Generative Models

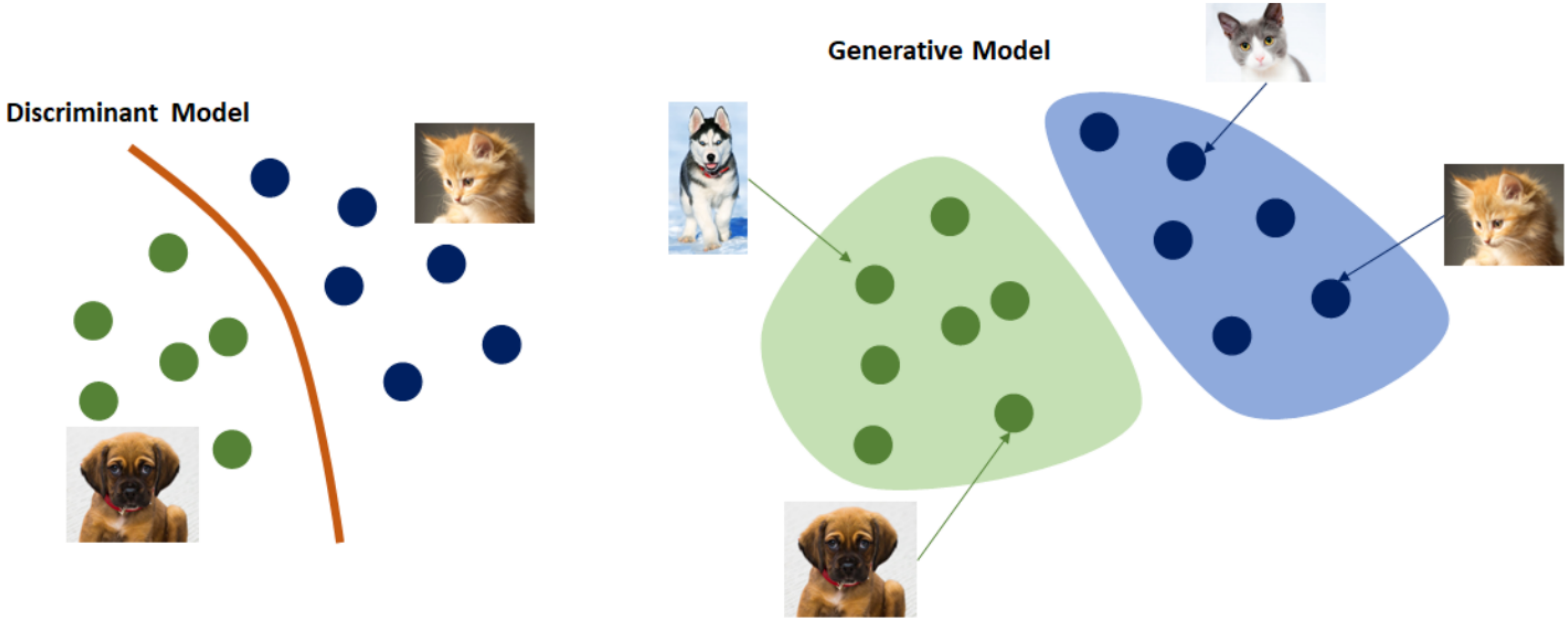
Generative Models

Two generative models facing neck to neck

- **VAE: rooted in Bayesian inference**

- GAN: rooted in game theory

Generative Models



Generative modeling: "*What I understand, I can **create***"

Generative Models

Generative models

Pros:

- Good at unsupervised learning
- We get the underlying idea of what each class data is built on

Cons:

- Computationally expensive

Examples: Naive Bayes, Gaussian mixture model, Hidden Markov Models (HMM)

Discriminant models

Pros:

- Need less data
- Computationally cheaper

Cons:

- Not useful for unsupervised learning
- Can be more difficult to interpret

Examples: Logistic regression, SVM, Neural Networks, DT, RF, KNN

Generative Models

The entire data set can be viewed as a sample drawn from the following generative process:

For $i = 1$ to N

- Pick a cluster k under distribution $\{\pi_1, \ldots, \pi_K\}$
- Generate a point according to k^{th} cluster's distribution

Challenge: estimate $\{\pi_1, \ldots, \pi_K\}$ and parameters for each cluster such that the likelihood of the dataset is maximized

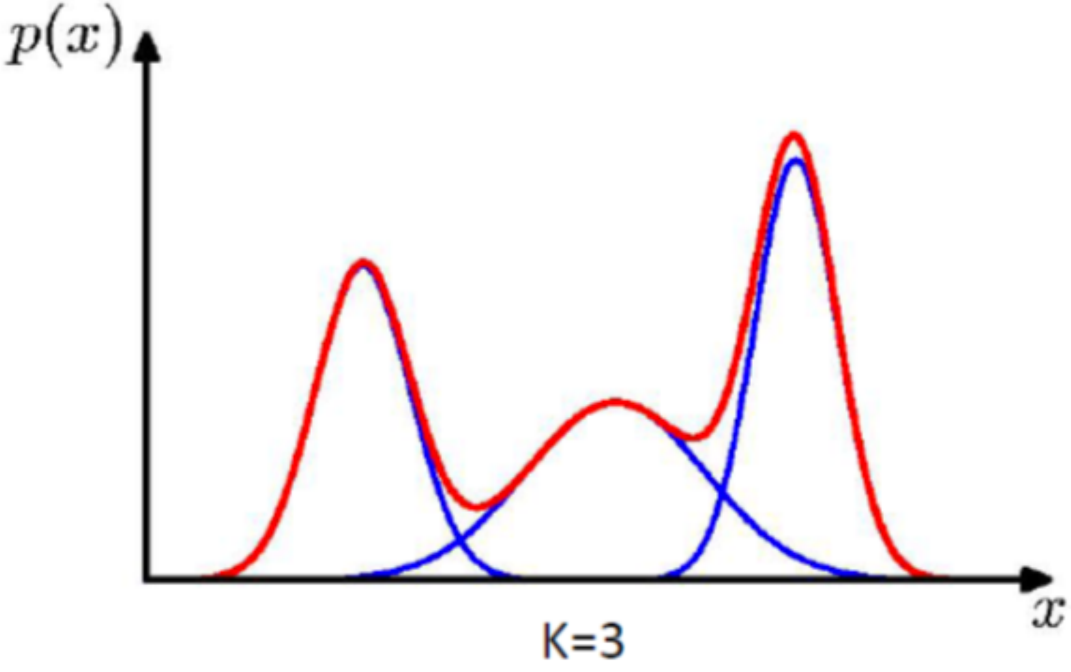
Generative Models

1D Mixture of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \underbrace{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{Component}}$$

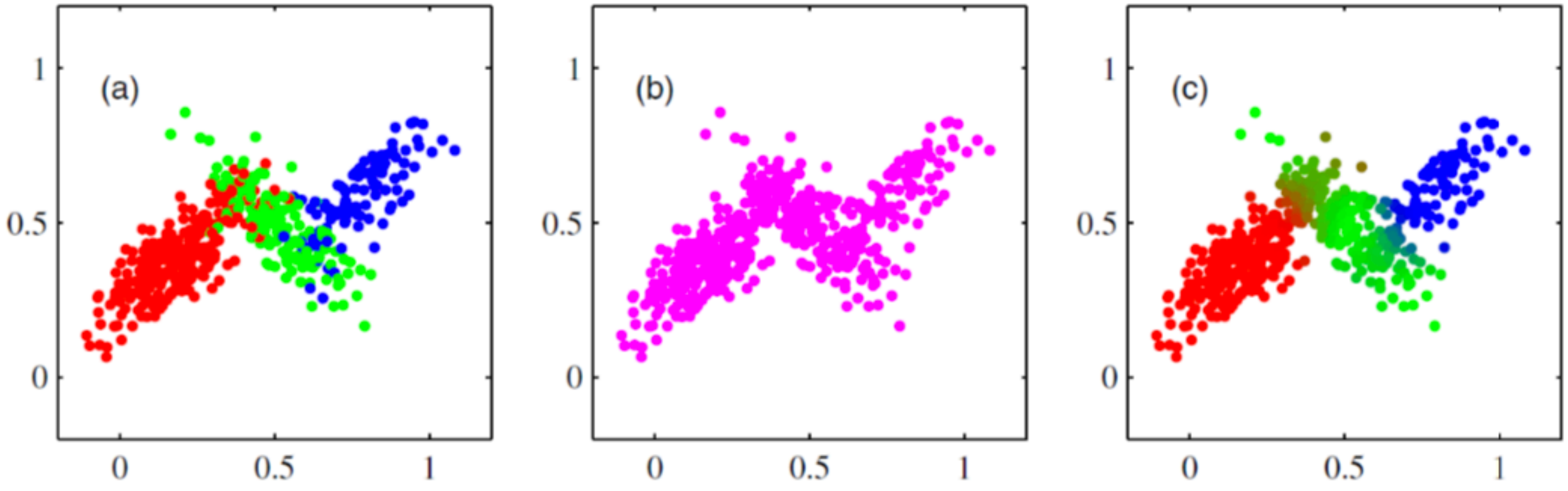
Mixing coefficient

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$



Generative Models

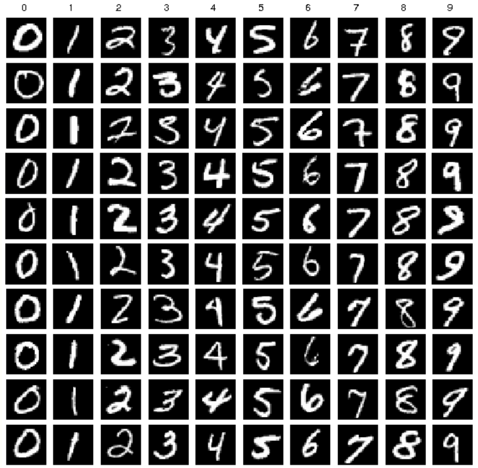
2D mixture of Gaussians



(a) Ground truth
(b) Training data
(c) Result from the GMM model

Generative Models

What about MNIST?



A very simple model: $\mathbf{z} \in \{0, \ldots, 9\}$ indicates the digit.

- $p(\mathbf{z} = k) = \pi_k$, with $\pi_k \geq 0$, $\sum_k \pi_k = 1$
- $p(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

In total, we get a Gaussian Mixture Model with 10 components:

$$p(\mathbf{x} \mid \theta) = \int p(\mathbf{x}, \mathbf{z} \mid \theta)\, \mathrm{d}\mathbf{z} = \sum_{k=0}^{9} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

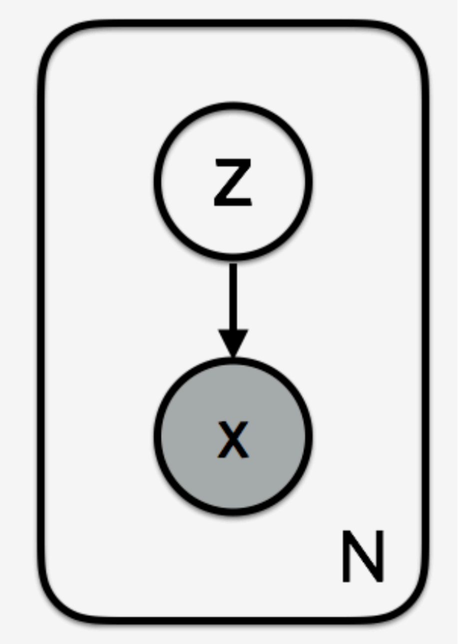
$\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \mid k = 0, \ldots, 9\}$

Generative Models

- Assume data is generated by a hidden variable Z:

For each datapoint i:

- Draw latent variables $z_i \sim p(z)$
- Draw datapoint $x_i \sim p(x|z)$



- We only observe X under the joint distribution:

$$p(x, z) = p(x|z)p(z)$$

- Learning, typically by maximum likelihood:

$$\theta^{\mathrm{ML}} = \underset{\theta}{\operatorname{argmax}}\, P(x_1, \ldots, x_n|\theta)$$

Generative Models

Why to use a Generative Models?

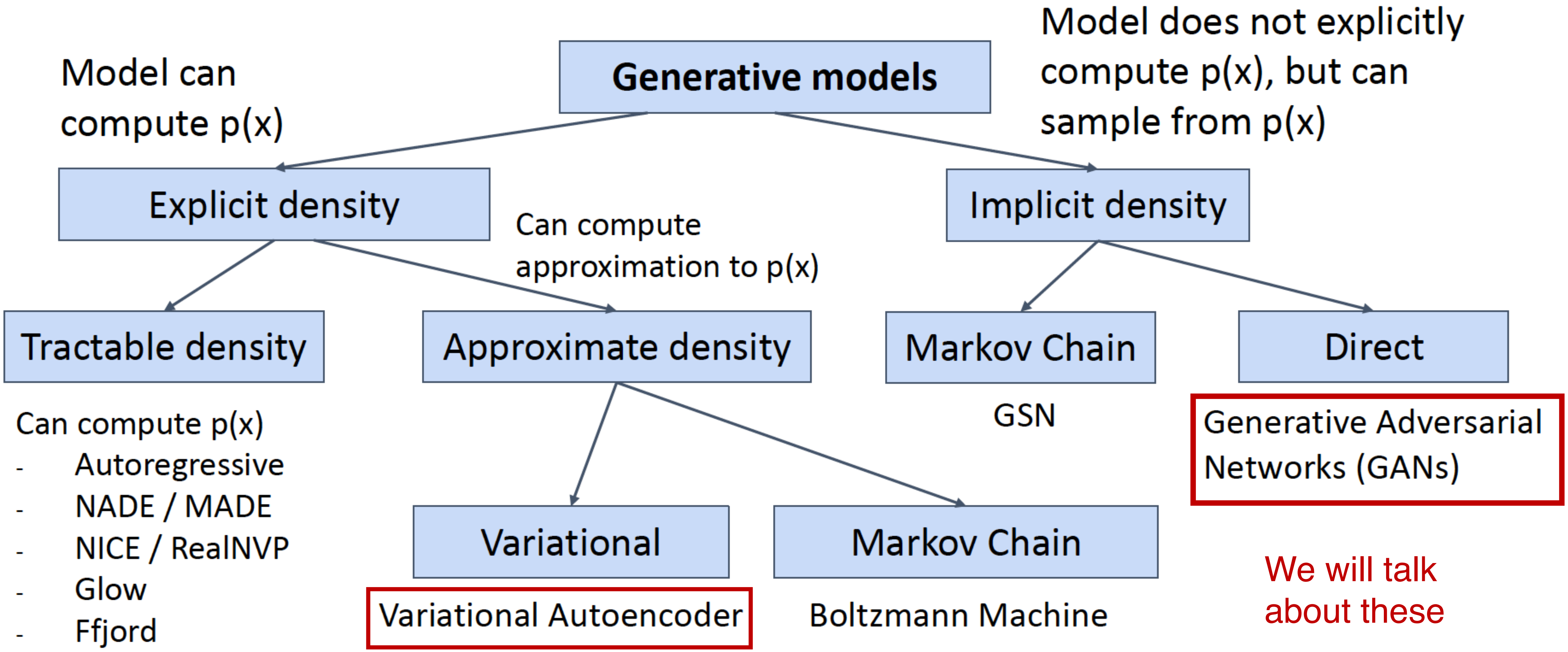
- More expressive power: GMM is better than a single Gaussian
- Can generate new data

The primary role of the latent variables is to allow a complicated distribution over the observed data to be represented in terms of simpler conditional distributions.

Generative Model Tasks:

- Approximate posterior inference over z: given an input x, what are its latent factors? → $p(Z|X)$
- Approximate marginal inference over x*: what is $p(x^*)$

Taxonomy of Generative Models



Variational Inference

- Because Z is hidden, we can't model it directly. We use Bayes rule:

$$p(\mathbf{Z}|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{Z})\ p(\mathbf{Z})}{p(\mathbf{X})}$$

- Examine the denominator $p(x)$. This is called the evidence, and we can calculate it by marginalizing out the latent variables:

$$p(\mathbf{X}) = \int p(\mathbf{X}|\mathbf{Z})\ p(\mathbf{Z})\ d\mathbf{Z}$$

Unfortunately, this integral requires exponential time to compute as it needs to be evaluated over all configurations of latent variables.

Variational Inference

- Variational Inference approximates *p(Z|X)* using a function *q(Z)*

- Other approaches (not covered in this course):
  - MCMC: asymptotically exact, but computationally expensive
  - Gibbs Sampling

Variational Inference

Kullback–Leibler divergence measures the difference between two probability distributions:

For discrete distributions *P* and *Q* defined on the same probability space, the KL divergence of *P* and *Q* is defined as:

$$D_{\mathrm{KL}}(P \parallel Q) = -\sum_{x \in \mathcal{X}} P(x) \log\left(\frac{Q(x)}{P(x)}\right)$$

which is equivalent to

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

Variational Inference

- Variational Inference approximates *p(Z|X)* using a function *q(Z)* by

Minimize $$KL\left[q(Z)\|p(Z|X)\right]$$

$$
\begin{aligned}
KL\left[q(Z) \| p(Z|X)\right] &= \int_Z q(Z) \log \frac{q(Z)}{p(Z|X)} \\
&= -\int_Z q(Z) \log \frac{p(Z|X)}{q(Z)} \\
&= -\left(\int_Z q(Z) \log \frac{p(X,Z)}{q(Z)} - \int_Z q(Z) \log p(X) \right) \\
&= -\int_Z q(Z) \log \frac{p(X,Z)}{q(Z)} + \log p(X) \int_Z q(Z) \\
&= -L + \log p(X)
\end{aligned}
$$

ELBO (Evidence Lower Bound)

Therefore:

$\boxed{(\text{max ELBO wrt } q) \iff (\ q(Z) \text{ is as close as possible to } p(Z|X)\ )}$

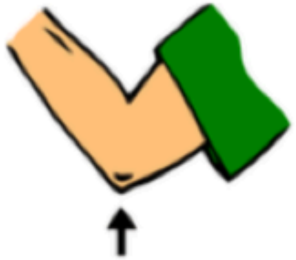
Variational Inference

- Variational Inference approximates *p(Z|X)* using a function *q(Z)* by

Minimize $KL\left[q(Z) \| p(Z|X)\right]$

- We just showed the above is equivalent to max ELBO.

$$\int_Z q(Z) \log \frac{p(X,Z)}{q(Z)} \quad \text{OR} \quad \mathbb{E}_{q(Z)} \log \frac{p(X|Z)p(Z)}{q(Z)}$$



We can write the ELBO in a few different ways

$$
\begin{aligned}
\text{ELBO} = \quad & \mathbb{E}_{q(Z)} \log \frac{p(X|Z)p(Z)}{q(Z)} \\
= \quad & \mathbb{E}_{q(Z)} \log p(X|Z) + \mathbb{E}_{q(Z)} \log \frac{p(Z)}{q(Z)} \\
= \quad & \mathbb{E}_{q(Z)} \log p(X|Z) - KL(q(Z)||p(Z))
\end{aligned}
$$

= reconstructed loglikelihood - a KL penalty (regularizer) term

The first term represents the reconstruction loss and the second term ensures that our learned distribution *q(Z)* is similar to the **prior** distribution *p(Z)*.

We now have followed the recipe for variational inference.

- A probability model $p(X, Z)$ of latent variables and data
- A variational family $q(Z)$ for the latent variables to approximate our posterior $p(Z|X)$
- Use the variational inference algorithm (max ELBO) to learn the parameters

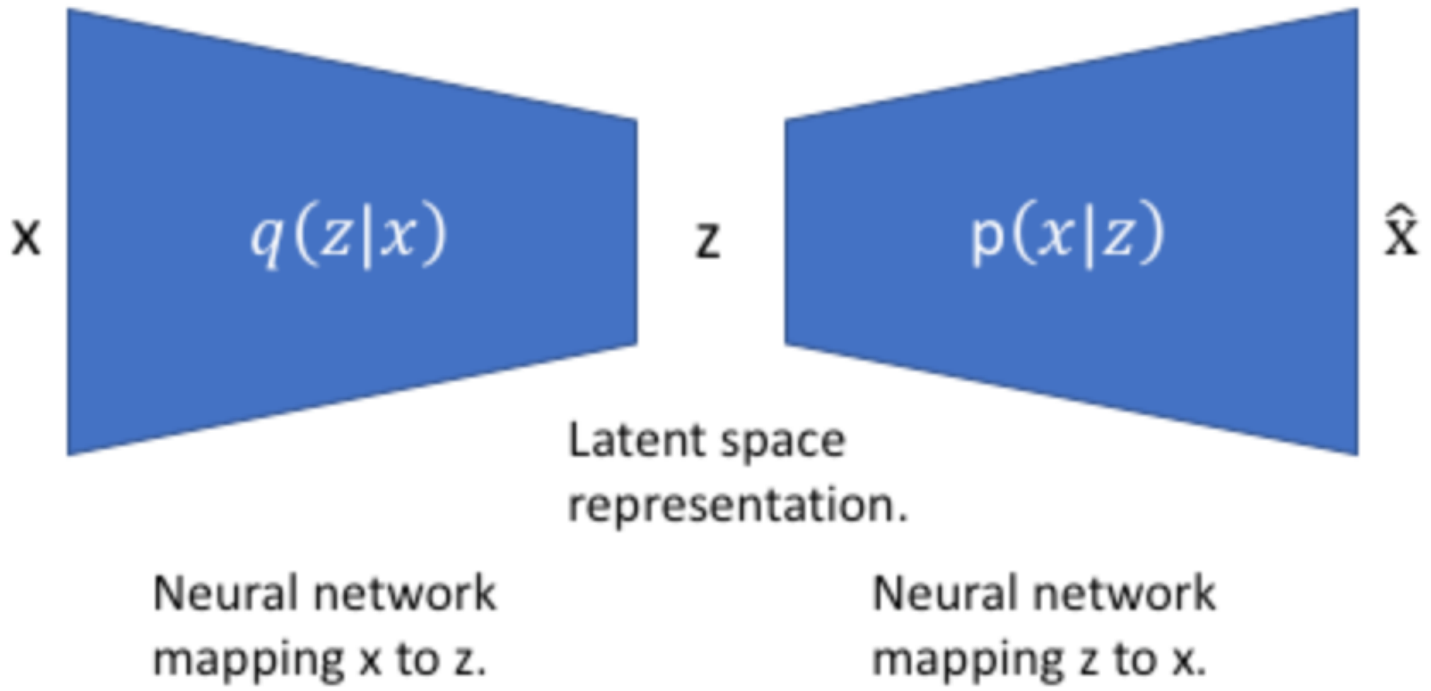
We now have followed the recipe for variational inference.

- A probability model *p(X, Z)* of latent variables and data
- A variational family *q(Z)* for the latent variables to approximate our posterior *p(Z|X)*
- Use the variational inference algorithm (max ELBO) to learn the parameters

What's the connection with VAE?

VAE and Variational Inference

We can use neural networks to model *q(Z)* and *p(X|Z)*:



VAE and Variational Inference

VAE loss function in the sample code given in BB

```
# this is the model
vae = Model(x, x_decoded_mean)

# constrution and kl divergence
xent_loss = K.sum(K.binary_crossentropy(x, x_decoded_mean), axis=-1)
kl_loss = - 0.5 * K.sum(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
vae_loss = K.mean(xent_loss + kl_loss)

# add loss
vae.add_loss(vae_loss)
vae.compile(optimizer='rmsprop')
vae.summary()
```