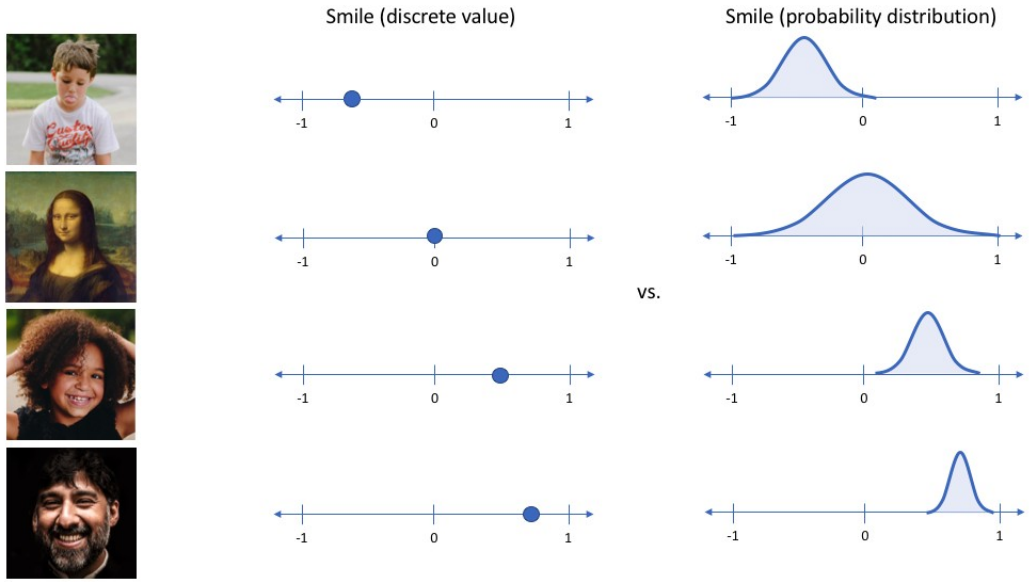
- A variational autoencoder resembles a classical autoencoder and is a neural network consisting of an encoder, a decoder and a loss function.
- The compressed representation is a probability distribution.
- They let us design generative models of data

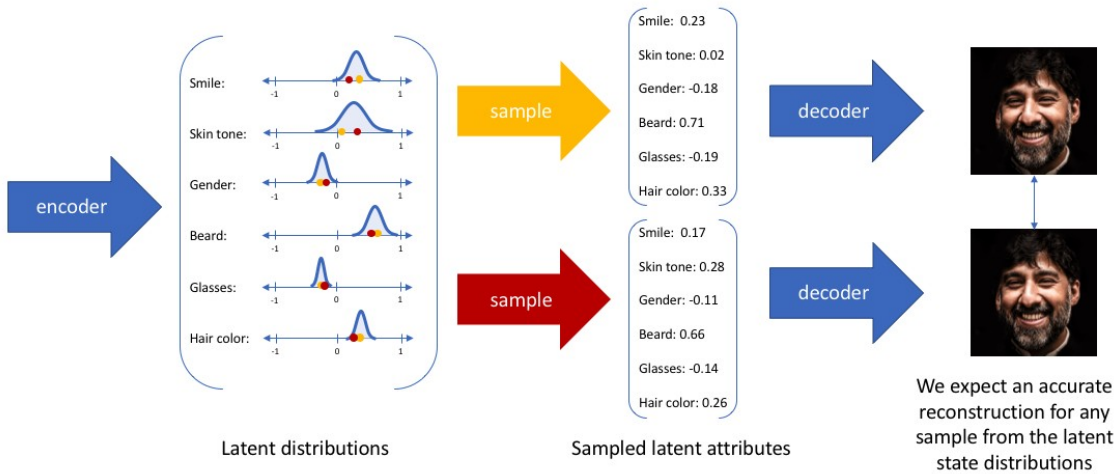
Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

VAE – Intuition









Source: https://www.jeremyjordan.me/variational-autoencoders/

Variational Autoencoder

- **Recognition Model**
  - The encoder model is sometimes referred as a recognition model

- **Generative Model**
  - The decoder model is sometimes referred as generative model

Variational Autoencoder

- By constructing our encoder model to output a range of possible values (a statistical distribution) from which we'll randomly sample to feed into our decoder model

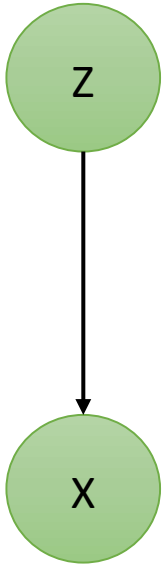
- For any sampling of the latent distributions, we're expecting our decoder model to be able to accurately reconstruct the input.

- Thus, values which are nearby to one another in latent space should correspond with very similar reconstructions.

VAE – Simple Understanding

Variational Autoencoder

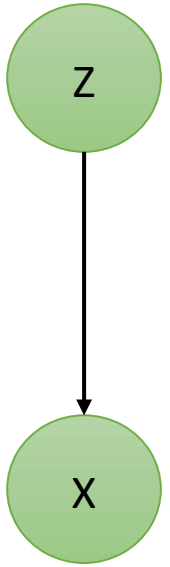
- Suppose that there exists some hidden variable z which generates an observation x



- We can only see x, but we would like to infer the characteristics of z.

Variational Autoencoder

- In other words, we'd like to compute p(z|x).



$$p(z|x) = \frac{p(x|z)\,p(z)}{p(x)}$$

- Unfortunately, computing p(x) is quite difficult.

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

- This usually turns out to be an intractable distribution

Variational Autoencoder

- **Solution**
  - Variational inference to estimate this value.
  - Monte Carlo approach – simulation based method.

Variational Inference

- Let's approximate $p(z|x)$ by another distribution $q(z|x)$ which we'll define such that it has a tractable distribution.

- If we can define the parameters of $q(z|x)$ such that it is very similar to $p(z|x)$, we can use it to perform approximate inference of the intractable distribution.

Variational Autoencoder

- KL divergence is a measure of difference between two probability distributions.

- Thus, if we wanted to ensure that q(z|x) was similar to p(z|x), we could minimize the KL divergence between the two distributions.

$$\min KL\left(q\left(z|x\right)||p\left(z|x\right)\right)$$

- We can minimize the above expression:

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

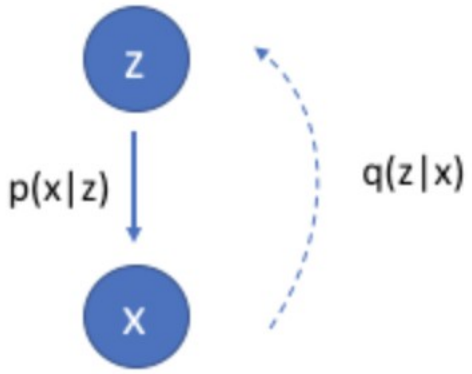
Variational Autoencoder

$$E_{q(z|x)} \log p\,(x|z) - KL\,(q\,(z|x)\,||p\,(z))$$

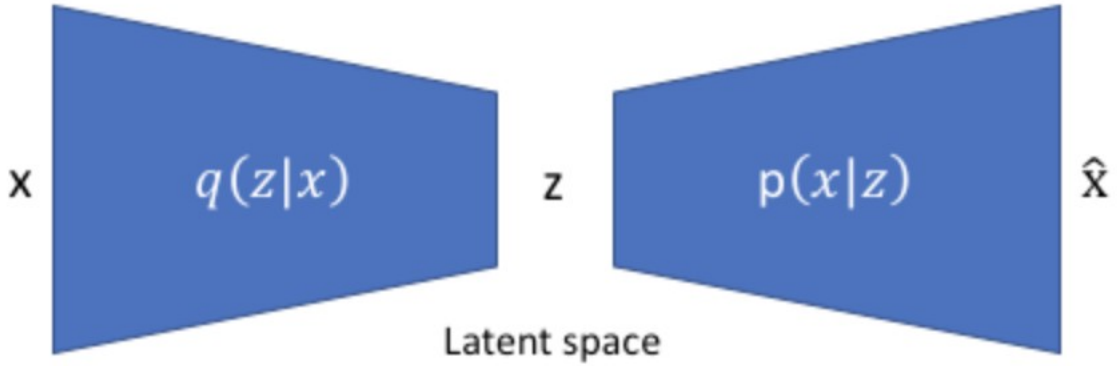
- First term represents the reconstruction likelihood
- Second term ensures that our learned distribution q is similar to the true prior distribution p.

Variational Autoencoder

- We can further construct this model into a neural network architecture
- Where, the encoder model learns a mapping from x to z and the decoder model learns a mapping from z back to x.



We'd like to use our observations to understand the hidden variable.



Latent space representation.

Neural network mapping x to z.

Neural network mapping z to x.

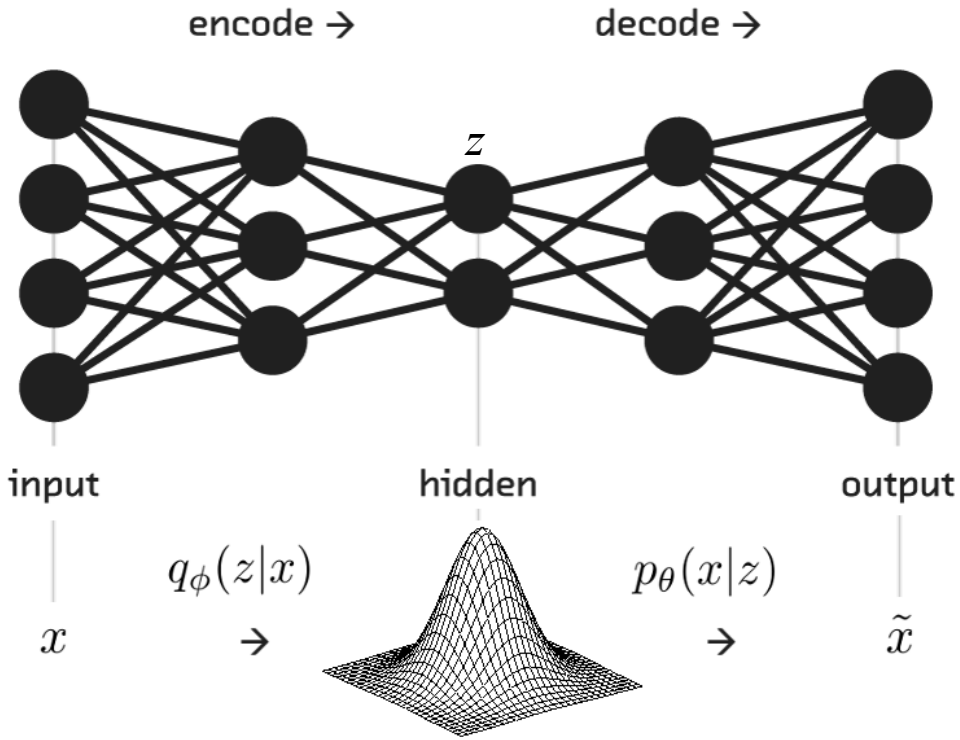
Variational Autoencoder

- Our loss function for this network will consist of two terms:
  - First penalizes reconstruction error
  - Second term which encourages our learned distribution $q(z|x)$ to be similar to the true prior distribution $p(z)$

  - **Assumption**
    - A unit Gaussian distribution, for each dimension j of the latent space.

VAE – Derivation

Variational Autoencoder

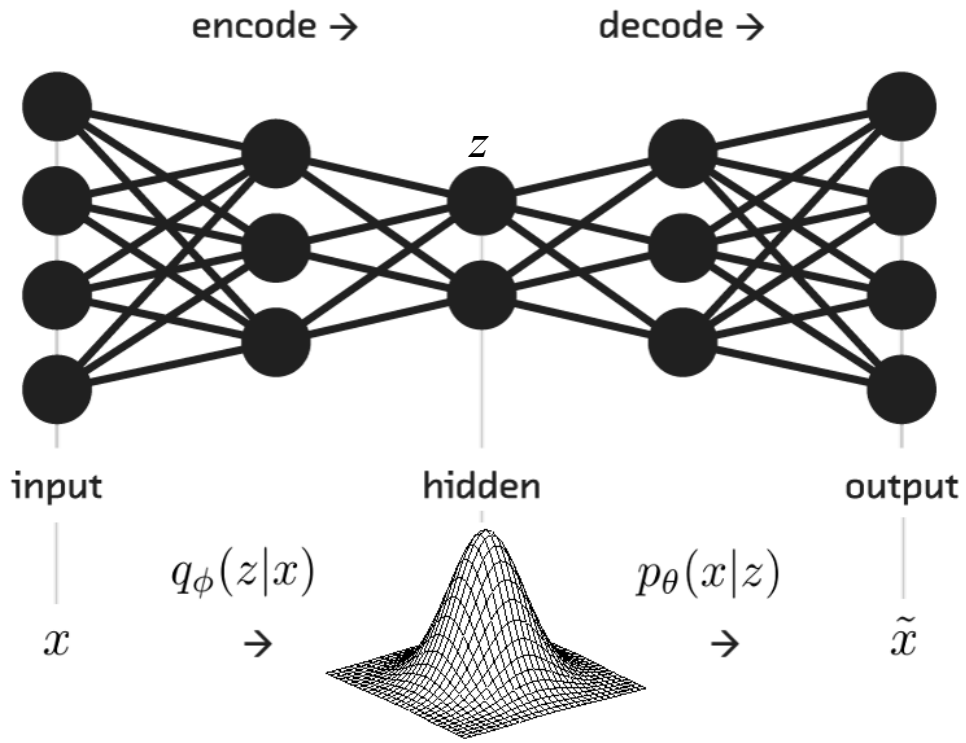


With this Bayesian perspective, the encoder becomes a variational inference network, mapping observed inputs to (approximate) posterior distributions over latent space, and the decoder becomes a generative network, capable of mapping arbitrary latent coordinates back to distributions over the original data space

Source: https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in.html

VAE Goal!!

- The goal of VAE is to find a distribution $q_\phi(z|x)$ of some latent variables, which we can sample from $z \sim q_\phi(z|x)$, to generate new samples $\tilde{x}$ from $p_\theta(x|z)$



Decoder – Generator

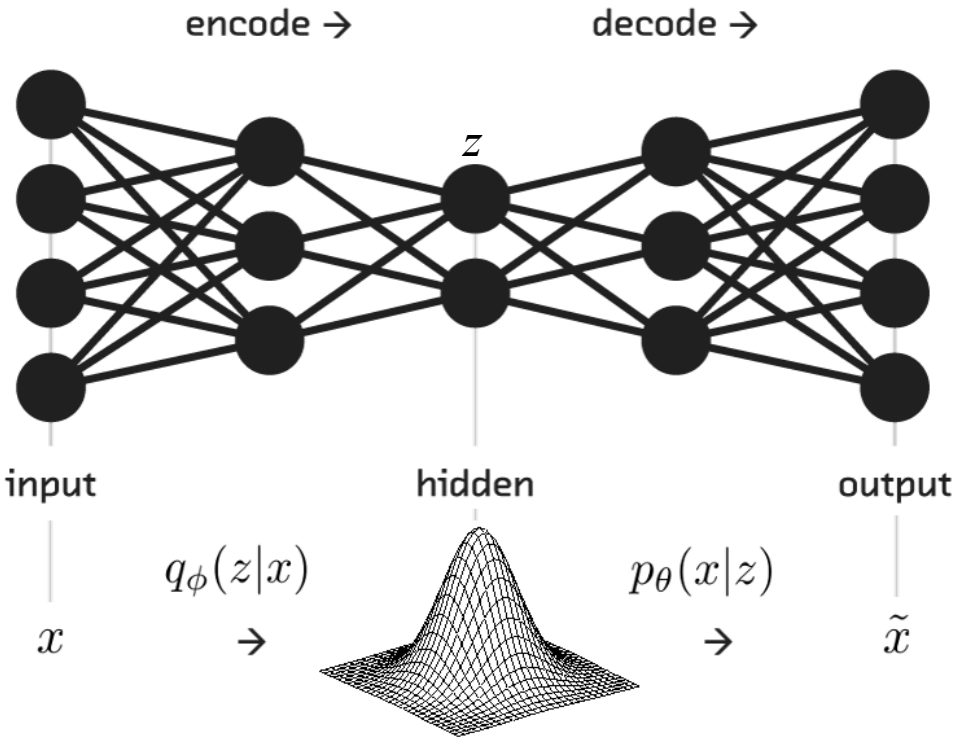
- Probabilistic spin on autoencoders - will let us sample from the model to generate the data.
- How should we represent our model?
  - Choose prior p(z) to be simple. i.e., Gaussian (Reasonable)
  - Conditional p(x|z) is complex due to generation process (Neural network)
- How to train the model?
  - Learn model parameters to maximize the likelihood of data

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

Intractable to compute p(x|z) for every z

Decoder NN

Simple Gaussian Prior



It is also strategy for training generative models from Fully Visible Belief Nets — pixel CNN or pixel RNN

Reason!!

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

- This integral is not available in closed form or is intractable (i.e., requires exponential time to compute) due to multiple integrals involved for latent variable vector z.

We can't directly compute it

Another prospective – Posterior density

- We can only see x, but we would like to infer the characteristics of z.
- In other words, we'd like to compute $p(z|x)$.

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

- Unfortunately, computing $p(x)$ is quite difficult.

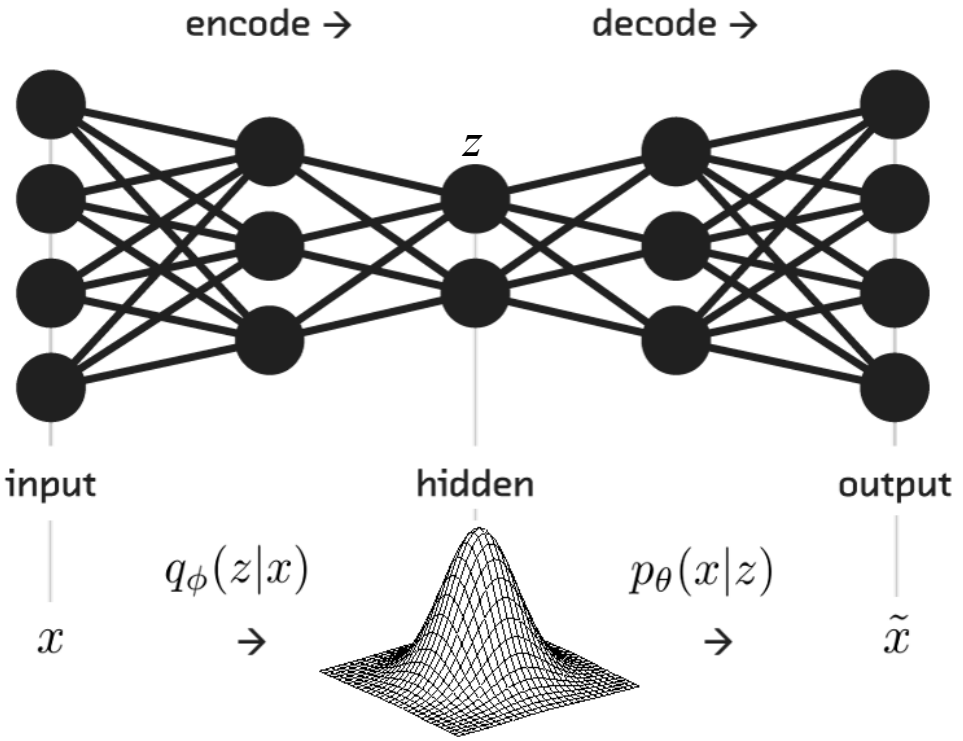
$$p(x) = \int p(x|z)p(z)dz$$

Alternative Solution

- The alternative is to approximate $p(z|x)$ by another distribution $q(z|x)$ (i.e., encoder network) which is defined in such a way that it has tractable solution.

- This is done using variational inference.

Will see that this allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize.



Variational Inference

- Variational inference is to pose the inference problem as an optimization problem.

- **How?**
  - By approximating $p(z|x)$ using $q(z|x)$ – where $q(z|x)$ has a simple distribution such as Gaussian.

- How does this distribution relate?
  - Let us calculate KL divergence between $p(z|x)$ and $q(z|x)$.

KL Divergence

$$D_{KL}(q(z|x)||p(z|x) = \sum_{z} q(z|x)\ log\left(\frac{q(z|x)}{p(z|x)}\right)$$

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

$$= E_{z\sim q(z|x)}[logq(z|x) - \log p(z|x)]$$ (Logarithms)

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

Apply the Bayes rule here!!

KL Divergence

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

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

We can $\log \frac{p(x|z)p(z)}{p(x)} = \log p(x|z) + \log p(z) - \log p(x)$

$$D_{KL}(q(z|x)||p(z|x) = E_{z\sim q(z|x)}[\log q(z|x) - \log p(x|z) - \log p(z) + \log p(x)]$$

Since, the expectation is over z and $p(x)$ doesn't involve so we can move it out.

$$D_{KL}(q(z|x)||p(z|x) = E_{z\sim q(z|x)}[\log q(z|x) - \log p(x|z) - \log p(z)] + \log p(x)$$

KL Divergence

z: $z \sim q(z|x)$

$$D_{KL}(q(z|x)||p(z|x) = E_{z\sim q(z|x)}[\log q(z|x) - \log p(x|z) - \log p(z)] + \log p(x)$$

$$= E_z[\log q(z|x) - \log p(x|z) - \log p(z)] + \log p(x)$$

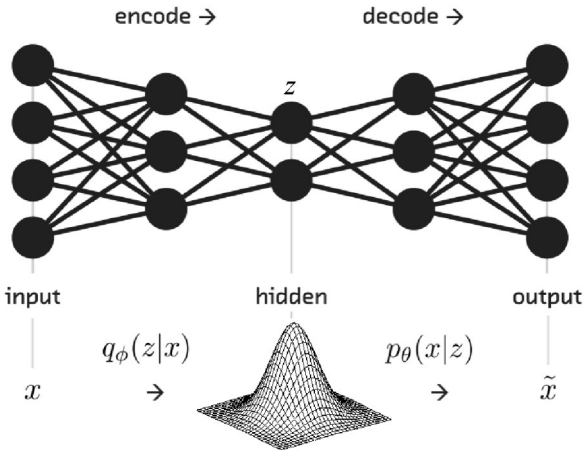
$$= -E_z\ [\log p(x|z)] + E_z[\log q(z|x) - \log p(z)] + \log p(x)$$

$$D_{KL}(q(z|x)||p(z|x) = -E_z[\log p(x|z)] + D_{KL}(q(z|x)||p(z)) + \log p(x)$$

$$E_z[\log p(x|z)] - D_{KL}(q(z|x)||p(z)) + D_{KL}(q(z|x)||p(z|x) = \log p(x)$$

Model parameters to Max. Likelihood

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



We start form above expression (data likelihood) and derived the following one!!

$$\log p(x) = E_z[\log p(x|z)] - D_{KL}(q(z|x)||p(z)) + D_{KL}(q(z|x)||p(z|x))$$

Decoder Network: We can calculate this term

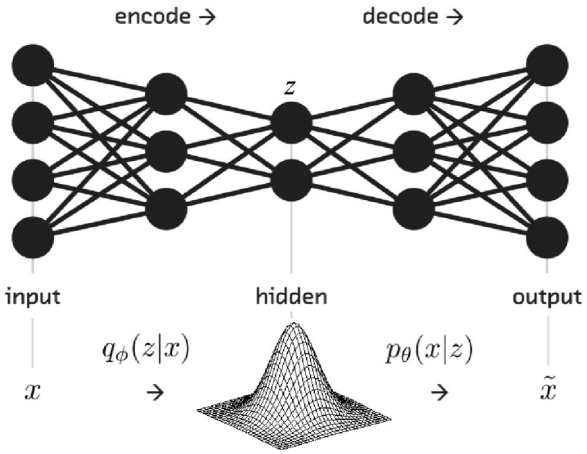
This KL term (between Gaussians for encoder and z prior) has nice close-form solution

$p(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

- First term represents the reconstruction likelihood
- Second term ensures that our learned distribution q is similar to the true prior distribution p.

Loss Function – ELBO

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



We start form above expression (data likelihood) and derived the following one!!

$$\log p(x) = E_z[\log p(x|z)] - D_{KL}(q(z|x)||p(z)) + D_{KL}(q(z|x)||p(z|x)$$

Tractable lower bound which we can take gradient of and optimize ($p(x|z)$ and KL term both are differentiable)

≥ 0

$$\mathcal{L}(x, \theta, \emptyset)$$

$$\log p(x) \geq \mathcal{L}(x, \theta, \emptyset)$$

Also known as Evidence Lower Bound Objective (ELBO)

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Loss Function – ELBO

$$\log p(x) = E_z[\log p(x|z)] - D_{KL}(q(z|x)||p(z)) + D_{KL}(q(z|x)||p(z|x)$$

$$\log p(x) \geq \mathcal{L}(x, \theta, \emptyset)$$

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

- This is great news, because we now have a loss function (the lower bound, $\mathcal{L}(x, \theta, \emptyset)$) that is tractable in the following manner:
  - The term $E_z[\log p(x|z)]$ can be approximated from the data using a minibatch, almost exactly like we do for the softmax loss.
  - We take some minibatch of m examples $x^{(1)}, \ldots, x^{(m)}$, and for each $x^{(i)}$, we calculate $q(z|x^{(i)})$, sample z from it, and then calculate $\log p(x^{(i)}|z)$.

KL Divergence – Closed Form

- The KL divergence has a closed form when q() and p() are Gaussian distributions. That is,

$$\mathrm{KL}(\mathcal{N}(\mu_0, \Sigma_0), \mathcal{N}(\mu_1, \Sigma_1)) = \frac{1}{2}\left[\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1}(\mu_1 - \mu_0) - d + \log \frac{\det \Sigma_1}{\det \Sigma_0}\right]$$

Training the Model

- Backpropagation works fine!!

- However, we simply cannot do this for a random sampling process.

$$z \sim q(z|x^{(i)})$$

- **Reparameterization trick:**
  - It is basically diverting the non-differentiable operation out of the network
  - So that, even though we still involve a thing that is non-differentiable, at least it is out of the network
  - Hence the network could still be trained.

Reparameterization Trick

- To do so, we sample $\varepsilon \sim N(0, 1)$ and calculate:

$$\mathbf{z} = \mu_\phi(\mathbf{x}^{(i)}) + \Sigma_\phi^{1/2}(\mathbf{x}^{(i)})\varepsilon$$

- Then, z will be a sample from $q(z|x^{(i)})$ as its a linear transformation of ε with mean $\mu_\emptyset(x^{(i)})$ and covariance $\Sigma_\emptyset(x^{(i)})$.

- The sampling operation now occurs only for ε, which we don't need to backpropagate through.

Reparameterization Trick

- If we sample z from a standard normal distribution, we could convert it to any Gaussian we want if we know the mean and the variance

- Hence, we could implement our sampling operation of z by

$$\mathbf{z} = \mu_\phi(\mathbf{x}^{(i)}) + \Sigma_\phi^{1/2}(\mathbf{x}^{(i)})\varepsilon$$

where ϵ~N(0,1)

- Then, z will be a sample from $q(z|x^{(i)})$ as its a linear transformation of ε with mean $\mu_\emptyset(x^{(i)})$ and covariance $\Sigma_\emptyset(x^{(i)})$.

- To do so, we sample $\varepsilon \sim N\ (0, 1)$

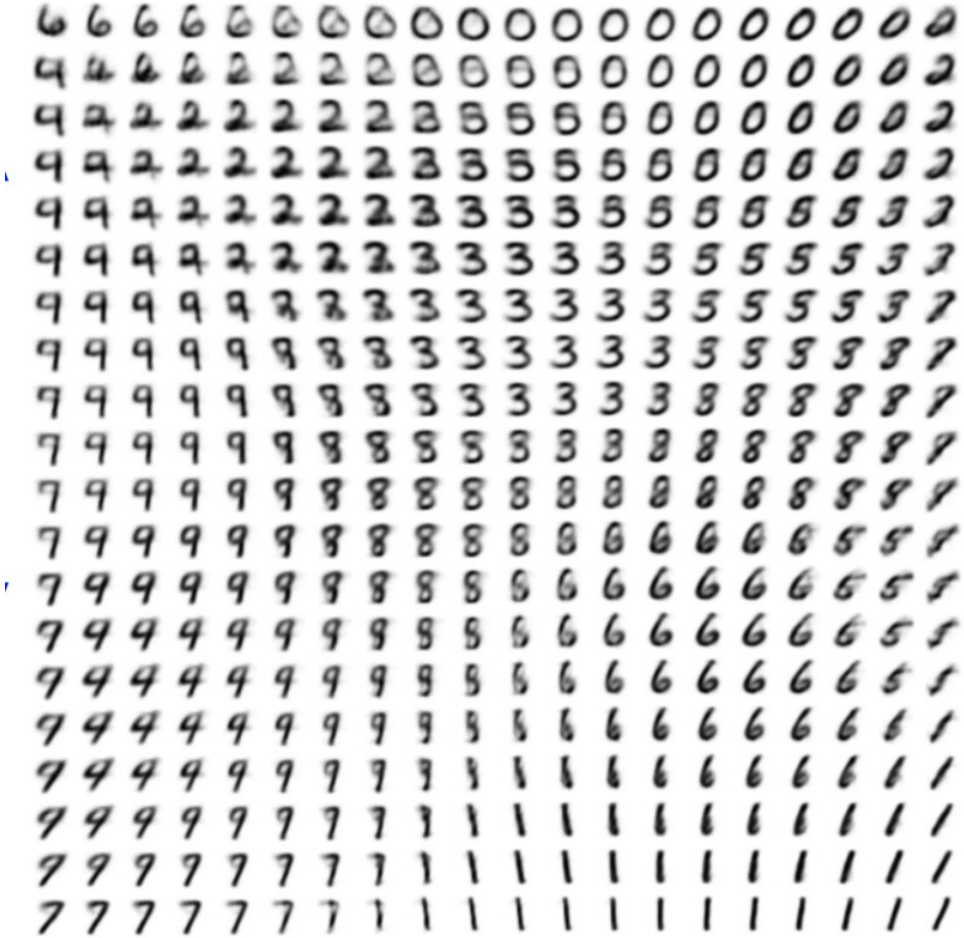
Reparameterization Trick

- Now, during backpropagation, we don't care anymore with the sampling process, as it is now outside of the network

- That means it doesn't depend on anything in the net, hence the gradient won't flow through it.

Variational Autoencoders: Generating Data!

- MNIST Dataset



Caveats about the VAE

- There are some caveats about the VAE that we need to be aware of.
  - Inference used a lower bound rather than the likelihood of the data. Empirically, it has been observed that there might still be a large gap between the maximum-likelihood and the bound.
  - There currently remains no proof that VAEs are asymptotically consistent.
  - Subjectively, VAEs generate images but other generative models do better.

Summary

- Autoencoders
- Undercomplete autoencoders
- Regularized autoencoders
  - Sparse autoencoders
  - Denoising autoencoders
- Variational autoencoders

Thank You ☺

Appendix

Why KL divergence is non-negative?

$$
\begin{aligned}
-D(p||q) &= -\sum_x p(x) \log_2 \frac{p(x)}{q(x)} \\
&= \sum_x p(x) \log_2 \frac{q(x)}{p(x)} \\
&\overset{(c)}{\leq} \log_2 \sum_x p(x) \frac{q(x)}{p(x)} \\
&= \log_2 1 \\
&= 0
\end{aligned}
$$

- Where at (c) we have used Jensen's inequality and the fact that log is a concave function.

Why KL divergence is non-negative?

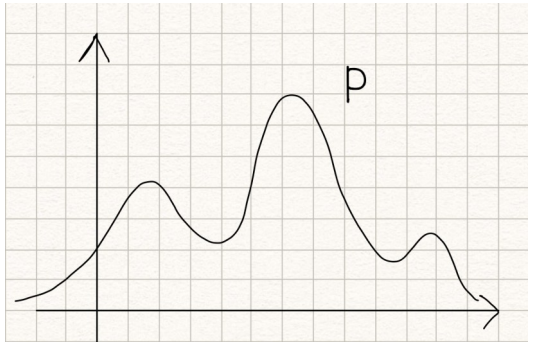
- The KL divergence is non-negative. An intuitive proof is that:
- if *P*=*Q*, the KL divergence is zero as:

$$\log \frac{P}{Q} = \log 1 = 0$$

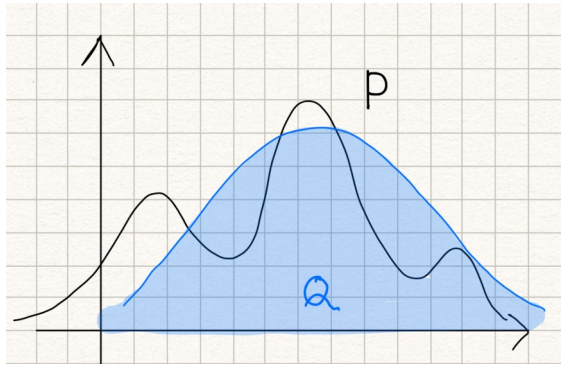
The KL divergence is asymmetric

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

- It can be deduced from the fact that the cross-entropy itself is asymmetric. The cross-entropy *H(P, Q)* uses the probability distribution *P* to calculate the expectation. The cross-entropy *H(Q, P)* uses the probability distribution *Q* to calculate the expectation.
- So, the KL divergence cannot be a distance measure as a distance measure should be symmetric.
- Suppose we have a probability distribution *P* which looks like below:

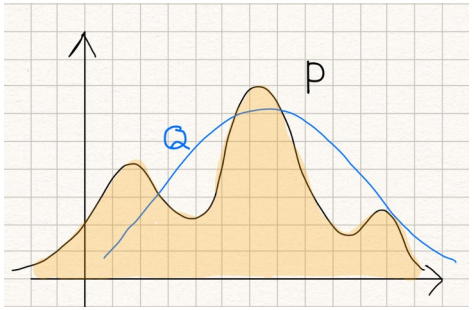


we want to approximate it with a normal distribution Q as:



$$D_{KL}(P||Q) = \mathbb{E}_{x\sim P}[\log \frac{P(x)}{Q(x)}]$$

If we swap P and Q, it means that we use the probability distribution P to approximate the normal distribution Q, and it'd look like



$$D_{KL}(Q||P) = \mathbb{E}_{x\sim Q}[\log \frac{Q(x)}{P(x)}]$$

Both cases measure the similarity between P and Q, but the result could be entirely different, and they are both useful.

Law of Total Probability Theorem

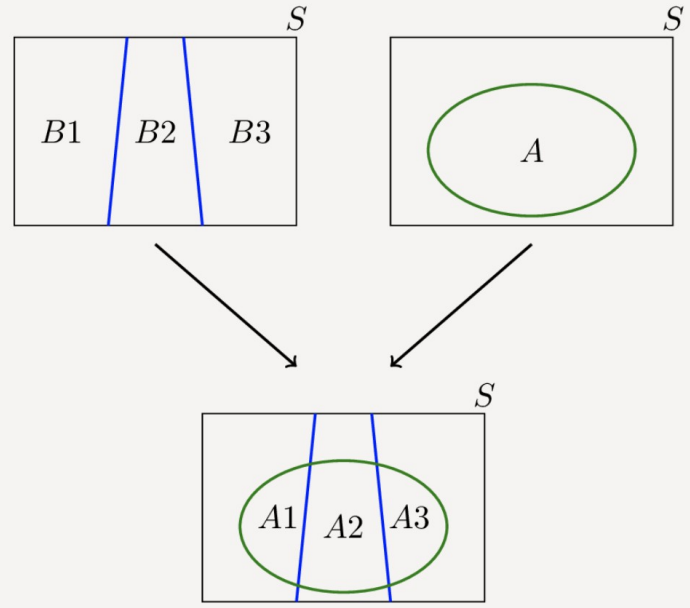
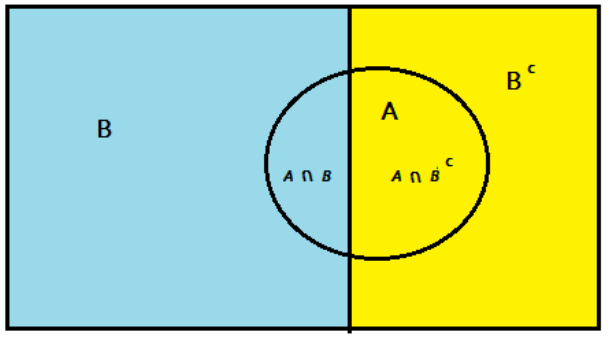


Fig.1.24 - Law of total probability.

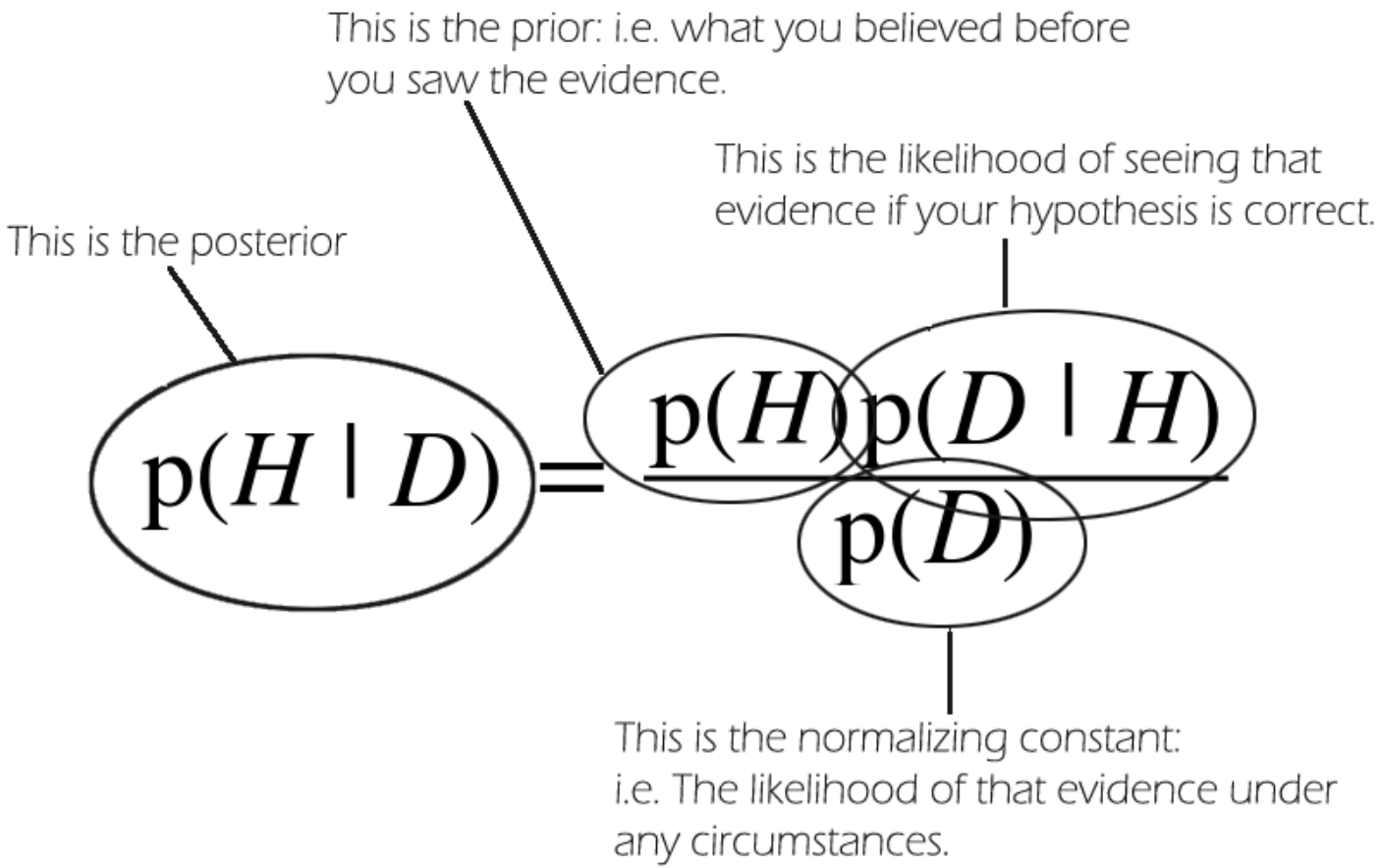
$$P(A) = P(A \cap B) + P(A \cap B^c)$$



Law of Total Probability:
If $B_1, B_2, B_3, \cdots$ is a partition of the sample space S, then for any event A we have

$$P(A) = \sum_i P(A \cap B_i) = \sum_i P(A|B_i)P(B_i).$$

Bayes Theorem



Source: https://medium.com/@akankshamalhotra24/naive-bayes-theorem-79832d506a63