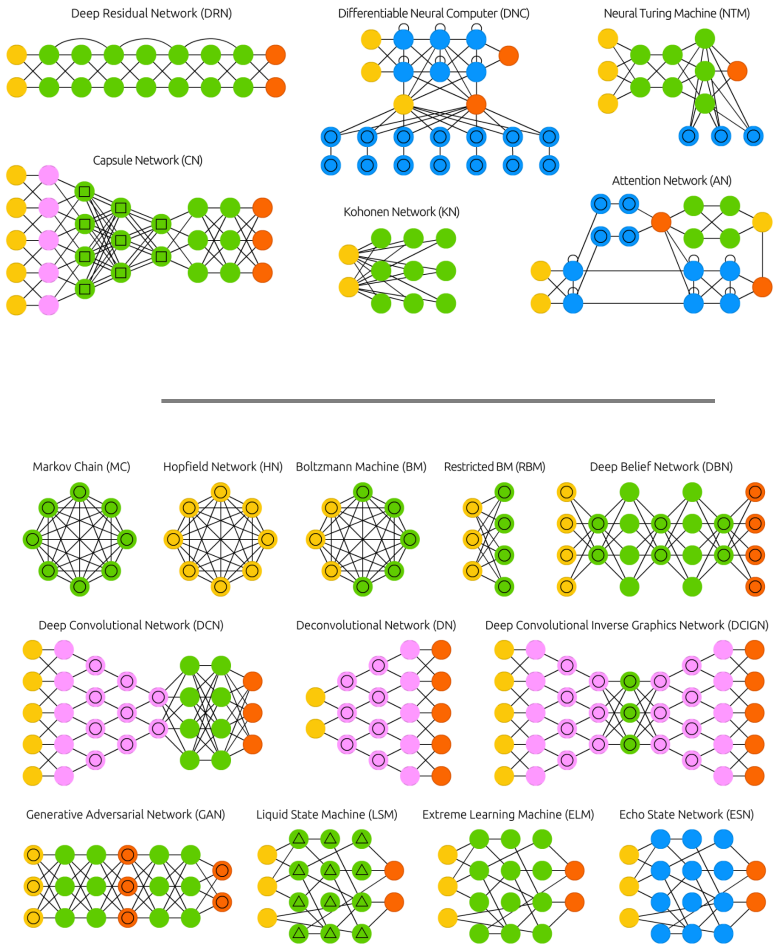
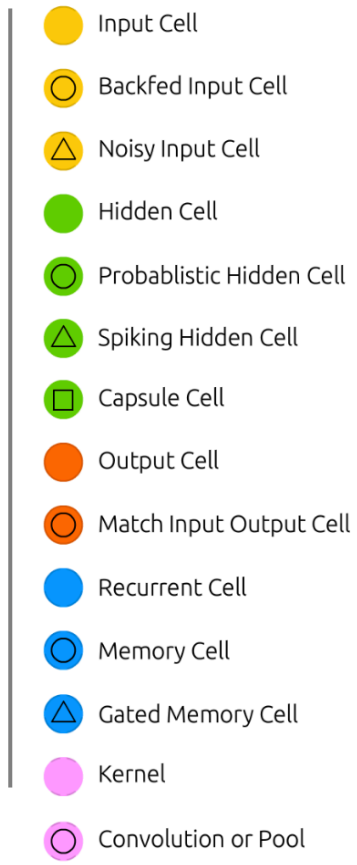
Autoencoders & Variational Autoencoders

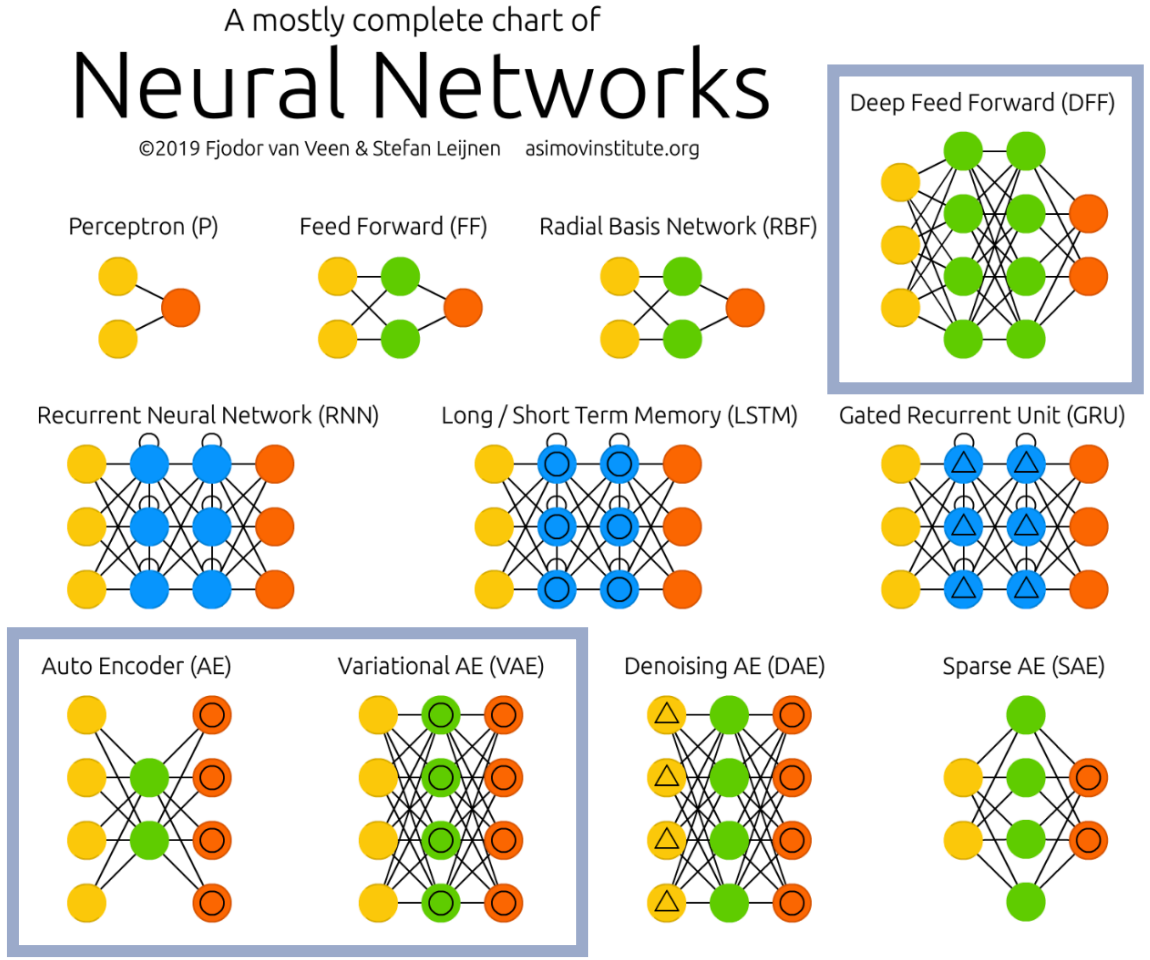
Marie-Eve Malette

Fin ML

Neural Network Architectures





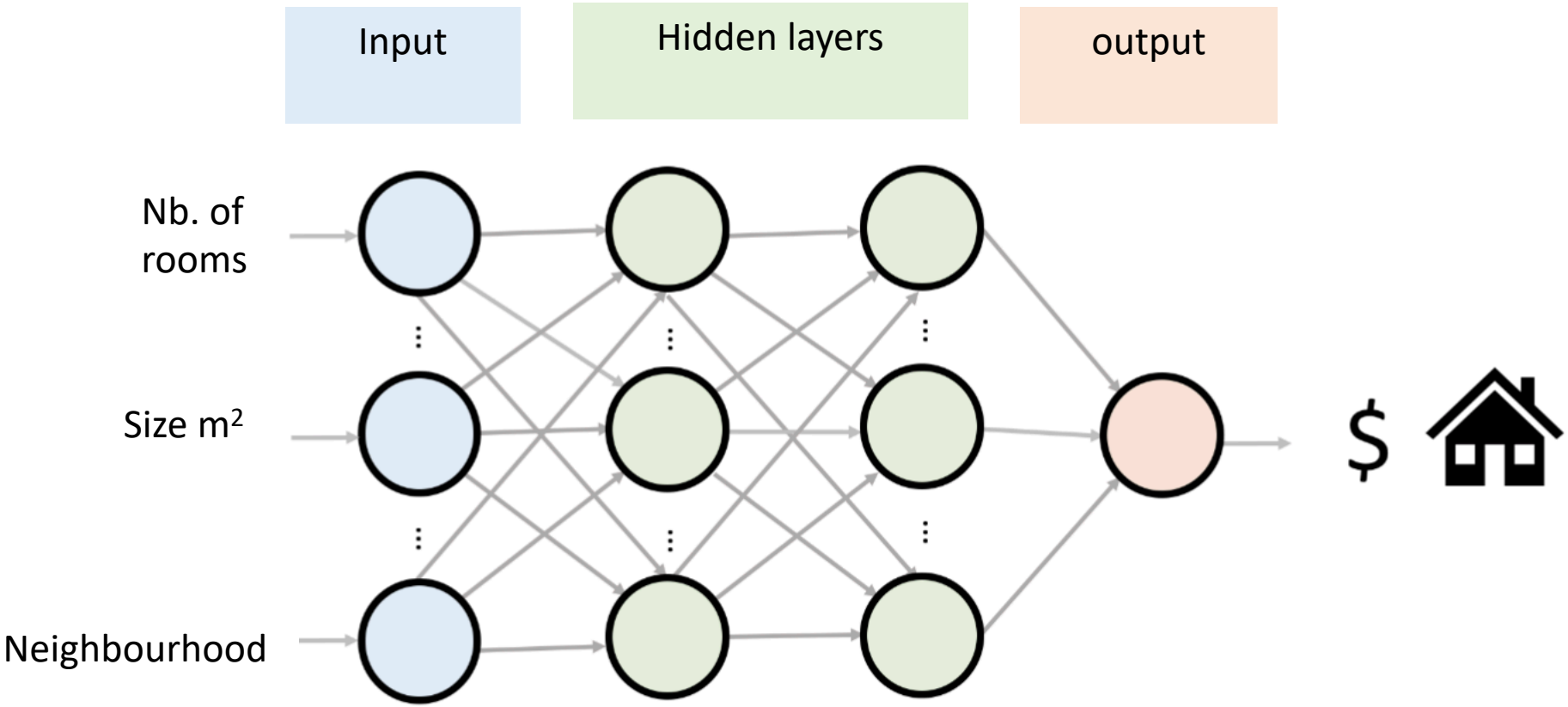


Fin ML

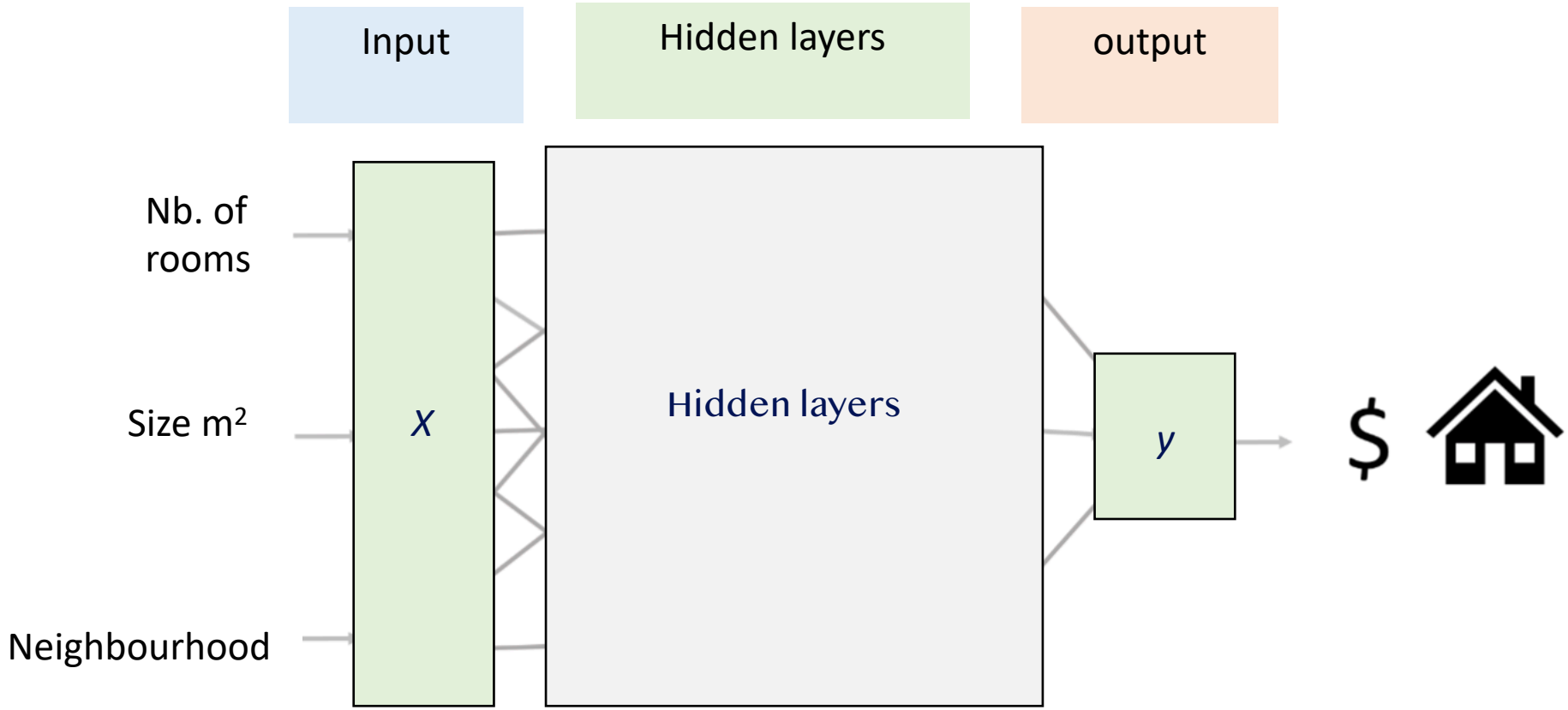
Objectives of this session

1. Autoencoders
2. Variational autoencoder (VAE)
3. Risk evaluation using a conditional variational autoencoder (CVAE)

From Neural Networks to Autoencoders

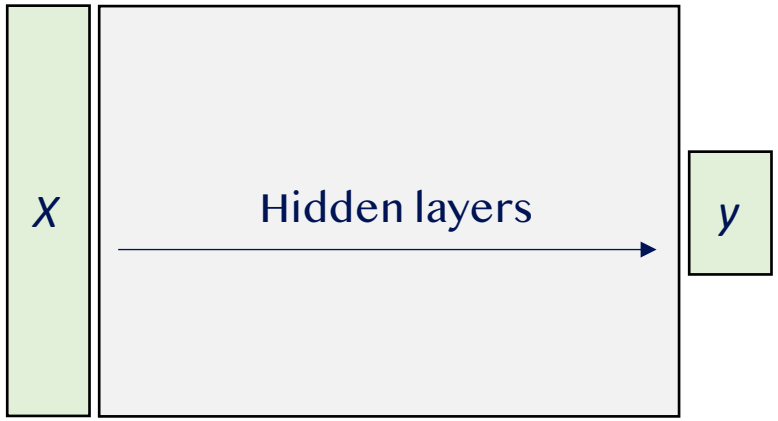


From Neural Networks to Autoencoders



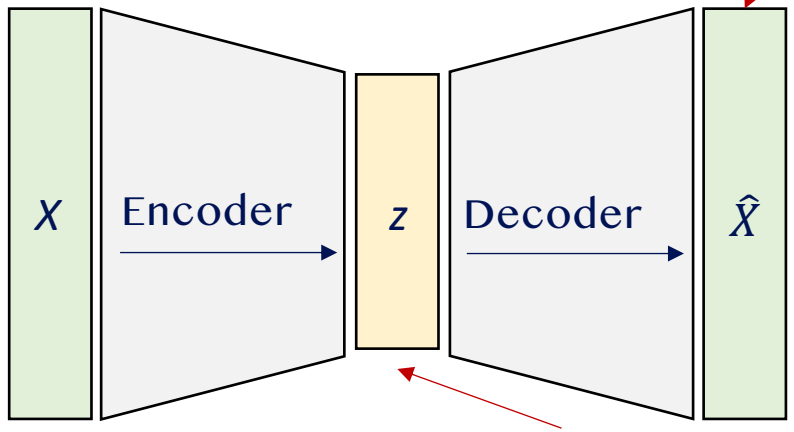
From Neural Networks to Autoencoders

Vanilla Neural Network



Maps $X \rightarrow y$

Autoencoder



Reconstructs the input

Maps $X \rightarrow X$

Hidden layers encodes the input into a latent vector and decodes it into a reconstrucion of the inputs

Fin ML

From Neural Networks to Autoencoders

We don't really care about the output of the network… it is just a replica of the input.

But we do care about the latent representation… why?

Because it contains the same information as the input in lesser dimension. It is a dense represensation of the inputs.

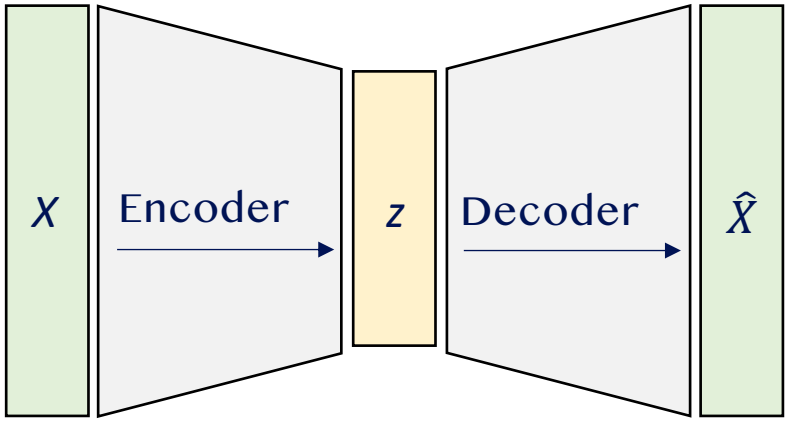
Autoencoders fall into the unsupervised learning category

They only use the inputs X

Useful to leverage unlabeled data

Motivation: Extract meaningful features

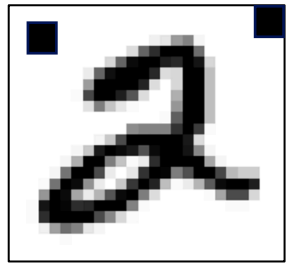
Autoencoder



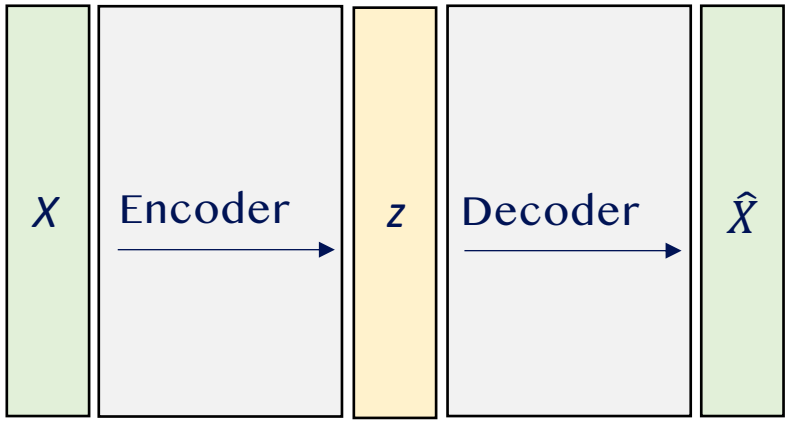
Maps $X \rightarrow X$

From Neural Networks to Autoencoders

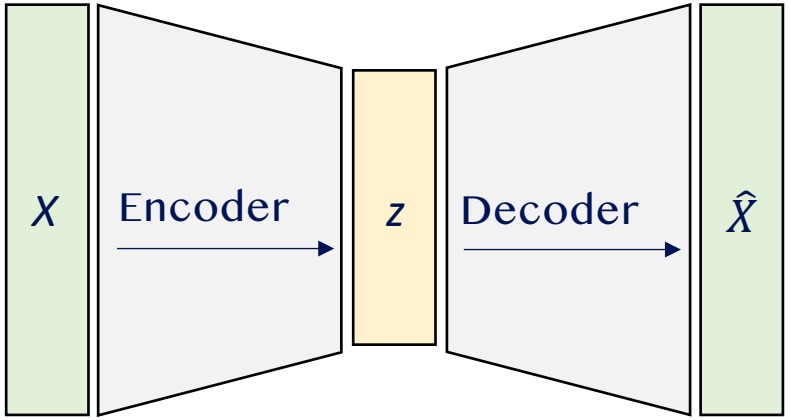
Why the funnel shape?



Flat architecture ???



Autoencoder



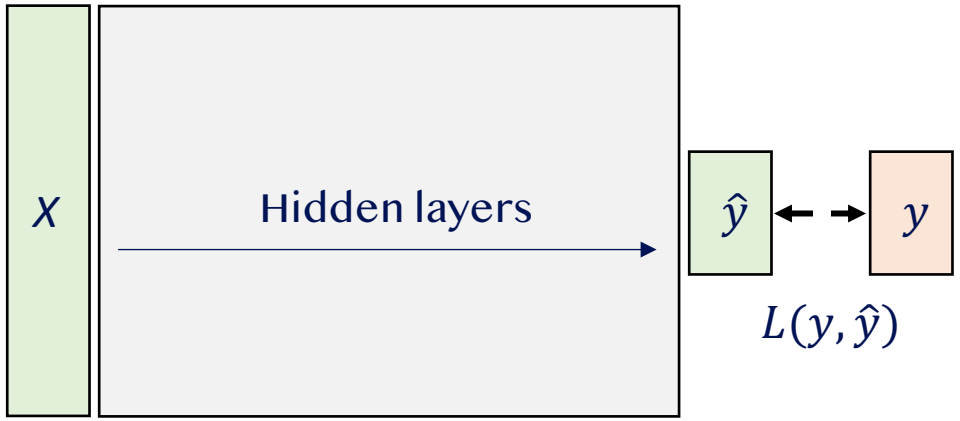
The network would learn to copy the input to the next layer
and would not be encouraged to learn meaningful features

From Neural Networks to Autoencoders

Example: Comparison of loss functions in a Regression problem

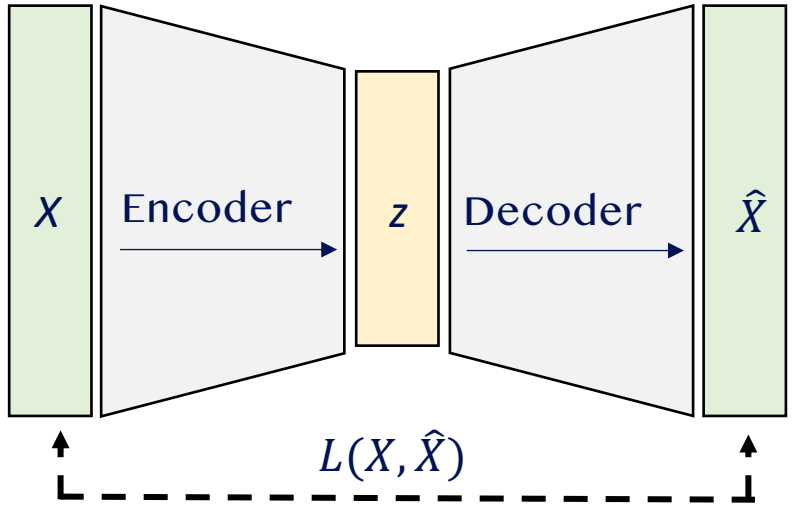
```
# Vanilla neural network
model.fit(X, y)

# Autoencoder
model.fit(X, X)
```

Vanilla Neural Network



$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2$$

Autoencoder



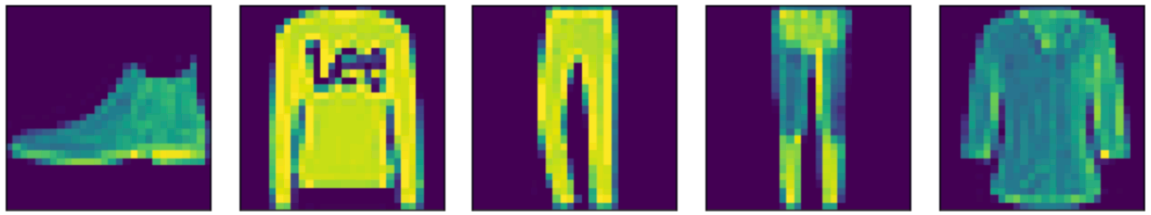
$$L(X, \widehat{X}) = \frac{1}{N}\sum_{i=1}^{N}(X - \widehat{X})^2$$

Fin ML

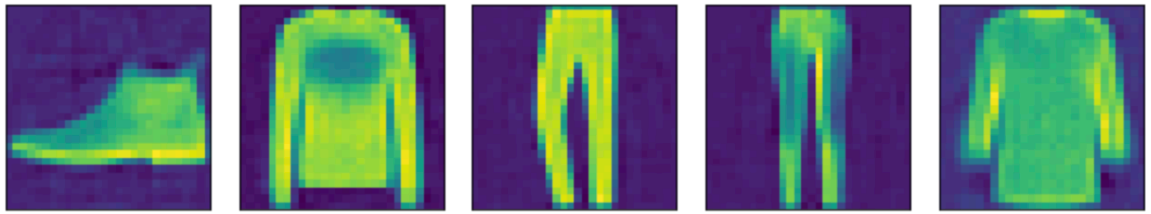
Autoencoders

Application: Dimensionality reduction – Feature extraction

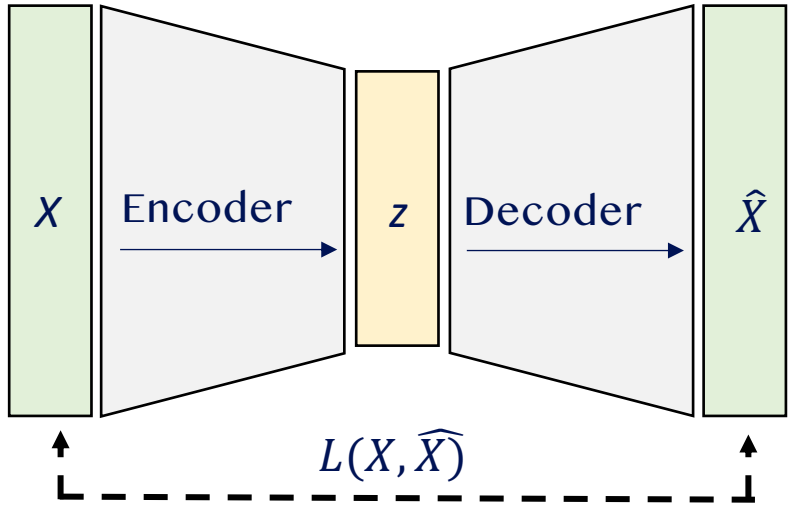
Original Images



Images reconstructed from Autoencoder



Autoencoder



Once we are done training, we can discard the decoder

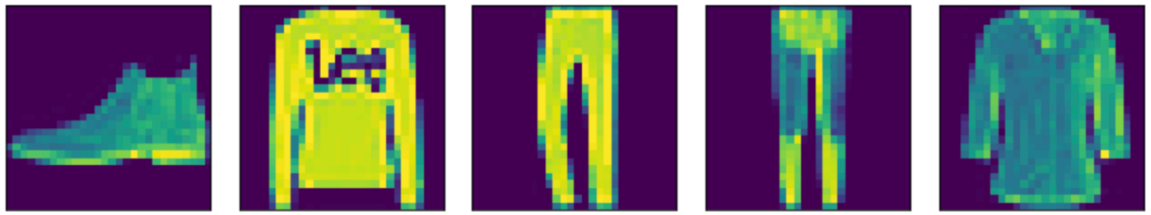
Fin ML

https://iq.opengenus.org/applications-of-autoencoders/

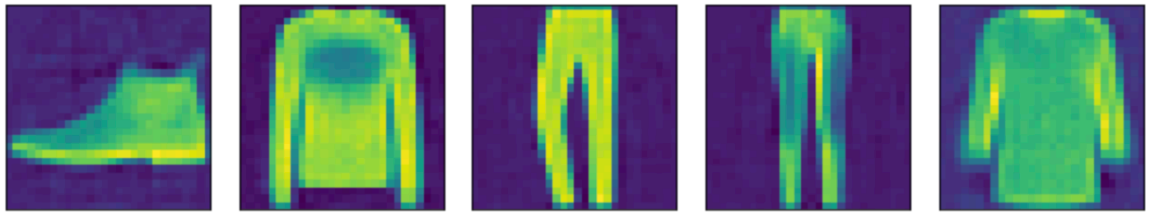
Autoencoders

Application: Dimensionality reduction – Feature extraction

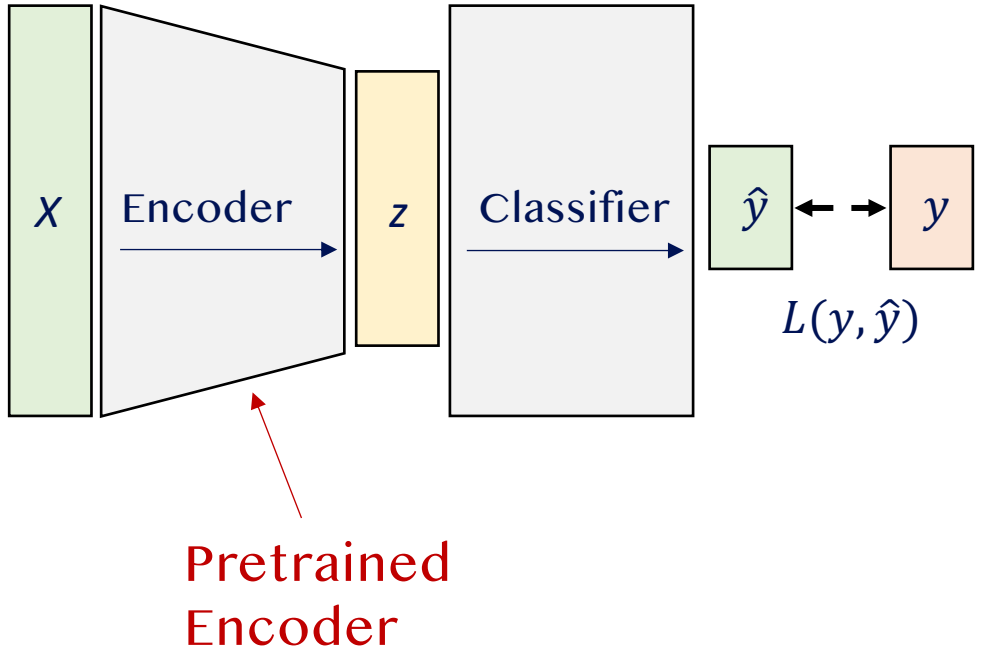
Original Images



Images reconstructed from Autoencoder



Autoencoder



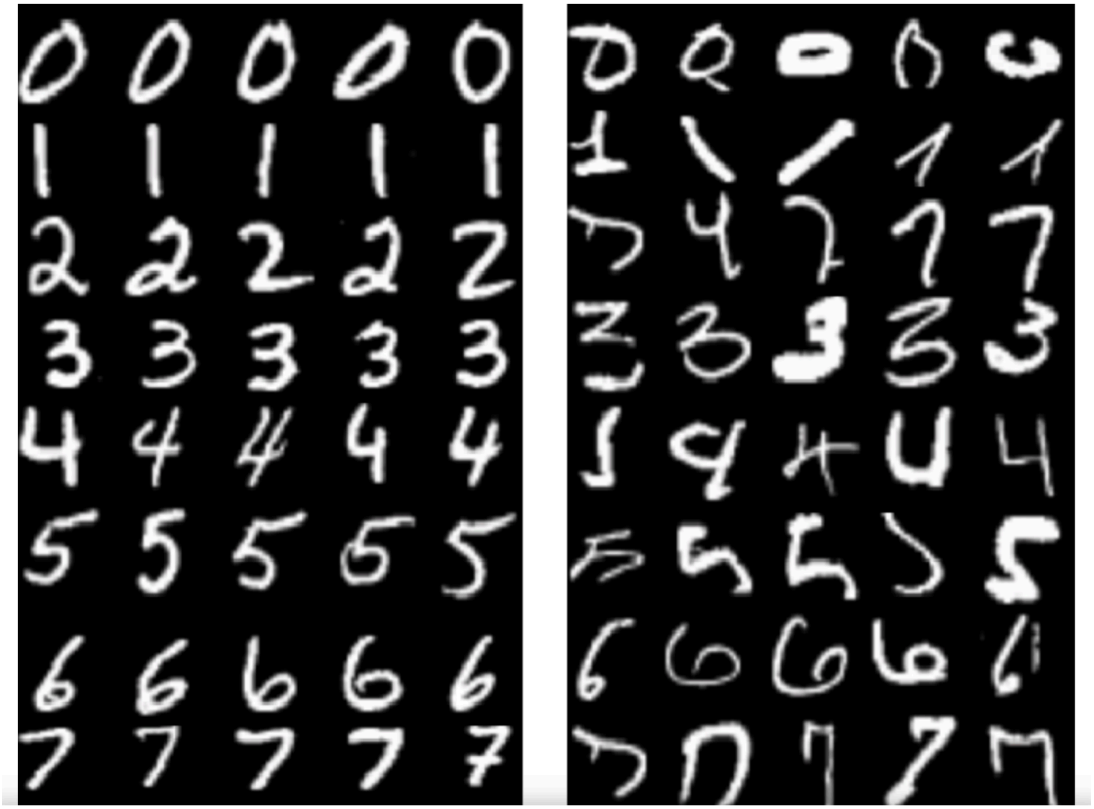
Fin ML

https://iq.opengenus.org/applications-of-autoencoders/

Autoencoders

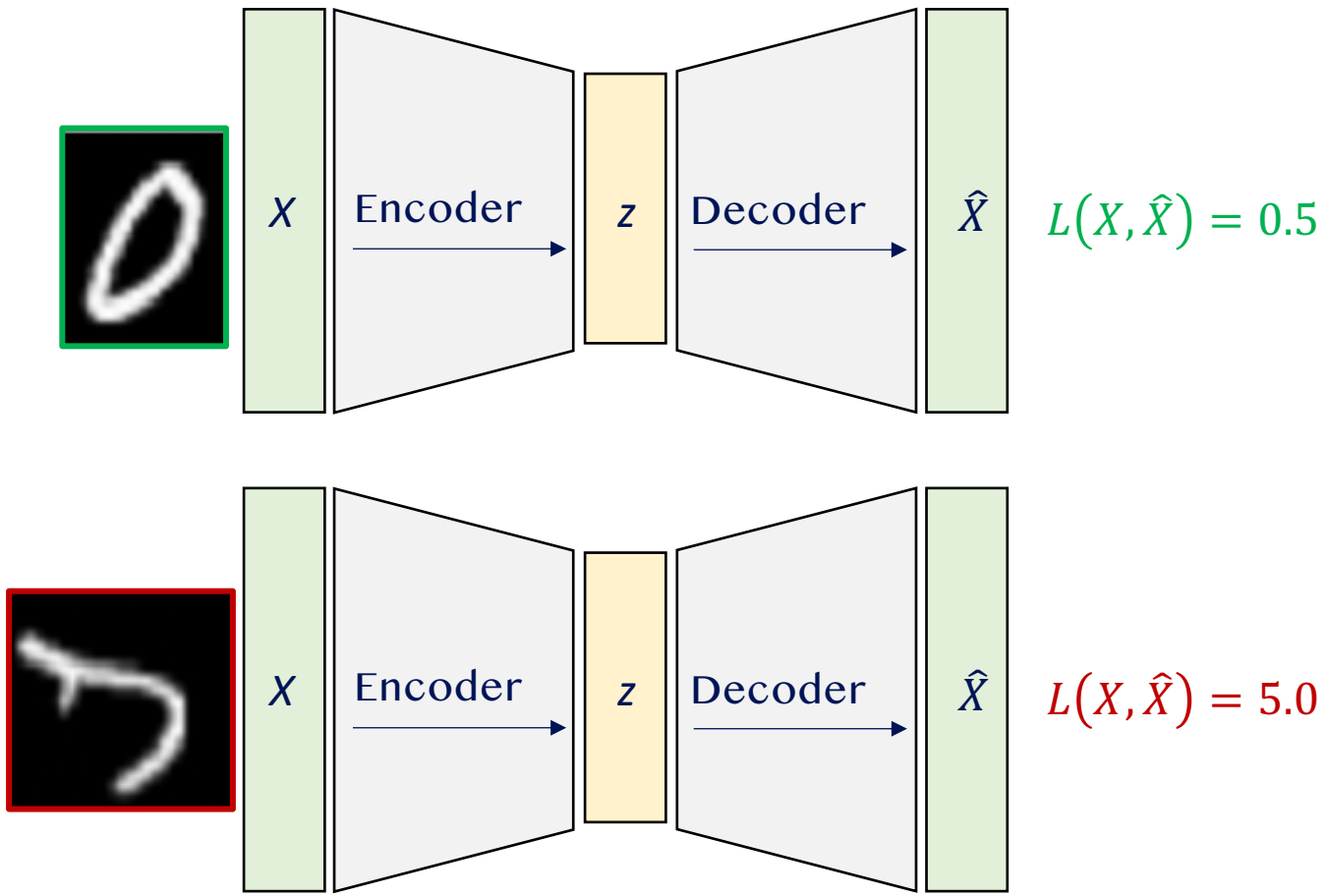
Application: Anomaly detection

Anomalies would be detected by choosing a threshold for the reconstruction loss



Normal inputs

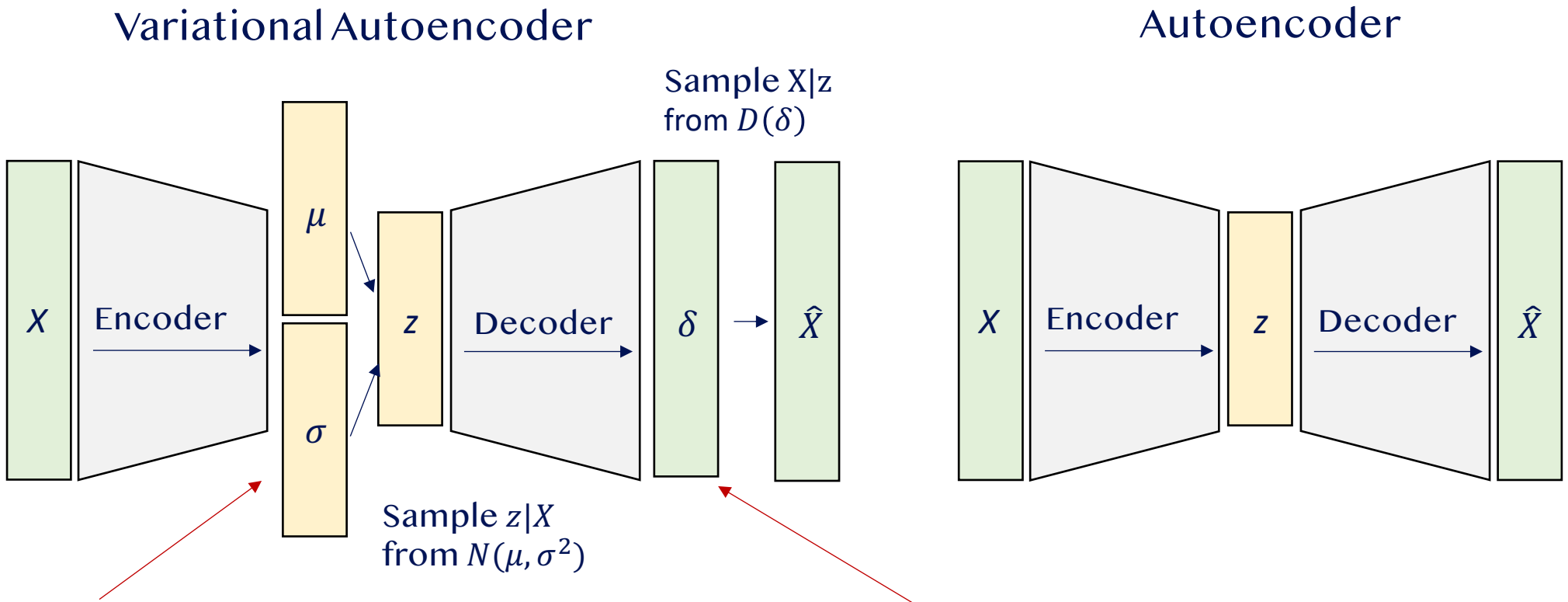
In-class anomalies



Fin ML

From Autoencoders to Variational Autoencoders

An Autoenconder with a probabilistic twist



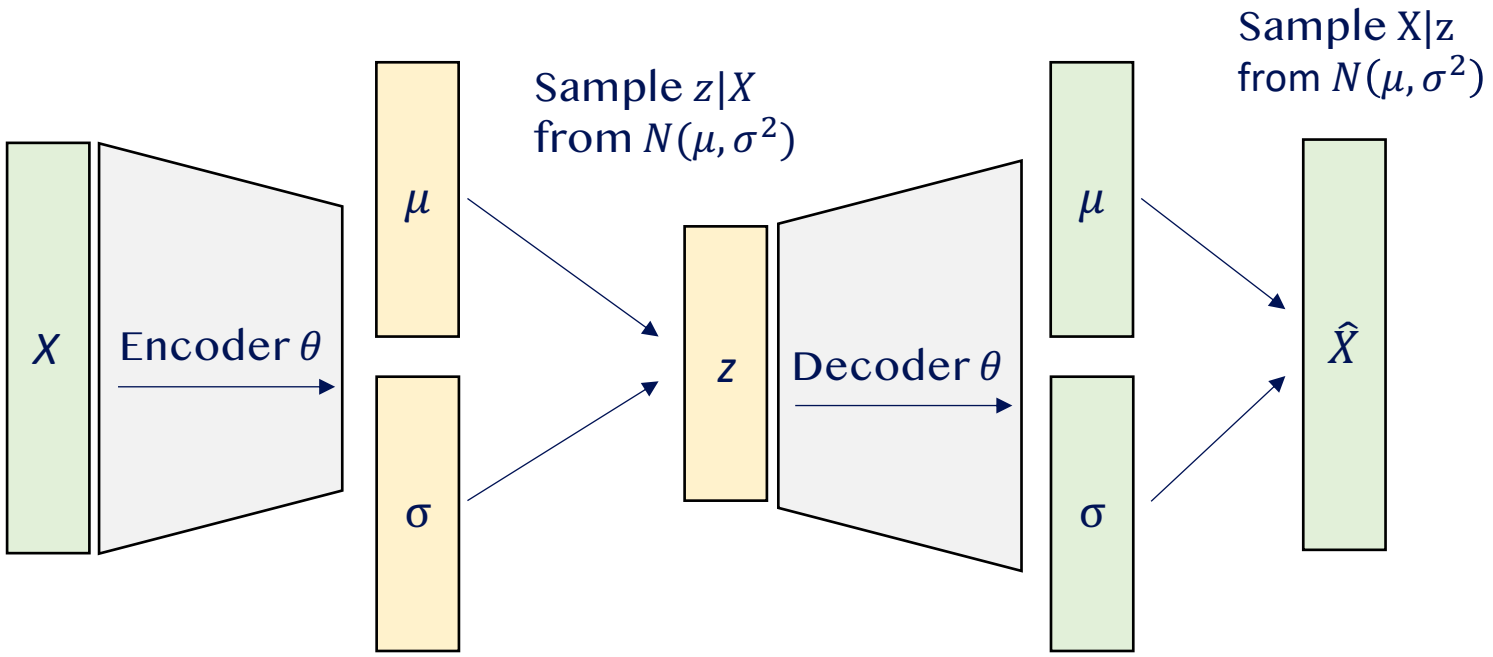
Fin ML

Variational Autoencoders

Implementation Choice - Gaussian

Gaussian example: $\boldsymbol{\delta = (\mu, \sigma^2)}$

$\boldsymbol{X \sim N(\mu, \sigma^2)}$



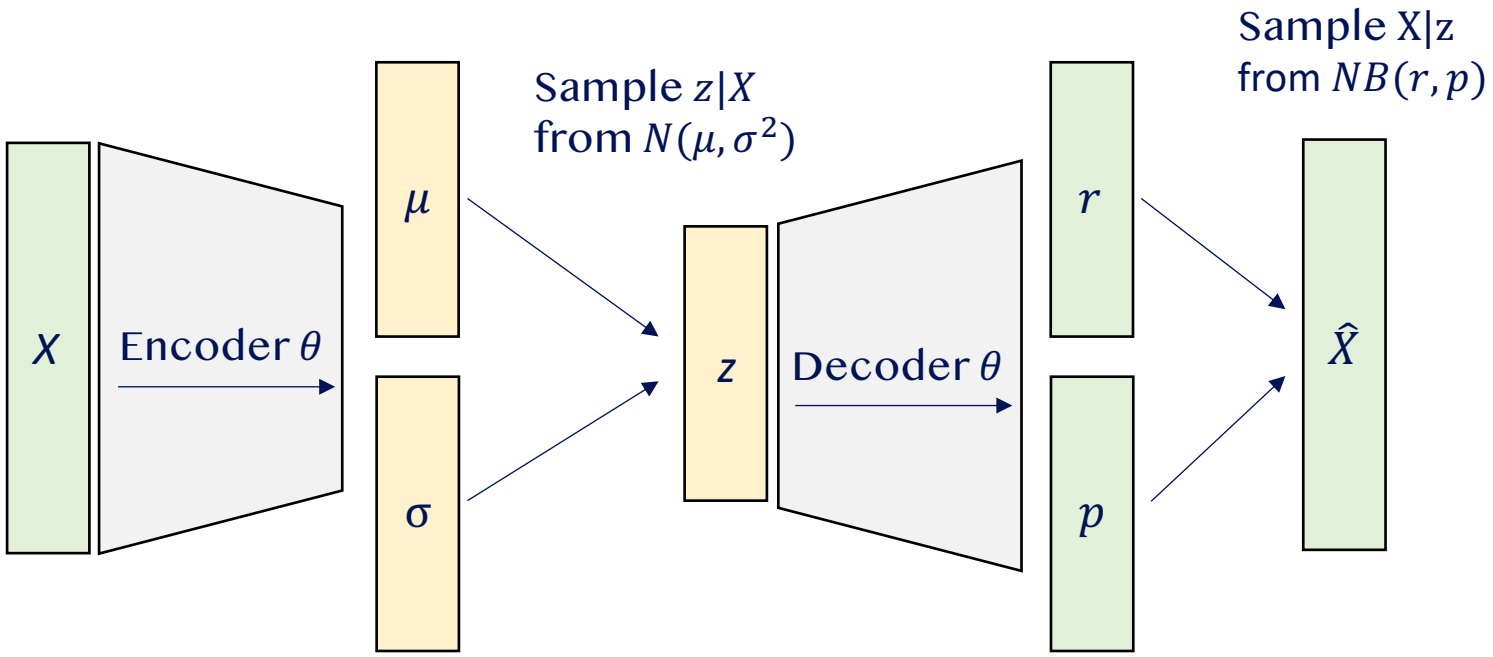
Fin ML

Variational Autoencoders

Implementation Choice – Negative Binomial

Negative binomial example: $\boldsymbol{\delta = (r, p)}$

$\boldsymbol{X \sim NB(r, p)}$

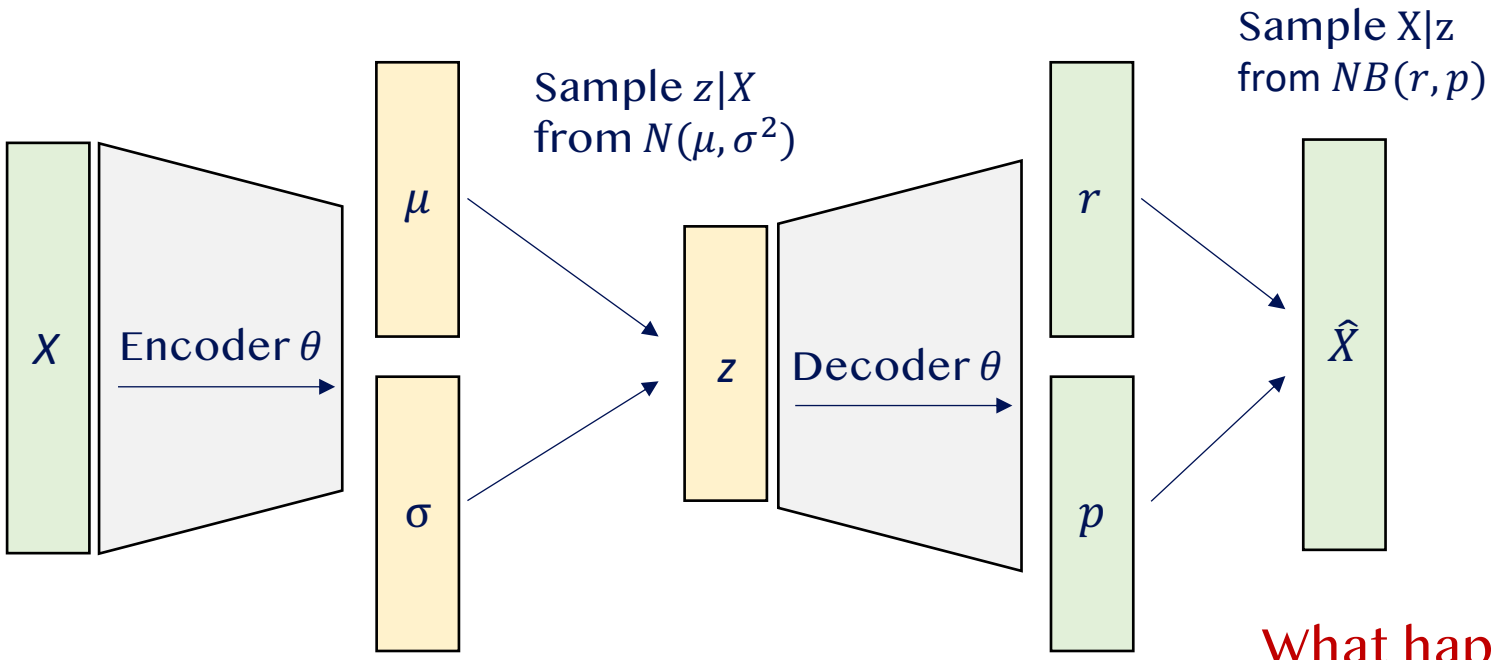


Variational Autoencoders

Implementation Choice – Negative Binomial

Negative binomial example: $\boldsymbol{\delta = (r, p)}$

$\boldsymbol{X \sim NB(r, p)}$



What happens if not all the variables have the same distribution?

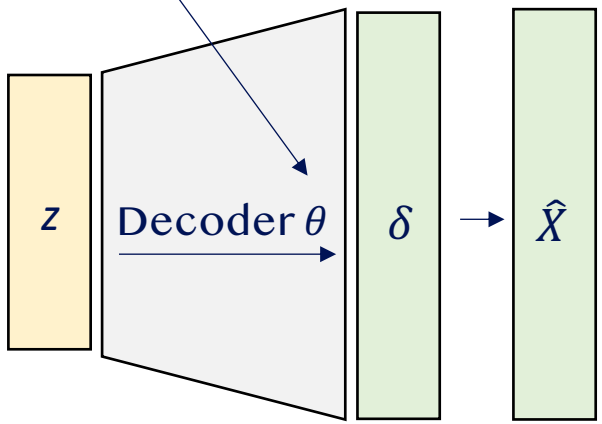
See HI-VAE: Nazabal et al. (2018)

Variational Autoencoders

The generative model mindset

We are given a dataset of N datapoints X

We want to estimate the parameters θ



Sample from the true pior $p_\theta(z)$

Sample from the true conditional $p_\theta(X|z)$

Fin·ML

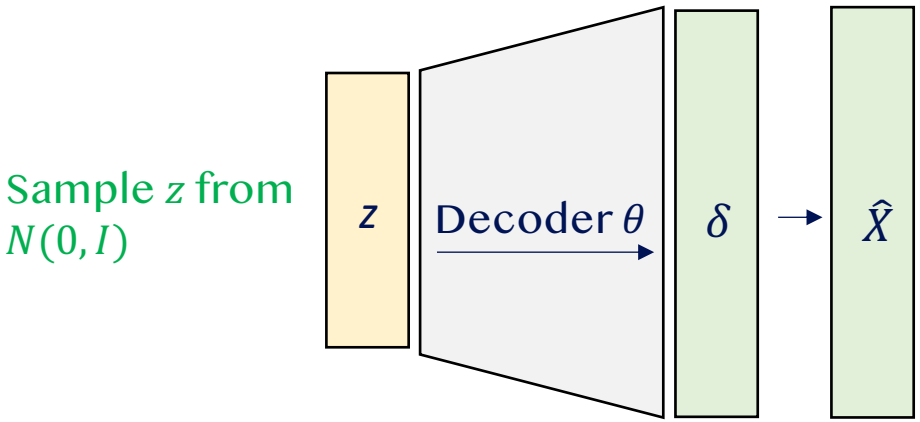
Some of the following slides were taken from Fei-Fei Li & Justin Johnson & Serena Yeung (2017)

Variational Autoencoders

The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ

In order to do this, we need to choose a distribution of our latent vector z. A gaussian distribution for example: $p_\theta(z) = N(0, I)$.



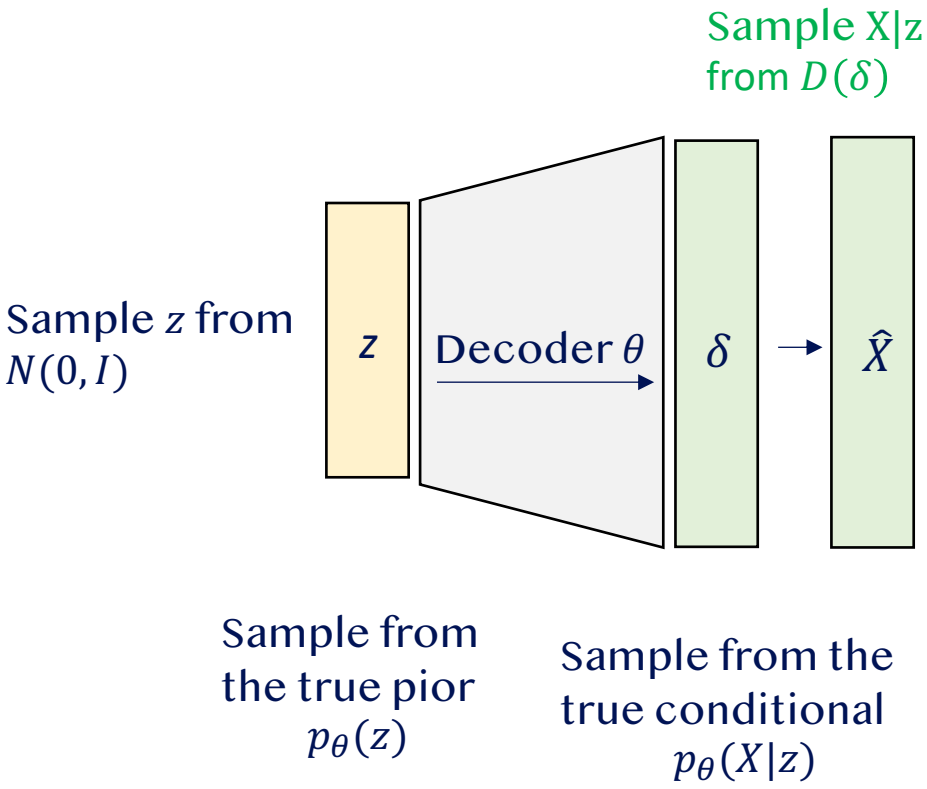
Sample from the true pior $p_\theta(z)$

Sample from the true conditional $p_\theta(X|z)$

Variational Autoencoders

The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ



In order to do this, we need to choose a distribution of our latent vector z. A gaussian distribution for example.

The conditional distribution $p_\theta(X|z)$ is modeled using the decoder network.

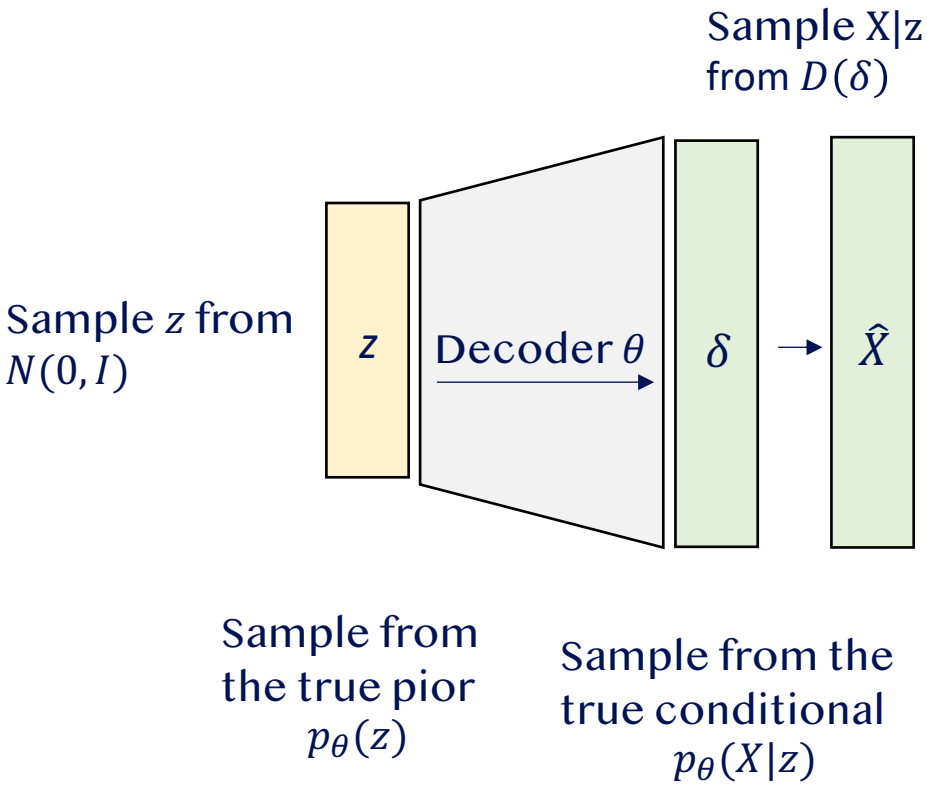
So we need to choose one or many distributions for this conditional distribution $p_\theta(X|z) = D(\delta)$.

Fin ML

Variational Autoencoders

The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ



In order to do this, we need to choose a distribution of our latent vector z. A gaussian distribution for example.

The conditional distribution $p_\theta(X|z)$ is modeled using the decoder network.

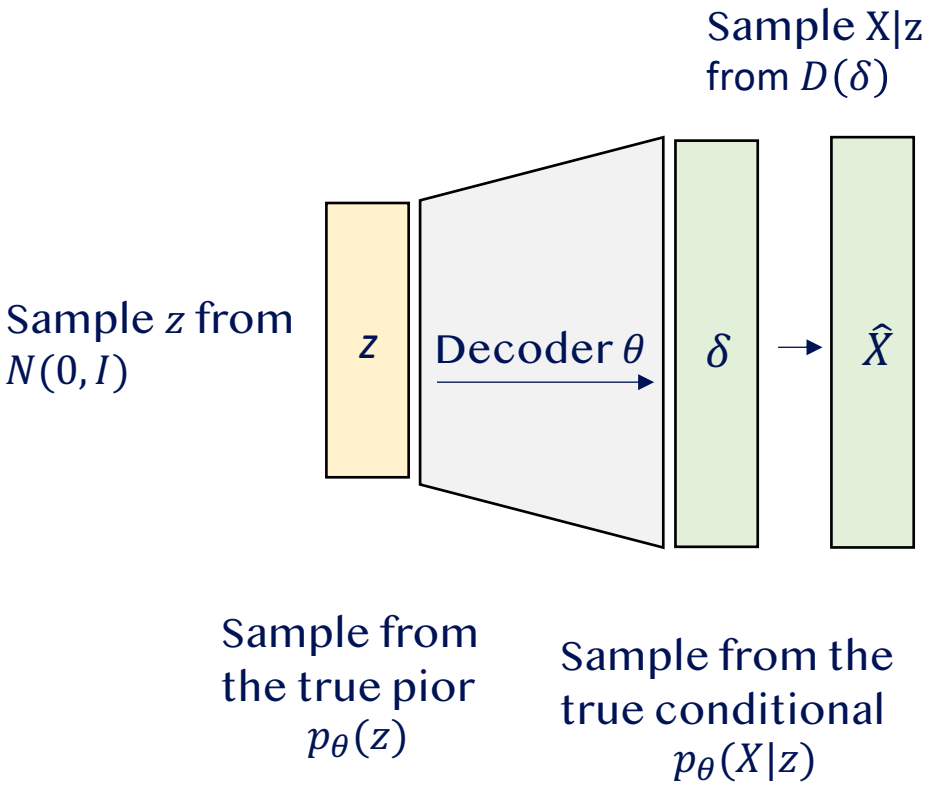
So we need to choose one or many distributions for this conditional distribution $p_\theta(X|z) = D(\delta)$.

How are we going to train this generative model?

Variational Autoencoders

The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ



In order to do this, we need to choose a distribution of our latent vector z. A gaussian distribution for example.

The conditional distribution is more complex, so using a neural network (encoder) is a great way to model complex functions. But we also need to choose a or many distributions for the conditional distribution.

How are we going to train this generative model?

We want to maximize the likelihood of our training data:

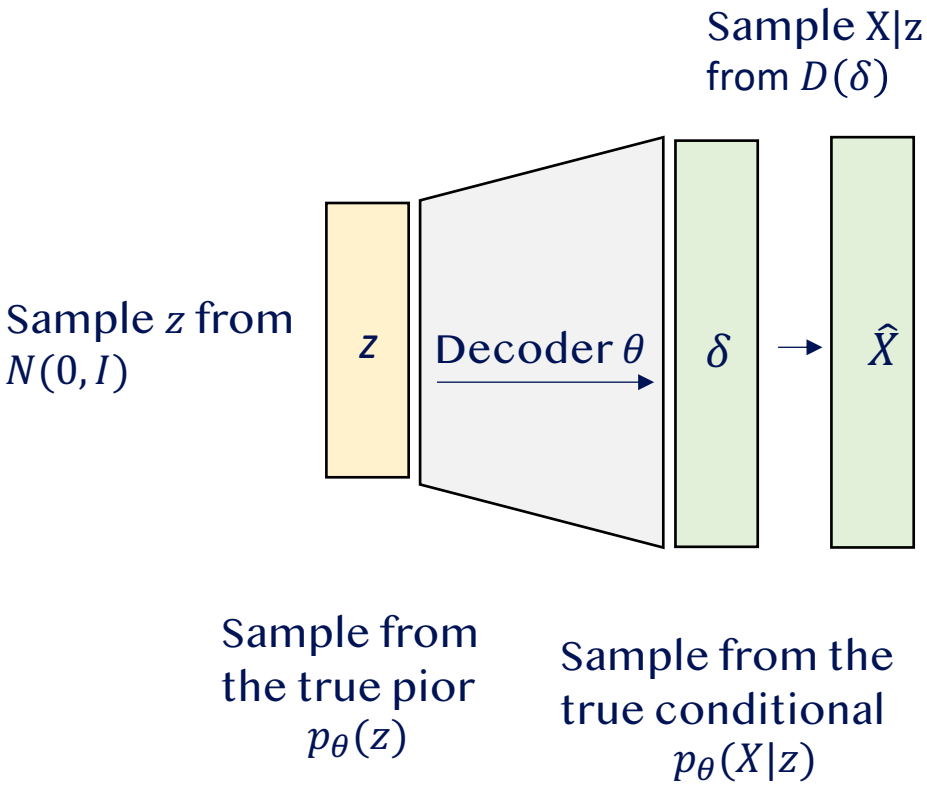
$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

Fin ML

Variational Autoencoders

The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ



In order to do this, we need to choose a distribution of our latent vector z. A gaussian distribution for example.

The conditional distribution is more complex, so using a neural network (encoder) is a great way to model complex functions. But we also need to choose a or many distributions for the conditional distribution.

How are we going to train this generative model?

We want to maximize the likelihood of our training data:

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

There is however a problem with this likelihood function

Fin ML

Variational Autoencoders

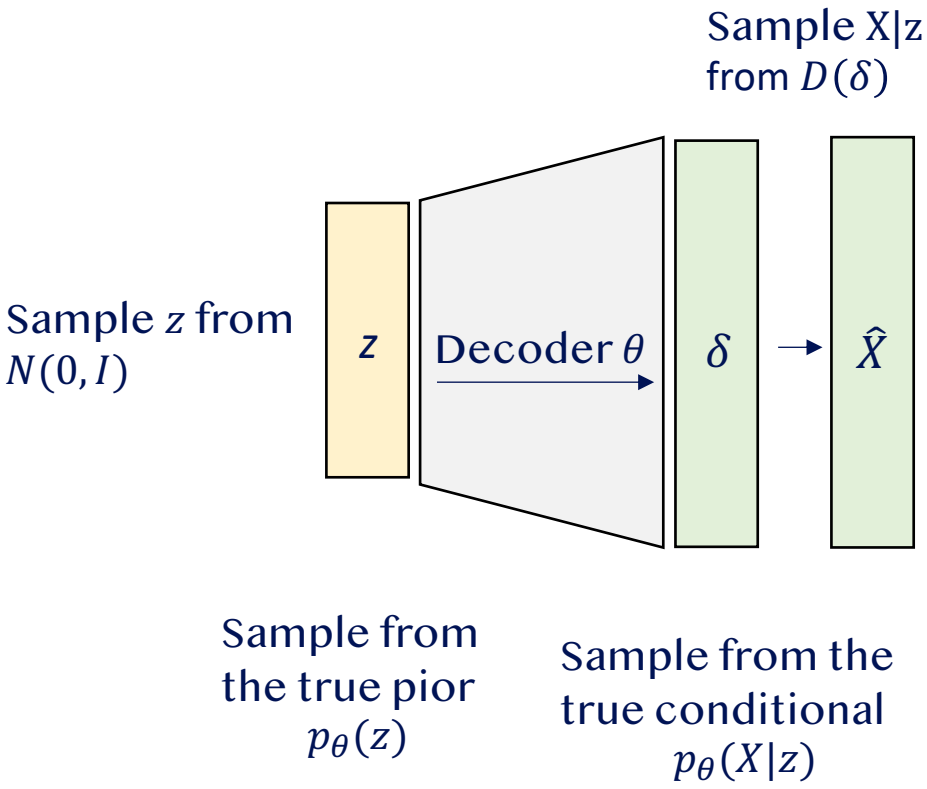
The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ

Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

Can we differentiate it?



Fin ML

Variational Autoencoders

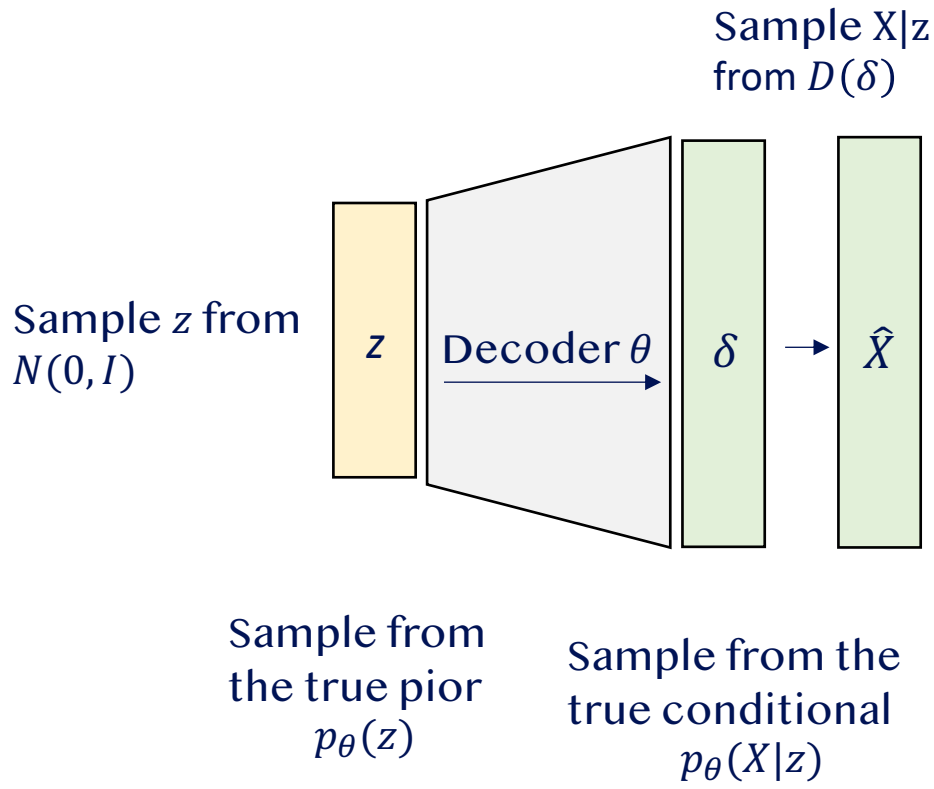
The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ

Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

Can we differentiate it?
Yes, we can. It's a decoder neural network



Fin ML

Variational Autoencoders

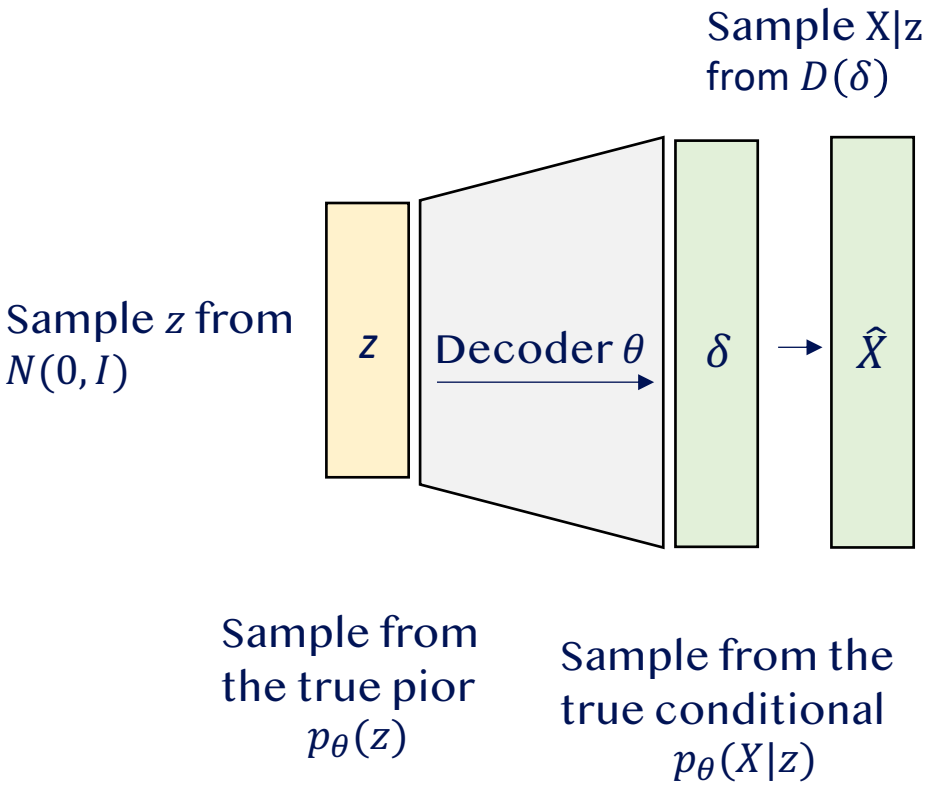
The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ

Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

Can we differentiate it?



Fin ML

Variational Autoencoders

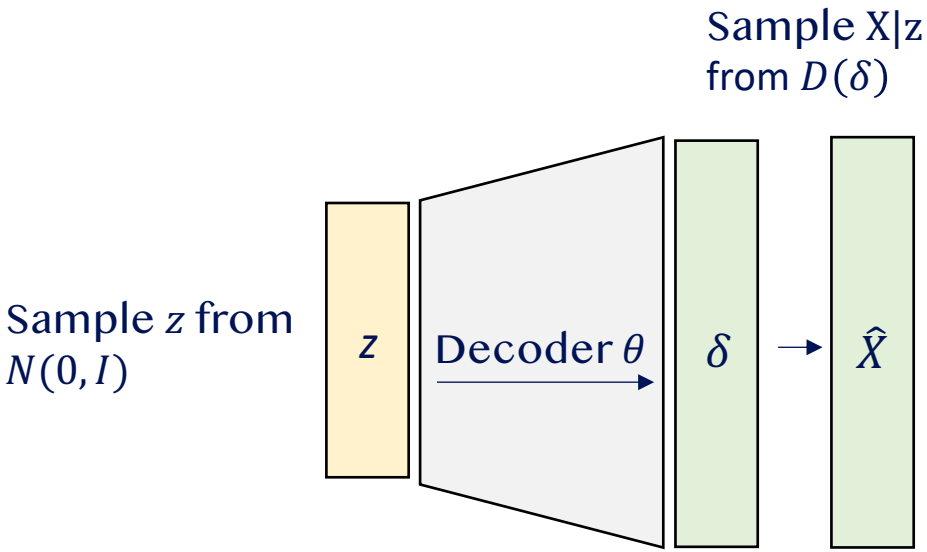
The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ

Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

Can we differentiate it?
Yes, we can. It's a gaussian distribution.



Fin ML

Variational Autoencoders

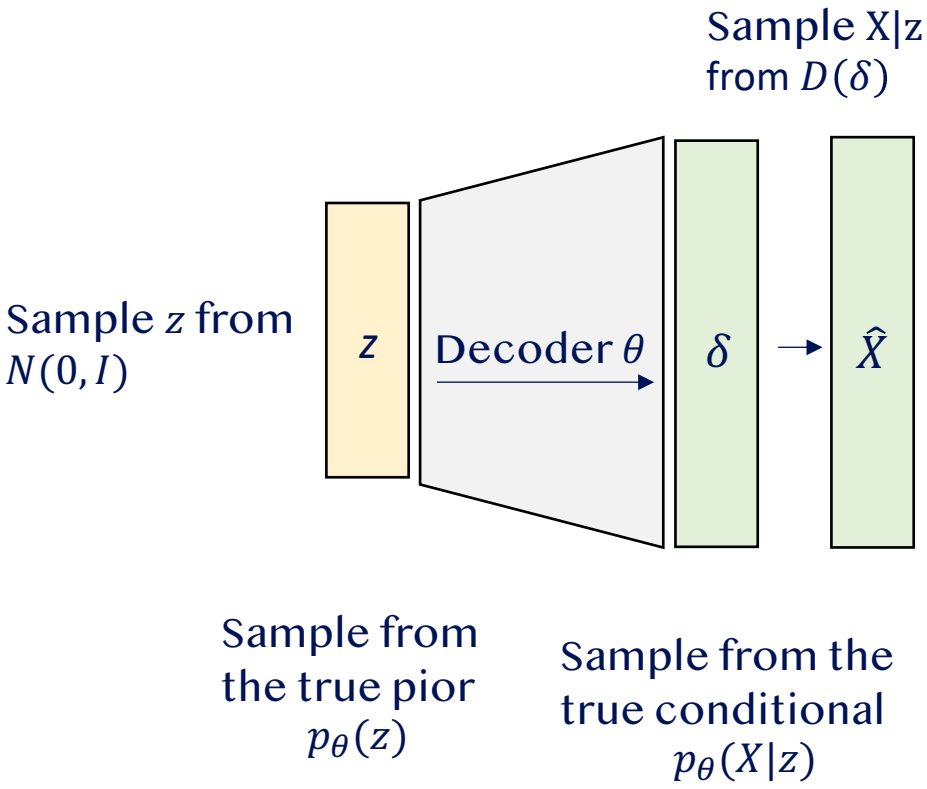
The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ

Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

Can we differentiate it?



Fin ML

Variational Autoencoders

The generative model mindset

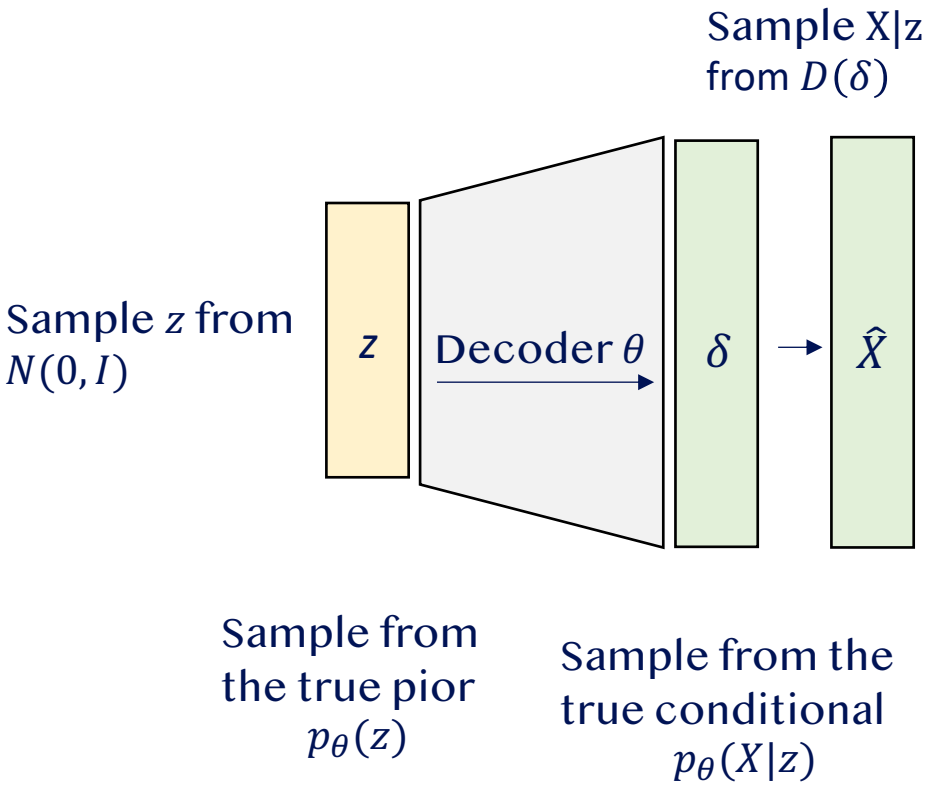
We are given a dataset of N datapoints X
We want to estimate the parameters θ

Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

Can we differentiate it?

No, the integrable makes the likelihood intractable.

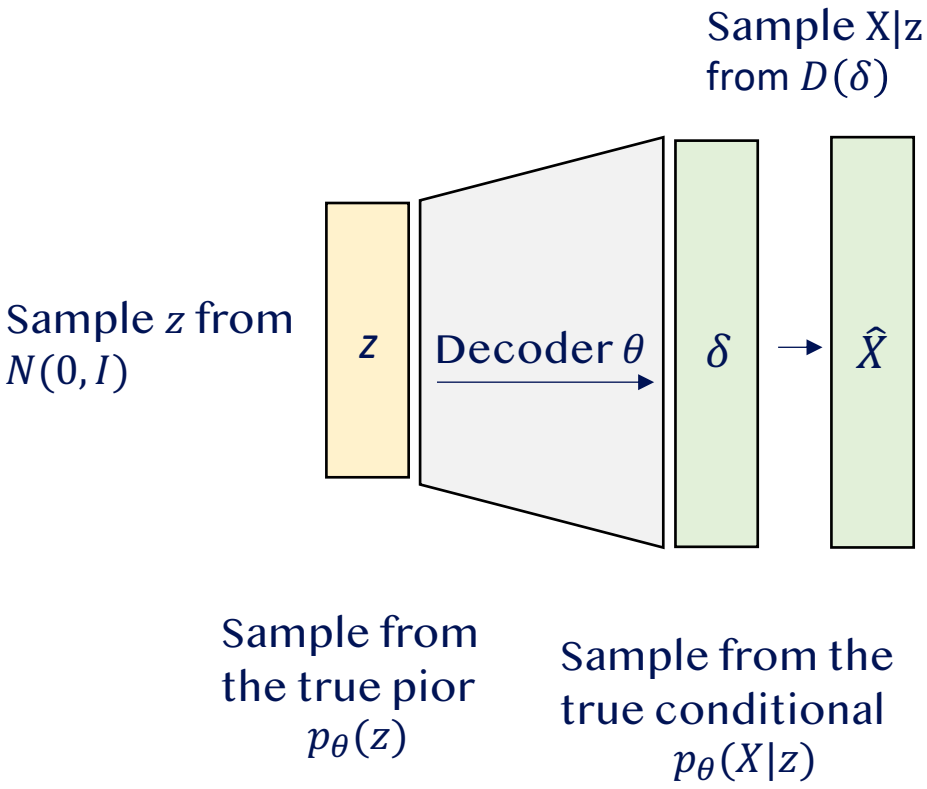


Fin ML

Variational Autoencoders

The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ



Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

The same issue arises with the posterior density

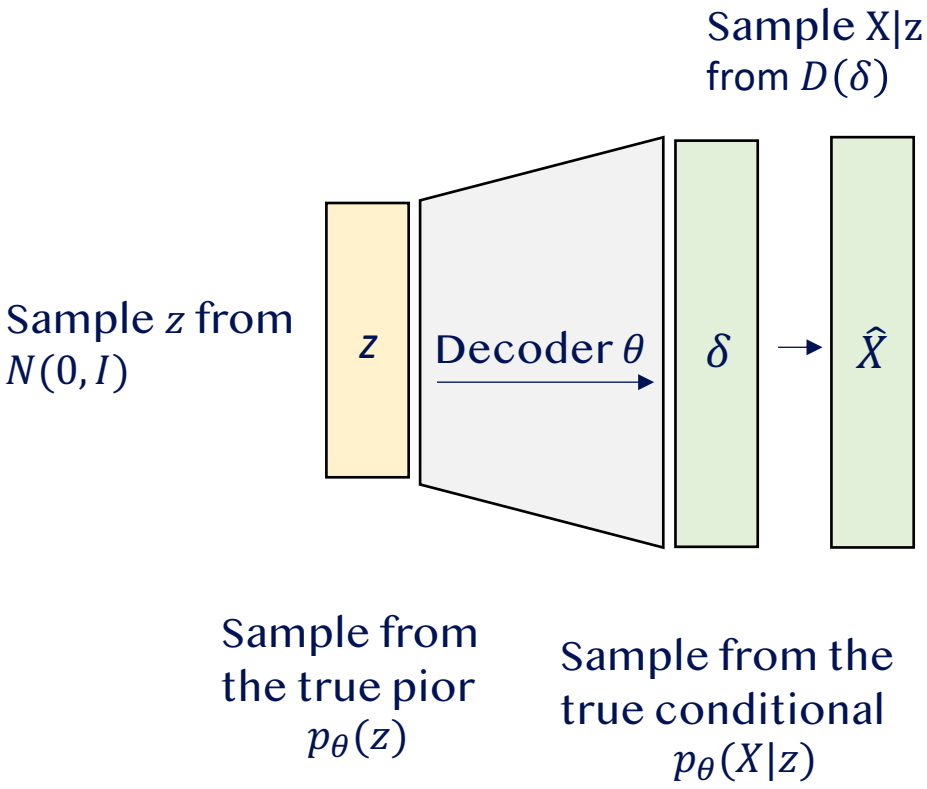
$$p_\theta(z|X) = \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(x)}$$

Again, the integrable from the likelihood makes the posterior density intractable

Variational Autoencoders

The generative model mindset

We are given a dataset of N datapoints X
We want to estimate the parameters θ



Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

The same issue arises with the posterior density

$$p_\theta(z|X) = \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(x)}$$

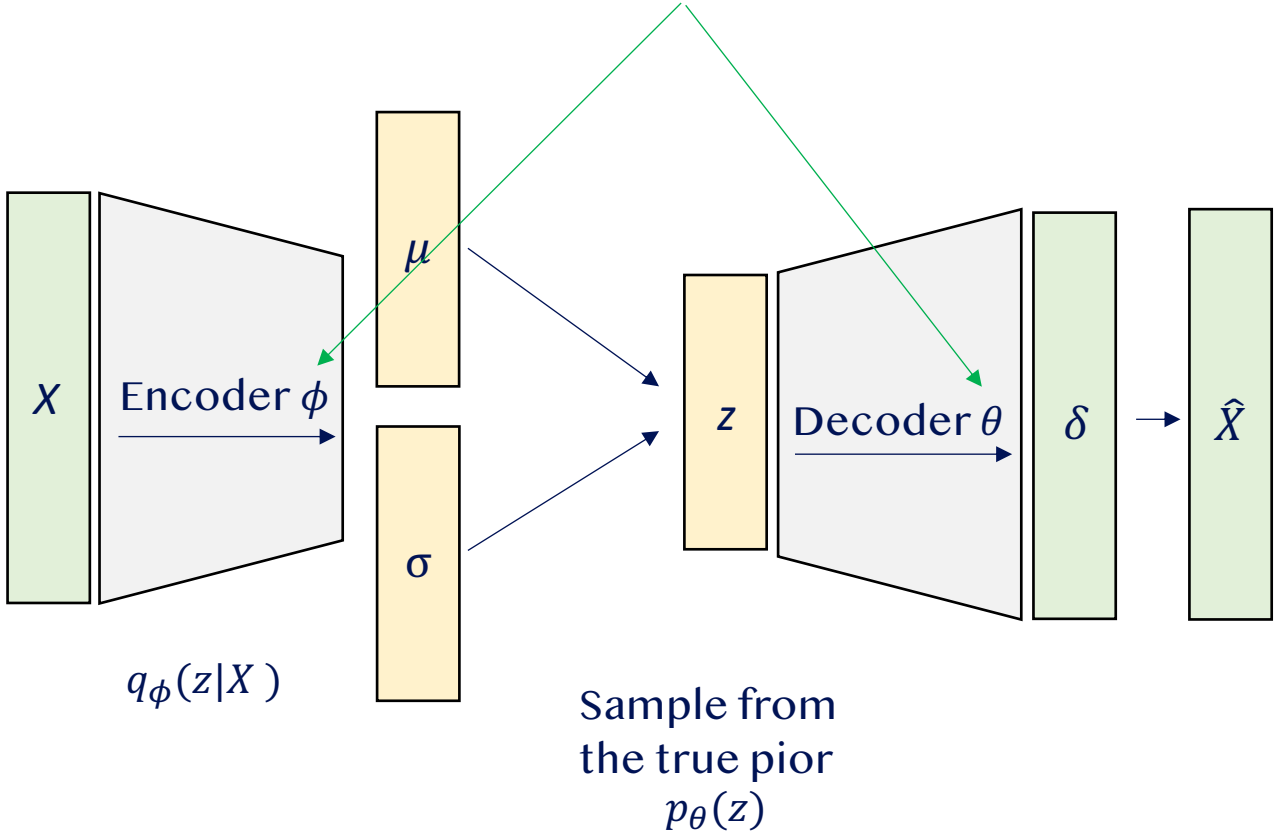
Again, the integrable from the likelihood makes the posterior density intractable

Variational Autoencoders

Decoder Network – The generative model

We are given a dataset of N datapoints X

We want to estimate the parameters θ, ϕ



Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

The same issue arises with the posterior density

$$p_\theta(z|X) = \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(x)}$$

Again, the integrable from the likelihood makes the posterior density intractable

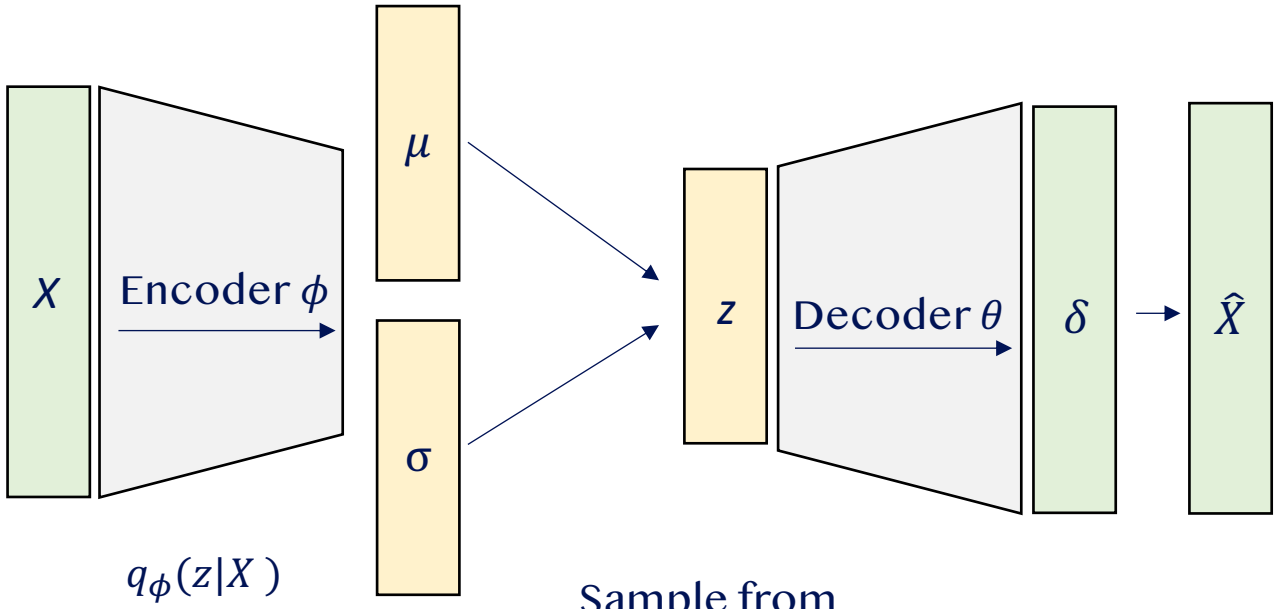
Solution is to approximate $p_\theta(z|X)$ with $q_\phi(z|X)$ using a encoder neural network

Fin ML

Variational Autoencoders

Decoder Network – The generative model

We are given a dataset of N datapoints X
We want to estimate the parameters θ, ϕ



Likelihhood

$$p_\theta(x) = \int p_\theta(X|z)\, p_\theta(z) dz$$

The same issue arises with the posterior density

$$p_\theta(z|X) = \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(x)}$$

Again, the integrable from the likelihood makes the posterior density intractable

Solution is to approximate $p_\theta(z|X)$ with $q_\phi(z|X)$ using a encoder neural network

It is not a perfect solution, but it will allow us to derive a lower bound for our likelihood

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$ Taking the expectation wrt z, does not depend on z

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)} \frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z\sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)} \frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

$$= E_z\left[\log p_\theta(X|z)\right] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

KL divergence:

$$\boldsymbol{E\left[\log \frac{q}{p}\right] = KL[q||p]}$$

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)} \frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

$$= E_z\left[\log p_\theta(X|z)\right] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

$$= E_z\left[\log p_\theta(X|z)\right] - KL\left[q_\phi(z|X)||p_\theta(z)\right] + KL\left[q_\phi(z|X)||p_\theta(z|X)\right]$$

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z\sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)} \frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

$$= E_z\ [\log p_\theta(X|z)] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

$$= E_z\ [\log p_\theta(X|z)] - KL\left[q_\phi(z|X)||p_\theta(z)\right] + KL\left[q_\phi(z|X)||p_\theta(z|X)\right]$$

From our decoder network, we can estimate $p_\theta(X|z)$ by sampling z many times . (Reparametrization trick)

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)} \frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

$$= E_z\left[\log p_\theta(X|z)\right] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

$$= \underbrace{E_z\left[\log p_\theta(X|z)\right]}_{\text{From our decoder network, we can estimate } p_\theta(X|z) \text{ by sampling z many times . (Reparametrization trick)}} - \underbrace{KL\left[q_\phi(z|X)||p_\theta(z)\right]}_{\text{KL divergence between 2 gaussians = differentiable closed-form}} + KL\left[q_\phi(z|X)||p_\theta(z|X)\right]$$

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)} \frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

$$= E_z\left[\log p_\theta(X|z)\right] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

$$= \underbrace{E_z\left[\log p_\theta(X|z)\right]}_{\text{From our decoder network, we can estimate } p_\theta(X|z) \text{ by sampling z many times . (Reparametrization trick)}} \underbrace{- KL\left[q_\phi(z|X)||p_\theta(z)\right]}_{\text{KL divergence between 2 gaussians = differentiable closed-form}} + \underbrace{KL\left[q_\phi(z|X)||p_\theta(z|X)\right]}_{\text{KL divergence is intractable of } p_\theta(z|X)}$$

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)} \frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

$$= E_z\left[\log p_\theta(X|z)\right] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

$$= \underbrace{E_z\left[\log p_\theta(X|z)\right]}_{\text{From our decoder network, we can estimate } p_\theta(X|z) \text{ by sampling z many times . (Reparametrization trick)}} \underbrace{- KL\left[q_\phi(z|X)||p_\theta(z)\right]}_{\text{KL divergence between 2 gaussians = differentiable closed-form}} \underbrace{+ KL\left[q_\phi(z|X)||p_\theta(z|X)\right]}_{\text{KL divergence is intractable of } p_\theta(z|X)}$$

But by definition a KL divergence is always bigger or equal to 0!

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z\sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\,p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\,p_\theta(z)}{p_\theta(z|X)}\,\frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

$$= E_z\,[\log p_\theta(X|z)] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

$$= \underbrace{E_z\,[\log p_\theta(X|z)] - KL\left[q_\phi(z|X)||p_\theta(z)\right]}_{L(X;\theta,\phi)} + \underbrace{KL\left[q_\phi(z|X)||p_\theta(z|X)\right]}_{\geq 0}$$

Instead of maximizing the likelihood, we maximize the lower bound

$\log p_\theta(x) \geq L(X;\theta,\phi)$ Lower bound $\longrightarrow$ $\theta^*,\phi^* = \text{argmax}\ L(X;\theta,\phi)$

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z\sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\,p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\,p_\theta(z)}{p_\theta(z|X)}\,\frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

$$= E_z\,[\log p_\theta(X|z)] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

$$= E_z\,[\log p_\theta(X|z)] - KL\left[q_\phi(z|X)||p_\theta(z)\right] + KL\left[q_\phi(z|X)||p_\theta(z|X)\right]$$

$L(X;\theta,\phi)$ ≥ 0

Trains the model to reconstruct the data as well as possible

Instead of maximizing the likelihood, we maximize the lower bound

$\log p_\theta(x) \geq L(X;\theta,\phi)$ Lower bound $\longrightarrow$ $\theta^*,\phi^* = \text{argmax}\ L(X;\theta,\phi)$

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X:

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(X|z)\, p_\theta(z)}{p_\theta(z|X)} \frac{q_\phi(z|X)}{q_\phi(z|X)}\right]$$ Multiplying by a constant

Acts as a regularizer: Forces the encoder posterior approximation to be close to prior

$$= E_z\left[\log p_\theta(X|z)\right] - E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]$$

$$= E_z\left[\log p_\theta(X|z)\right] - KL\left[q_\phi(z|X)||p_\theta(z)\right] + KL\left[q_\phi(z|X)||p_\theta(z|X)\right]$$

$L(X;\theta,\phi)$ ≥ 0

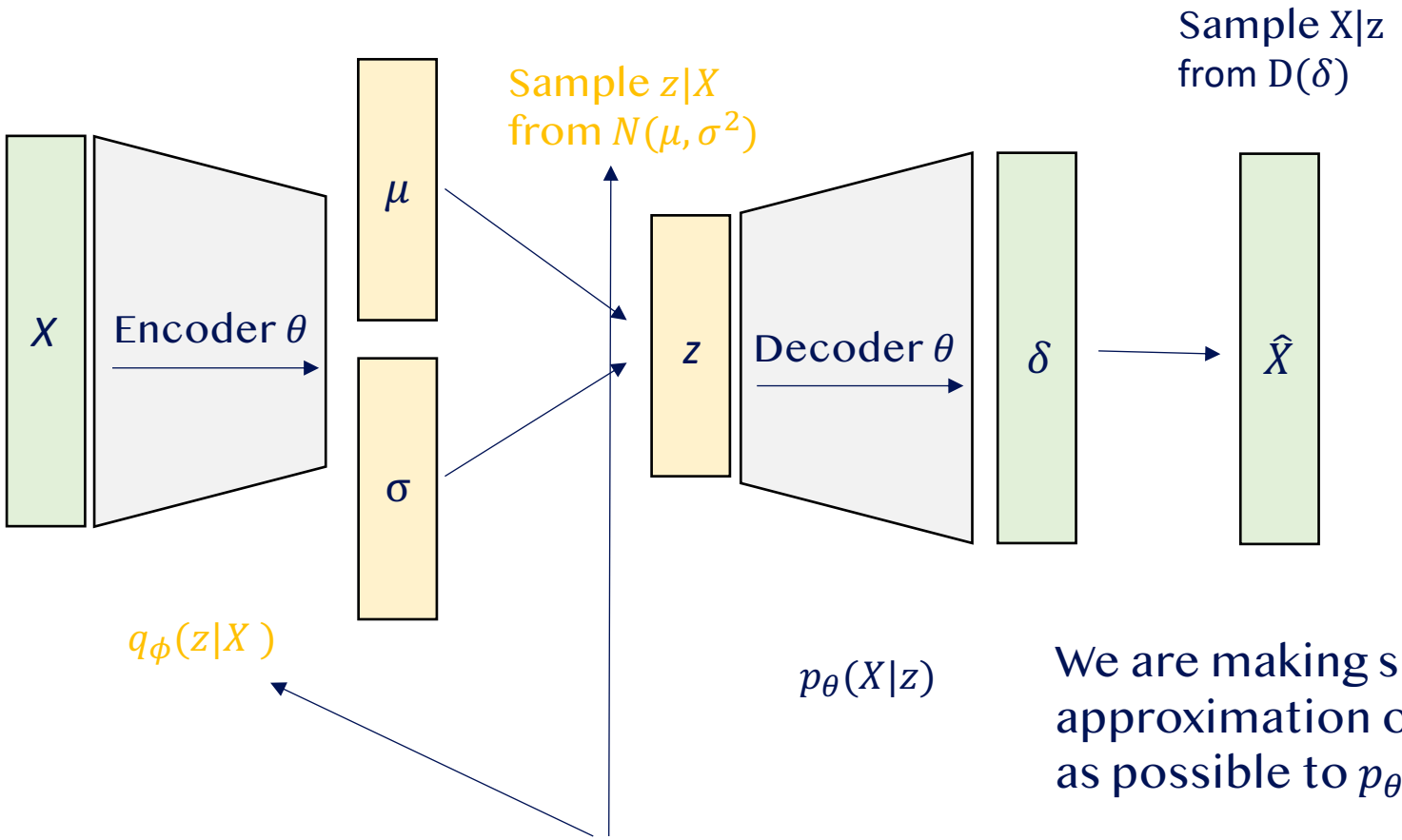
Instead of maximizing the likelihood, we maximize the lower bound

$\log p_\theta(x) \geq L(X;\theta,\phi)$ Lower bound $\longrightarrow$ $\theta^*, \phi^* = \text{argmax}\ L(X;\theta,\phi)$

Fin ML

Variational Autoencoders

Understanding the lower bound



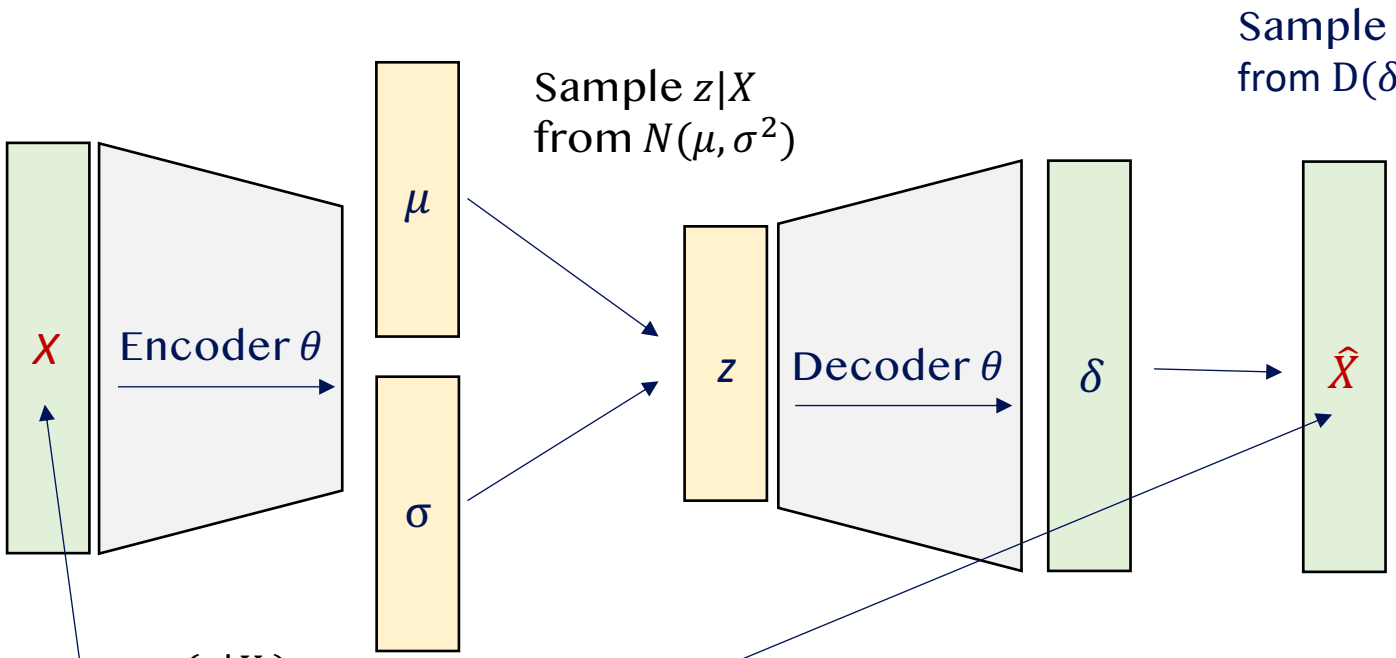
We are making sure the encoder approximation of $p_\theta(z|X)$ is as close as possible to $p_\theta(z) = \mathrm{N}(0, \mathrm{I})$.

$$L(X;\theta,\phi) = E_z\ [\log p_\theta(X|z)] - KL\left[q_\phi(z|X)||p_\theta(z)\right]$$

Fin ML

Variational Autoencoders

Understanding the lower bound



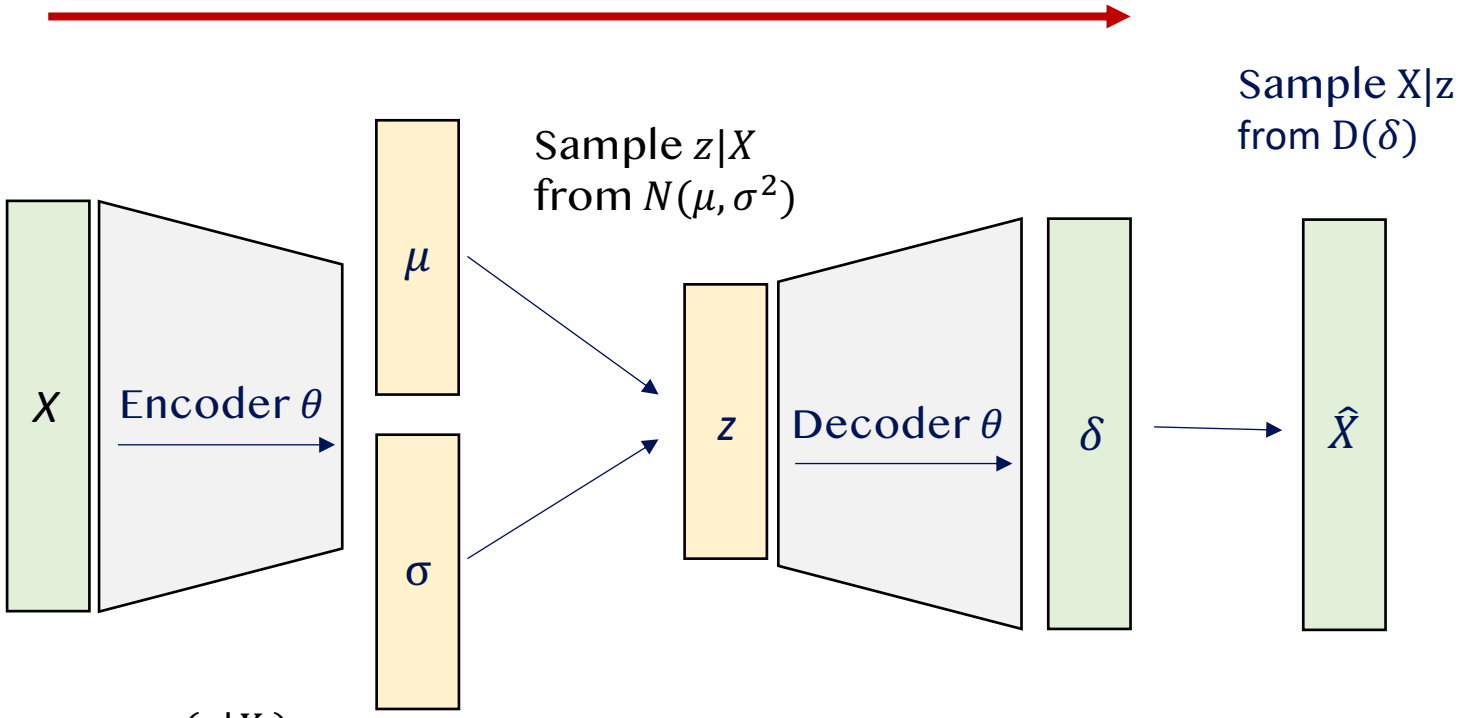
Maximize the likelihood of the training data

$$L(X;\theta,\phi) = E_z\,[\log p_\theta(X|z)] - KL\left[q_\phi(z|X)||p_\theta(z)\right]$$

Fin ML

Variational Autoencoders

Training

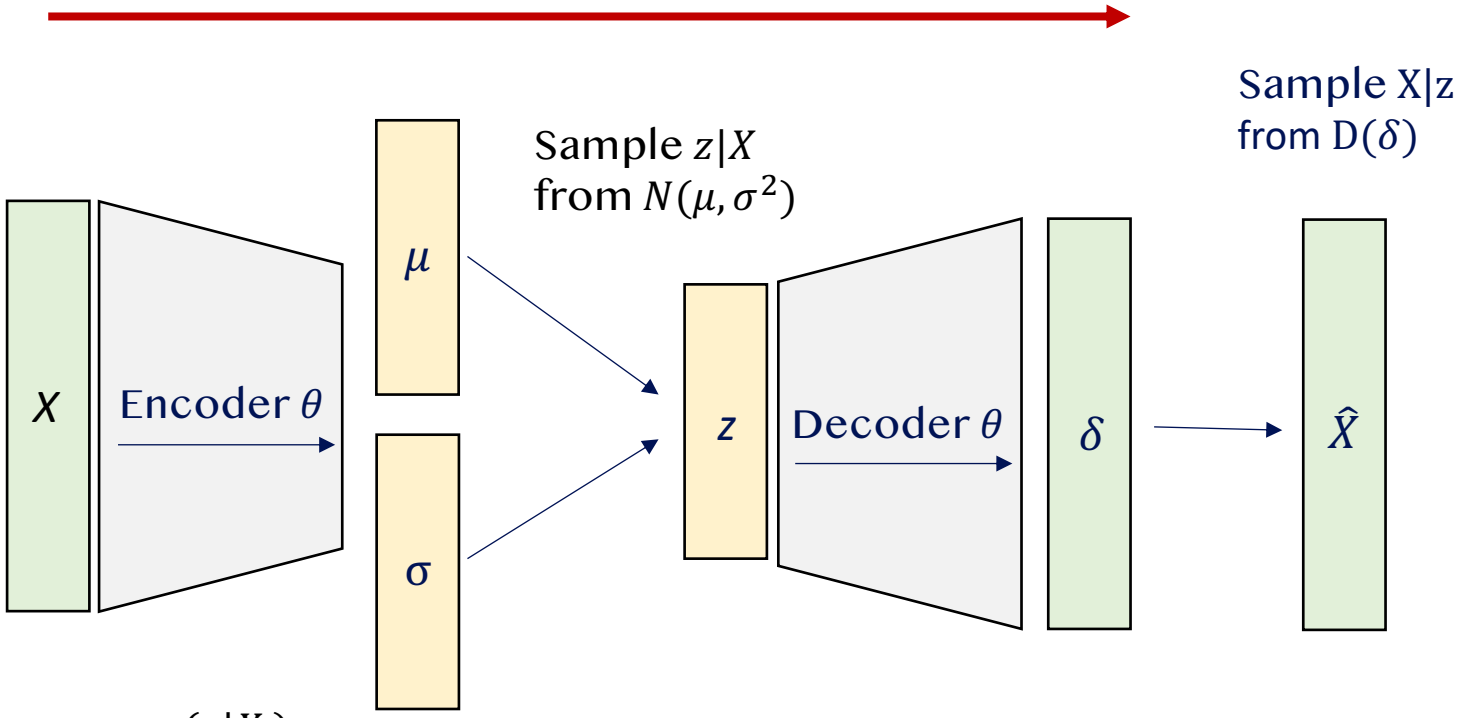


As we saw earlier, for a batch of training data, we can to a forward pass

$$L(X;\theta,\phi) = E_z\ [\log p_\theta(X|z)] \quad - KL\left[q_\phi(z|X)||p_\theta(z)\right]$$

Variational Autoencoders

Training



As we saw earlier, for a batch of training data, we can to a forward pass

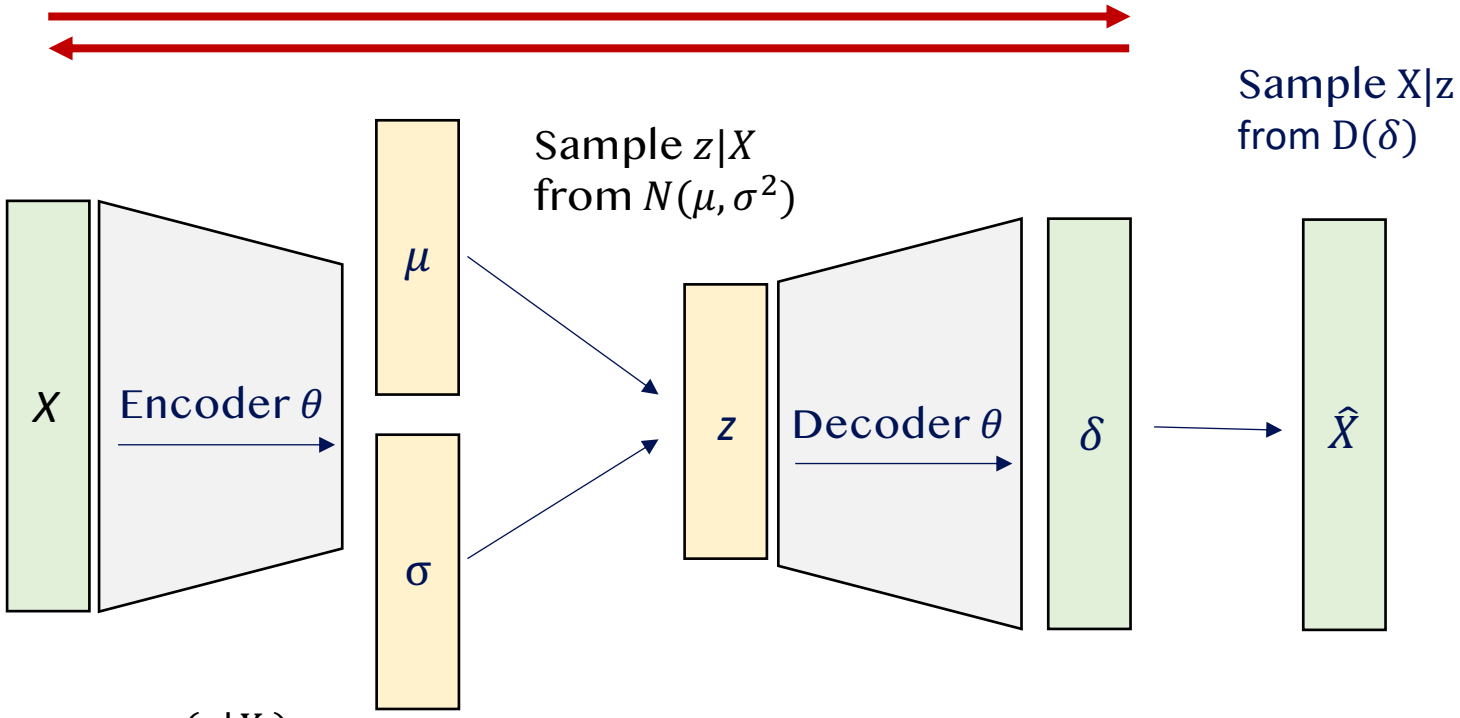
Compute our loss function

$$L(X;\theta,\phi) = E_z\ [\log p_\theta(X|z)] - KL\left[q_\phi(z|X)||p_\theta(z)\right]$$

Fin ML

Variational Autoencoders

Training



As we saw earlier, for a batch of training data, we can to a forward pass

Compute our loss function

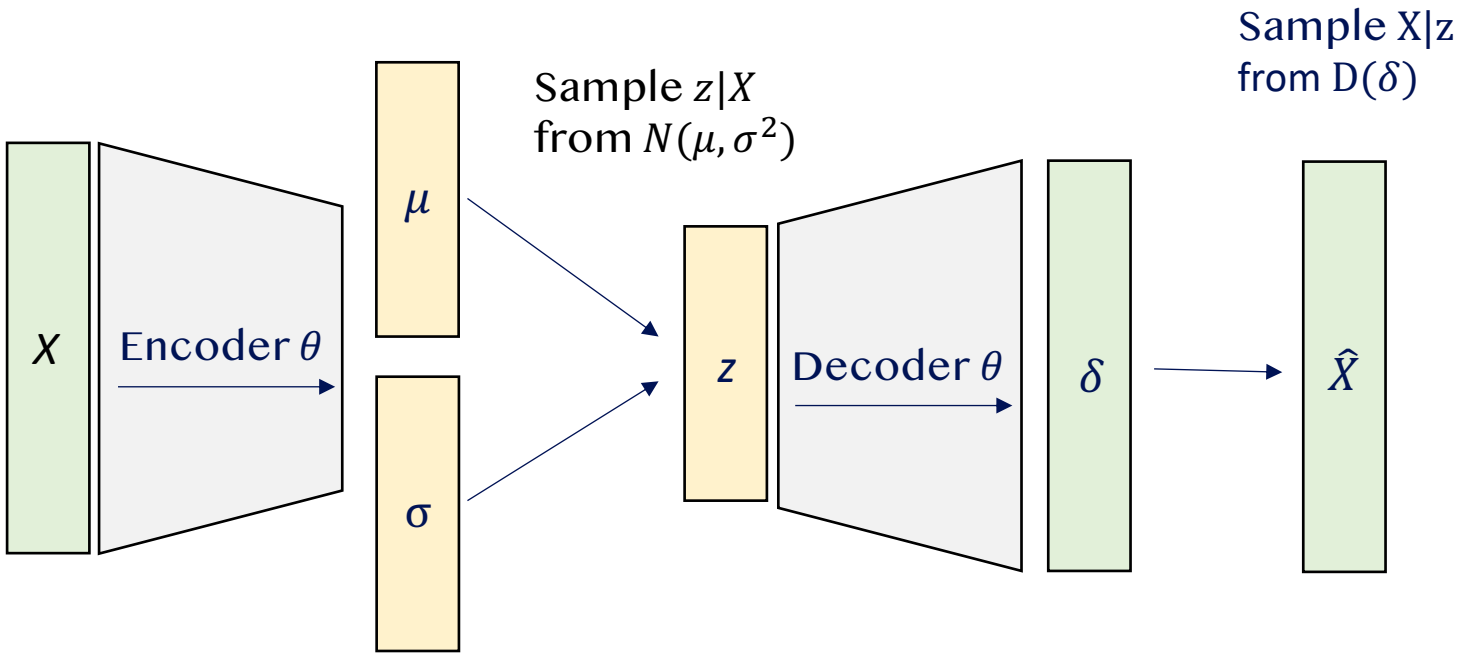
And backpropagate

$$L(X;\theta,\phi) = E_z\ [\log p_\theta(X|z)] - KL\left[q_\phi(z|X)||p_\theta(z)\right]$$

Fin ML

Variational Autoencoders

How do we use a VAE as a generator?



Conditional Variational Autoencoders

Generating by label

We just saw that it is possible to generate new data points that don't exist in the original dataset using a VAE.

Conditional Variational Autoencoders

Generating by label

We just saw that it is possible to generate new data points that don't exist in the original dataset using a VAE.

What if we want to generate new data points for a specific label?

Conditional Variational Autoencoders

Generating by label

We just saw that it is possible to generate new data points that don't exist in the original dataset using a VAE.

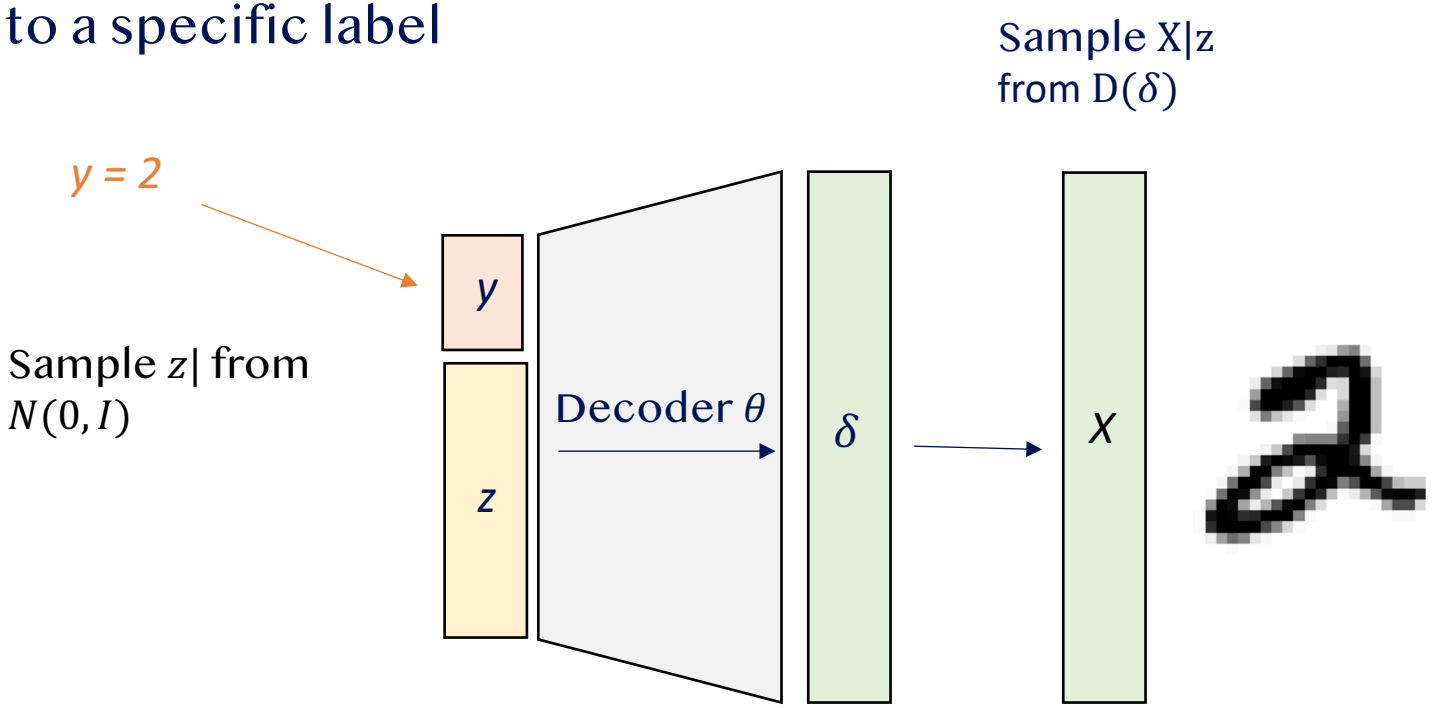
What if we want to generate new data points for a specific label?

This can be achieved using a Conditional VAE!

Conditional Variational Autoencoders

Generator with labels

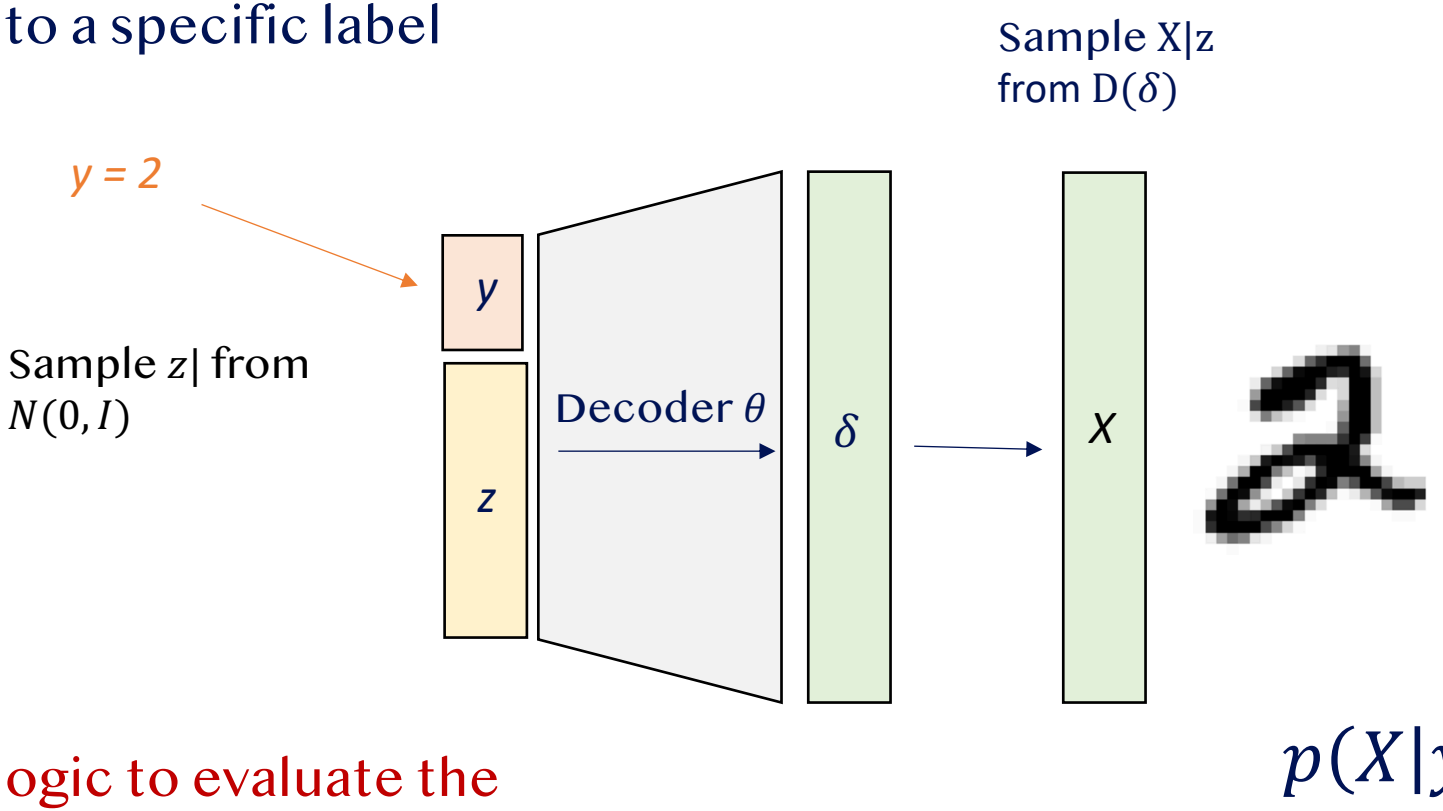
The idea is to train the CVAE so that we can generate data according to a specific label



Conditional Variational Autoencoders

Generator with labels

The idea is to train the CVAE so that we can generate data according to a specific label



$p(X|y)$

How can we use the same logic to evaluate the probability of an event?

Instead of modeling $p(X|y)$, we want to model $p(y|X)$

Conditional Variational Autoencoders

How can we use a CVAE to evaluate the probability of an event?

Insurance problem

Our goal is to evaluate the risk a carrier truck to have an accident so that we can accurately price the insurance premium.

Fin ML

Conditional Variational Autoencoders

How can we use a CVAE to evaluate the probability of an event?

Insurance problem

Our goal is to evaluate the risk a carrier truck to have an accident so that we can accurately price the insurance premium.

We have a database that contains ~50,000 datapoints (vehicle):

- We know how many accidents each vehicle had (y)
- We have a lot of information about this vehicle (X)
  - How many infractions it had
  - Features of the vehicle (axles, size, etc.)
  - How many days during a year it is allowed to circulate
  - …

Fin ML

Conditional Variational Autoencoders

How can we use a CVAE to evaluate the probability of an event?

Insurance problem

Our goal is to evaluate the risk a carrier truck to have an accident so that we can accurately price the insurance premium.

We have a database that contains ~50,000 datapoints (vehicle):

- We know how many accidents each vehicle had (y)
- We have a lot of information about this vehicle (X)
  - How many infractions it had
  - Features of the vehicle (axles, size, etc.)
  - How many days during a year it is allowed to circulate
  - …

What we want is to evaluate $P(y|X)$

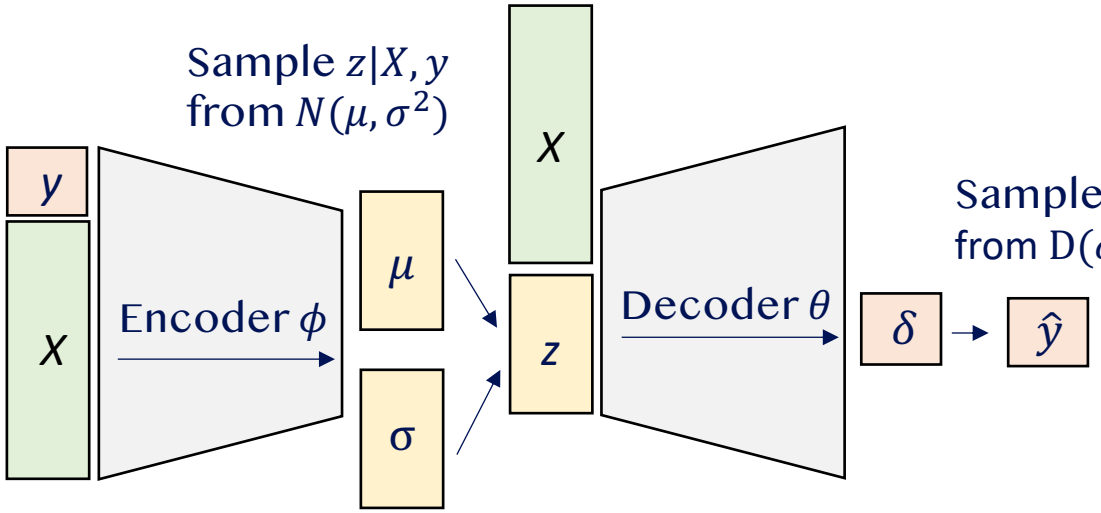
And that is a conditional probability!

Fin ML

Conditional Variational Autoencoders

Adapting the Network to our needs

We are given a dataset of N datapoints X, y
We want to estimate the parameters θ, ϕ



Intractable

Likelihood:

$$p_\theta(y|X) = \int p_\theta(y|X,z)\, p_\theta(z|X) dz$$

The same issue arises with the posterior density

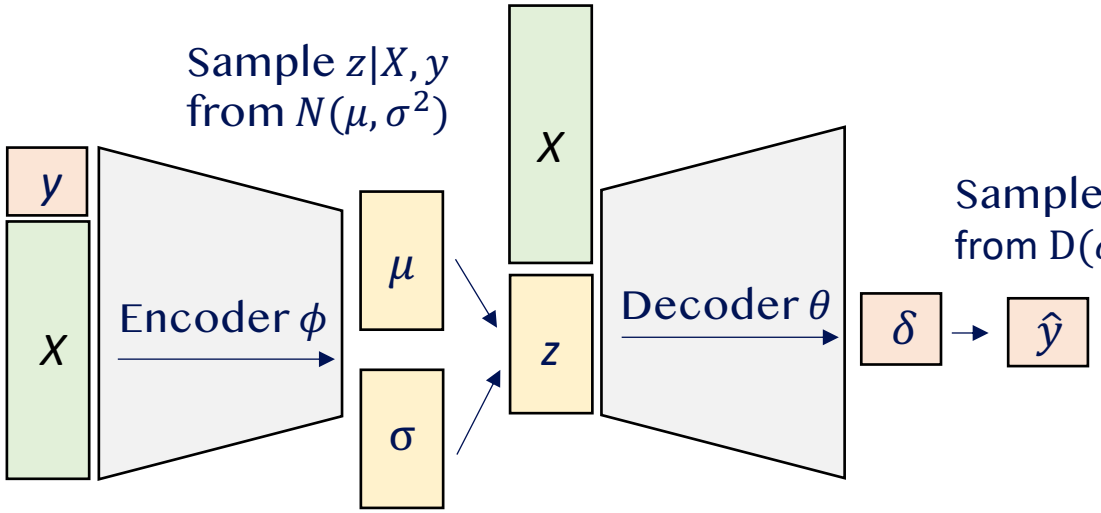
$$p_\theta(z|X,y) = \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(y|X)}$$

Fin ML

Conditional Variational Autoencoders

Adapting the Network to our needs

We are given a dataset of N datapoints X, y
We want to estimate the parameters θ, ϕ



Intractable

Likelihood:

$$p_\theta(y|X) = \int p_\theta(y|X,z)\, p_\theta(z|X) dz$$

The same issue arises with the posterior density

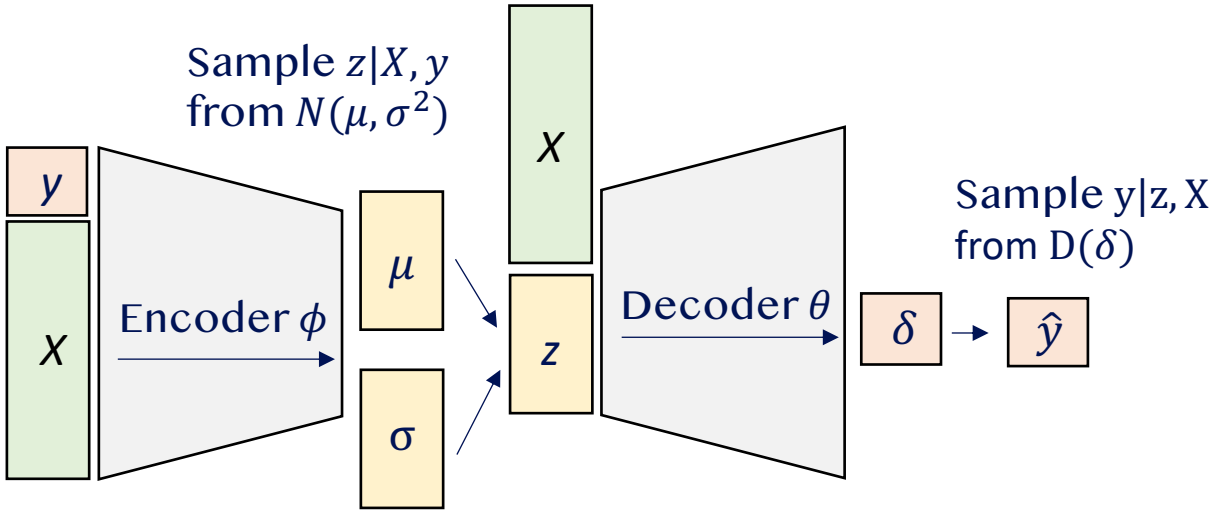
$$p_\theta(z|X,y) = \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(y|X)}$$

Fin ML

Conditional Variational Autoencoders

Adapting the Network to our needs

We are given a dataset of N datapoints X, y
We want to estimate the parameters θ, ϕ



Intractable

Likelihood:

$$p_\theta(y|X) = \int p_\theta(y|X,z)\, p_\theta(z|X) dz$$

The same issue arises with the posterior density

$$p_\theta(z|X,y) = \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(y|X)}$$

There are a few issues here:

1) Both the likelihood and posterior density are intractable

2) We need to feed y into the network

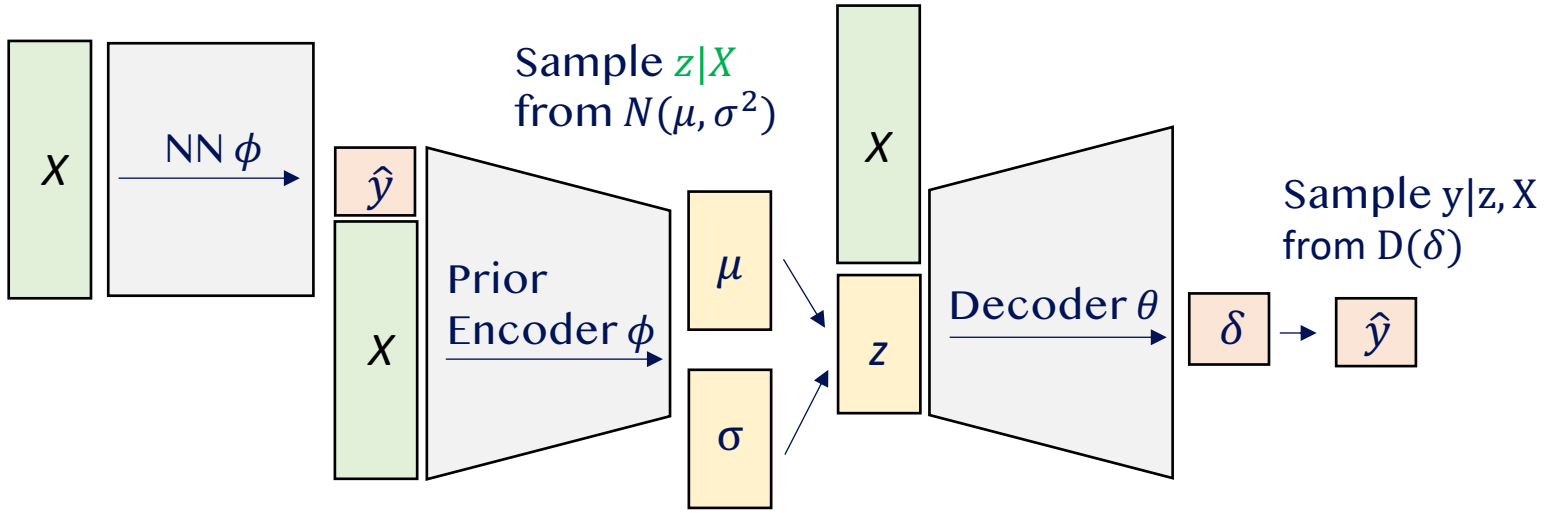
Fin ML

Sohn et al 2019

Conditional Variational Autoencoders

Adapting the Network to our needs

We are given a dataset of N datapoints X, y
We want to estimate the parameters θ, ϕ



Intractable

Likelihood:

$$p_\theta(y|X) = \int p_\theta(y|X,z)\, p_\theta(z|X) dz$$

The same issue arises with the posterior density

$$p_\theta(z|X,y) = \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(y|X)}$$

There are a few issues here:

1) Both the likelihood and posterior density are intractable

2) We need to feed y into the network

Solution:
1) Replace y by an approximation of y

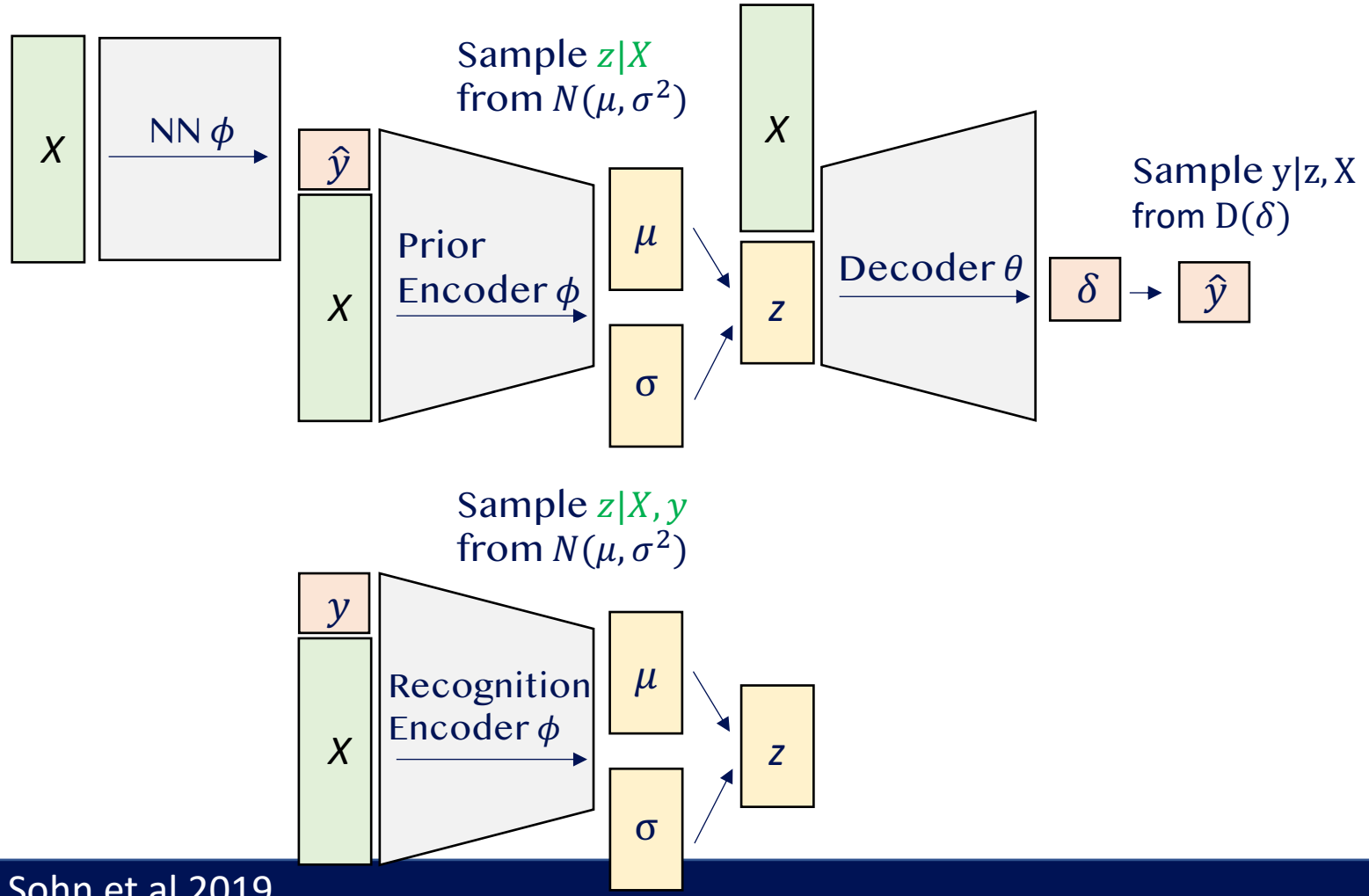
Fin ML

Sohn et al 2019

Conditional Variational Autoencoders

Adapting the Network to our needs

We are given a dataset of N datapoints X, y
We want to estimate the parameters θ, ϕ



Intractable

Likelihood:

$$p_\theta(y|X) = \int p_\theta(y|X,z)\, p_\theta(z|X) dz$$

The same issue arises with the posterior density

$$p_\theta(z|X,y) = \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(y|X)}$$

There are a few issues here:

1) Both the likelihood and posterior density are intractable
2) We need to feed y into the network

Solution:

1) Replace y by an approximation of y
2) Add a 'Support' Encoder Network to approximate $p_\theta(z|X,y)$ with $q_\phi(z|X,y)$

Fin ML

Sohn et al 2019

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$\log p_\theta(y|X) = E_{z \sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$ Taking the expectation wrt z, does not depend on z

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$\log p_\theta(y|X) = E_{z \sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$ Taking the expectation wrt z, does not depend on z

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X,y)}{p_\theta(z|X,y)}\right]$$ Baye's rule

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$$\log p_\theta(y|X) = E_{z \sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X,y)}{p_\theta(z|X,y)}\right]$$ Baye's rule

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X,y)}{p_\theta(z|X,y)} \frac{q_\phi(z|X,y)}{q_\phi(z|X,y)}\right]$$ Multiplying by a constant

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$$\log p_\theta(y|X) = E_{z\sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\,\frac{q_\phi(z|X,y)}{q_\phi(z|X,y)}\right]$$ Multiplying by a constant

$$= E_z\,[\log p_\theta(y|X,z)] - E_z\left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X)}\right] + E_z\left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X,y)}\right]$$

KL divergence:

$$\boldsymbol{E\left[\log \frac{q}{p}\right] = KL[q||p]}$$

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$\log p_\theta(y|X) = E_{z \sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$ Taking the expectation wrt z, does not depend on z

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\right]$$ Baye's rule

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)} \frac{q_\phi(z|X,y)}{q_\phi(z|X,y)}\right]$$ Multiplying by a constant

$$= E_z\ [\log p_\theta(y|X,z)] - E_z \left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X)}\right] + E_z \left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X,y)}\right]$$

KL divergence:

$$\boldsymbol{E\left[\log \frac{q}{p}\right] = KL[q||p]}$$

$$= E_z\ [\log p_\theta(y|X,z)] - KL\left[q_\phi(z|X,y)||p_\theta(z|X)\right] + KL\left[q_\phi(z|X,y)||p_\theta(z|X,y)\right]$$

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$\log p_\theta(y|X) = E_{z\sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\frac{q_\phi(z|X,y)}{q_\phi(z|X,y)}\right]$$ Multiplying by a constant

$$= E_z\left[\log p_\theta(y|X,z)\right] - E_z\left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X)}\right] + E_z\left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X,y)}\right]$$

KL divergence:

$$\boldsymbol{E\left[\log \frac{q}{p}\right] = KL[q||p]}$$

$$= \underbrace{E_z\left[\log p_\theta(y|X,z)\right]}_{} - KL\left[q_\phi(z|X,y)||p_\theta(z|X)\right] + KL\left[q_\phi(z|X,y)||p_\theta(z|X,y)\right]$$

From our decoder network, we can estimate $p_\theta(X|z)$ by sampling z many times . (Reparametrization trick)

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$\log p_\theta(y|X) = E_{z \sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$ Taking the expectation wrt z, does not depend on z

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\right]$$ Baye's rule

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)} \frac{q_\phi(z|X,y)}{q_\phi(z|X,y)}\right]$$ Multiplying by a constant

$$= E_z\ [\log p_\theta(y|X,z)] - E_z \left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X)}\right] + E_z \left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X,y)}\right]$$

$$= E_z\ [\log p_\theta(y|X,z)] - KL\left[q_\phi(z|X,y)||p_\theta(z|X)\right] + KL\left[q_\phi(z|X,y)||p_\theta(z|X,y)\right]$$

KL divergence:

$$\boldsymbol{E\left[\log \frac{q}{p}\right] = KL[q||p]}$$

From our decoder network, we can estimate $p_\theta(X|z)$ by sampling z many times . (Reparametrization trick)

KL divergence between 2 gaussians = differentiable closed-form

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$\log p_\theta(y|X) = E_{z \sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$ Taking the expectation wrt z, does not depend on z

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\right]$$ Baye's rule

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)} \frac{q_\phi(z|X,y)}{q_\phi(z|X,y)}\right]$$ Multiplying by a constant

$$= E_z\ [\log p_\theta(y|X,z)] - E_z \left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X)}\right] + E_z \left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X,y)}\right]$$

KL divergence:

$$\boldsymbol{E\left[\log \frac{q}{p}\right] = KL[q||p]}$$

$$= E_z\ [\log p_\theta(y|X,z)] - KL[q_\phi(z|X,y)||p_\theta(z|X)] + KL[q_\phi(z|X,y)||p_\theta(z|X,y)]$$

From our decoder network, we can estimate $p_\theta(X|z)$ by sampling z many times . (Reparametrization trick)

KL divergence between 2 gaussians = differentiable closed-form

KL divergence is intractable of $p_\theta(z|X,y)$

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$\log p_\theta(y|X) = E_{z \sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$ Taking the expectation wrt z, does not depend on z

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\right]$$ Baye's rule

$$= E_z \left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)} \frac{q_\phi(z|X,y)}{q_\phi(z|X,y)}\right]$$ Multiplying by a constant

$$= E_z\ [\log p_\theta(y|X,z)] - E_z \left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X)}\right] + E_z \left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X,y)}\right]$$

$$= E_z\ [\log p_\theta(y|X,z)] - KL\left[q_\phi(z|X,y)||p_\theta(z|X)\right] + KL\left[q_\phi(z|X,y)||p_\theta(z|X,y)\right]$$

KL divergence:

$$\boldsymbol{E\left[\log \frac{q}{p}\right] = KL[q||p]}$$

From our decoder network, we can estimate $p_\theta(X|z)$ by sampling z many times . (Reparametrization trick)

KL divergence between 2 gaussians = differentiable closed-form

KL divergence is intractable of $p_\theta(z|X,y)$

But by definition a KL divergence is always bigger or equal to 0!

Fin ML

Variational Autoencoders

Deriving the lower bound

We can now derive the log likelihood of our training data X, y:

$$\log p_\theta(y|X) = E_{z\sim q_\phi(z|X,y)}[\log p_\theta(y|X)]$$ Taking the expectation wrt z, does not depend on z

$$= E_z\left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\right]$$ Baye's rule

$$= E_z\left[\log \frac{p_\theta(y|X,z)\, p_\theta(z|X)}{p_\theta(z|X,y)}\frac{q_\phi(z|X,y)}{q_\phi(z|X,y)}\right]$$ Multiplying by a constant

$$= E_z\left[\log p_\theta(y|X,z)\right] - E_z\left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X)}\right] + E_z\left[\log \frac{q_\phi(z|X,y)}{p_\theta(z|X,y)}\right]$$

KL divergence:

$$\boldsymbol{E\left[\log \frac{q}{p}\right] = KL[q||p]}$$

$$= E_z\left[\log p_\theta(y|X,z)\right] - KL\left[q_\phi(z|X,y)||p_\theta(z|X)\right] + KL\left[q_\phi(z|X,y)||p_\theta(z|X,y)\right]$$

$L(X;\theta,\phi)$ ≥ 0

Instead of maximizing the likelihood, we maximize the lower bound

$\log p_\theta(y|X) \geq L(X;\theta,\phi)$ Lower bound $\longrightarrow$ $\theta^*,\phi^* = \text{argmax}\ L(X;\theta,\phi)$

Fin ML

Tutorial

Conditional VAE with truck dataset

Goals:

- Practice implementing a CVAE in the context of insurance

Dataset
Pytorch
Classes in python
CVAE Architecture

Training procedure

Results of the model

Fin ML

Dataset

Insurance problem

Our goal is to evaluate the risk a carrier truck to have an accident so that we can accurately price the insurance premium.

We have a database that contains ~50,000 datapoints (vehicle):

- Number of accidents: y
- information about this vehicle: X
  - How many infractions it had
  - Features of the vehicle (axles, size, etc.)
  - …

- Already preprocessed
- Already divided into train/validation/test
- Variable names are hidden

Fin ML

Pytorch

Handling the data

Tensor

```
1  data_train = pd.read_csv('data/data_train_.csv')
2  print(type(data_train))
3
4  X_train = data_train.drop(['y'], axis=1).values
5  print(type(X_train))
6
7  X_train_tensor = torch.tensor(X_train)
8  print(type(X_train_tensor))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'numpy.ndarray'>
<class 'torch.Tensor'>
```

PyTorch uses torch.tensor, rather than numpy arrays, so we need to convert our data.

DataLoader

```
14 train_loader = DataLoader(TensorDataset(
15                           torch.tensor(X_train, dtype=torch.float),
16                           torch.tensor(y_train, dtype=torch.float)),
17                       batch_size=32, shuffle=True
18                       )
19
```

Pytorch

Classes and nn.Module

Define the architecture of the network

Defines how to get the output of the network. It is called when you apply the network to an input variable

```
1  class Decoder(nn.Module):
2      def __init__(self, input_dim, output_dim, n_neurons, dropout):
3          super(Decoder, self).__init__()
4
5          n1 = n_neurons[0]
6          n2 = n_neurons[1]
7
8          self.decoder = nn.Sequential(
9              nn.Linear(input_dim, n1),
10             nn.Tanh(),
11             nn.Dropout(p=dropout),
12             nn.BatchNorm1d(n1),
13
14             nn.Linear(n1, n2),
15             nn.Tanh(),
16             nn.Dropout(p=dropout),
17             nn.BatchNorm1d(n2),
18         )
19
20         self.fp = nn.Linear(n2, output_dim)
21         self.fr = nn.Linear(n2, output_dim)
22
23     def forward(self, z, X):
24         inputs = torch.cat((z, X), dim=1)
25         out = self.decoder(inputs)
26         r = torch.exp(self.fr(out))
27         p = 1/(1 + torch.exp(-self.fp(out)))
28         return r, p
```

The *nn* modules in PyTorch provides us a higher level API to build and train deep network.

Results

Likelihood Comparison	
NB-CVAE	64.61
NB-GLM	64.31
P-CVAE	63.44

Negative Binomial CVAE

Nb.Accidents	P(y=0|X)	P(y=1|X)	P(y=2|X)	P(y=3|X)
0	88.05 (2.71)	9.41 (1.78)	1.91 (0.62)	0.46 (0.21)
1	85.33 (3.51)	11.08 (2.08)	2.55 (0.87)	0.70 (0.35)
2	84.44 (3.72)	11.63 (2.17)	2.76 (0.94)	0.78 (0.39)
3 +	83.58 (3.96)	12.22 (2.24)	2.96 (1.03)	0.84 (0.44)

Negative Binomial GLM

Nb.Accidents	P(y=0|X)	P(y=1|X)	P(y=2|X)	P(y=3|X)
0	87.62	10.84	1.35	0.16
1	87.62	10.84	1.35	0.17
2	87.55	10.89	1.36	0.17
3 +	87.34	11.05	1.40	0.18

Poisson CVAE

Nb.Accidents	P(y=0|X)	P(y=1|X)	P(y=2|X)	P(y=3|X)
0	83.61 (3.50)	14.87 (2.81)	1.41 (0.63)	0.10 (0.08)
1	83.58 (3.77)	14.88 (3.00)	1.42 (0.70)	0.10 (0.09)
2	83.65 (3.70)	14.83 (2.96)	1.41 (0.67)	0.10 (0.08)
3 +	83.25 (4.54)	14.77 (2.94)	1.40 (0.66)	0.11 (0.08)

Fin ML

Results

Include a couple of points to conclude