Generative Models (Part I)

(2026)

Lecturer: Shaofeng Zhang

sherrylone.github.io



Contents

- Introduction
- Autoregressive Models
- Variational Autoencoder
- Generative Adversarial Net
- Diffusion Models



Contents

- **Introduction**
- Autoregressive Models
- Variational Autoencoder
- Generative Adversarial Net
- Diffusion Models



Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: (x, y) Learn a function to map x -> y

Examples: Classification, regression, Object detection, semantic Segmentation, image captioning, etc.

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: (x, y) Learn a function to map x -> y

Examples: Classification, regression, Object detection, semantic Segmentation, image captioning, etc.



Classification

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: (x, y) Learn a function to map x -> y

Examples: Classification, regression, Object detection, semantic Segmentation, image captioning, etc.



A cat sitting on a suitcase on the floor

Image captioning

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: (x, y) Learn a function to map x -> y

Examples: Classification, regression, Object detection, semantic Segmentation, image captioning, etc.



DOG, DOG, CAT

Object Detection

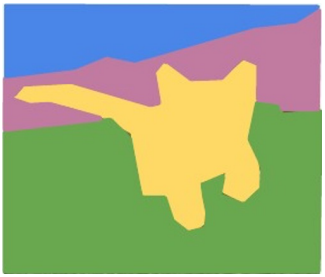
Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: (x, y) Learn a function to map x -> y

Examples: Classification, regression, Object detection, semantic Segmentation, image captioning, etc.



SKY, GRASS, CAT, TREE

Semantic Segmentation

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: (x, y) Learn a function to map x -> y

Examples: Classification, regression, Object detection, semantic Segmentation, image captioning, etc.

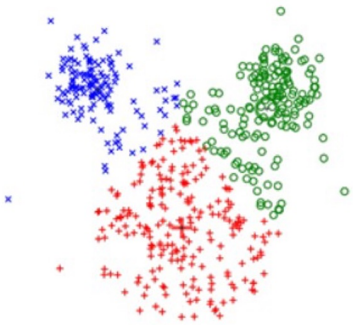
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn hidden structure in data

Examples: Clustering, dimensionality Reduction, density estimation, etc.

Supervised vs Unsupervised Learning



K-means clustering

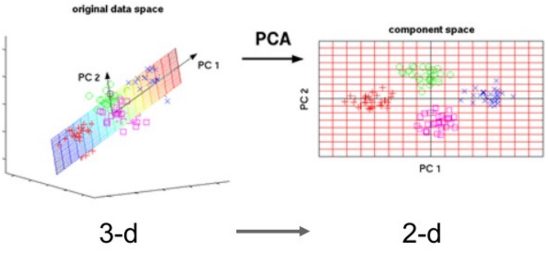
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn hidden structure in data

Examples: Clustering, dimensionality Reduction, density estimation, etc.

Supervised vs Unsupervised Learning



Principal Component Analysis (Dimensionality reduction)

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn hidden structure in data

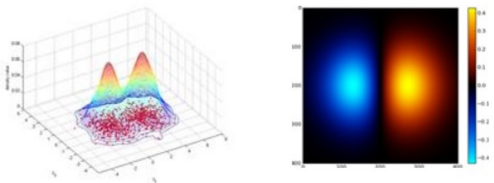
Examples: Clustering, dimensionality Reduction, density estimation, etc.

Supervised vs Unsupervised Learning



Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation



2-d density estimation
Modeling $p(x)$

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn hidden structure in data

Examples: Clustering, dimensionality Reduction, density estimation, etc.

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$

Data: x



Label: y
Cat

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$

Data: x



Label: y
Cat

Probability Recap:

Density Function
$p(x)$ assigns a positive number to each possible x; higher numbers mean x is more likely.

Density functions are **normalized:**

$$\int_X p(x)dx = 1$$

Different values of x **compete** for density.

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

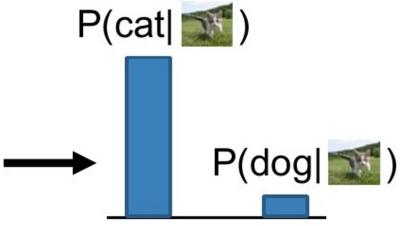
Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$

Data: x



Label: y
Cat



$$\int_X p(x)dx = 1$$

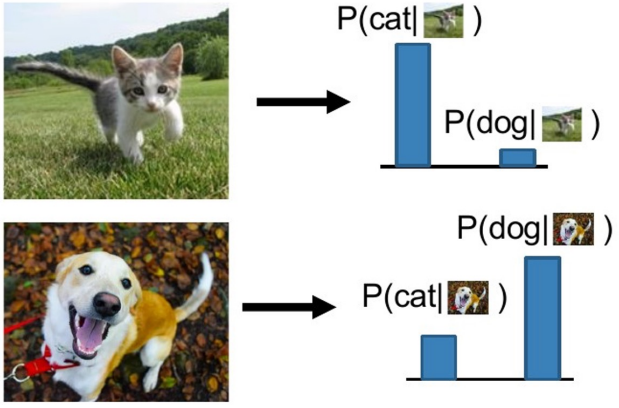
Different values of x **compete** for density.

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$



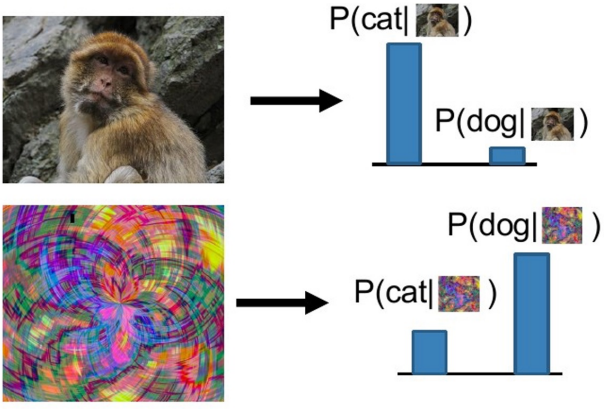
Possible **labels** for each image compete for probability.
No competition between **images**.

Generative vs Discriminative Models

Discriminative Model: Learn a probability distribution $p(y|x)$

Generative Model: Learn a probability distribution $p(x)$

Conditional Generative: Learn a probability distribution $p(x|y)$



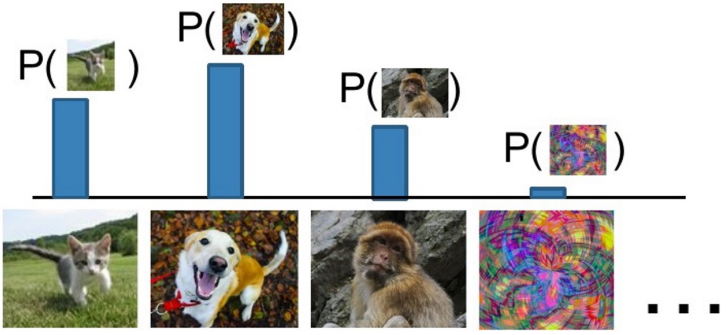
No way to handle unreasonable inputs; must give a label distribution for all possible inputs.

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$



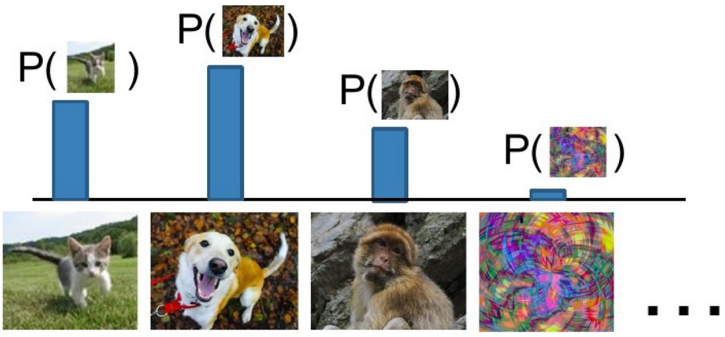
All possible **images** compete for probability mass

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$



All possible **images** compete for probability mass.

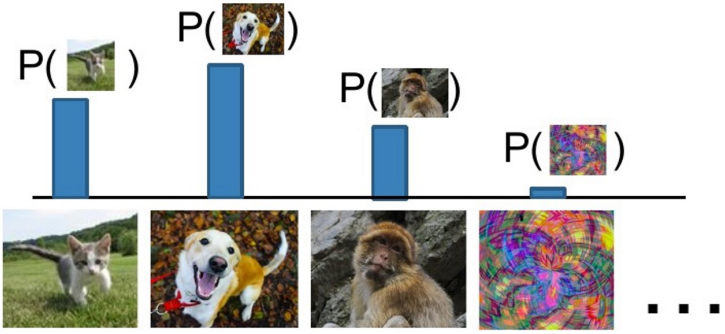
Requires deep understanding: Is a dog more likely to sit or stand? Is a 3-legged dog more likely than 3-armed monkey?

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$



All possible **images** compete for probability mass.

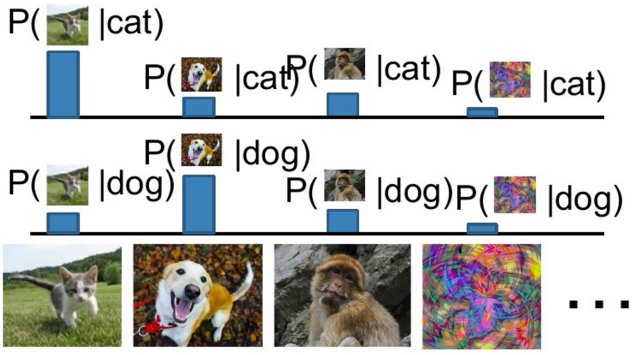
Model can ”reject” unreasonable inputs by giving them small probability mass

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$



Each possible **label** induces a competition across all possible **images**.

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$

Recall Bayes' Rule:

$$P(x|y) = \frac{P(y|x)}{p(y)} p(x)$$

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$

Recall Bayes' Rule:

$$P(x|y) = \frac{P(y|x)}{p(y)} p(x)$$

- $P(x|y)$: Conditional Generative Model
- $P(y|x)$: Discriminative Model
- $p(y)$: Prior over labels
- $p(x)$: Unconditional Generative Model

We can build a conditional generative model from other components ··· but not common in practice.

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

→ Assign labels to data
Feature learning (with labels)

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

→ Assign labels to data
Feature learning (with labels)

Generative Model:
Learn a probability distribution $p(x)$

→ Detect outliers
Feature learning (without labels)
Sample to generate new data

Conditional Generative:
Learn a probability distribution $p(x|y)$

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

→ Assign labels to data
Feature learning (with labels)

Generative Model:
Learn a probability distribution $p(x)$

→ Detect outliers
Feature learning (without labels)
Sample to generate new data

Conditional Generative:
Learn a probability distribution $p(x|y)$

→ Assign labels while rejecting outliers
Sample to generate data from labels

Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$

Generative models means either of these; conditional generative models are most common in practice

Why Generative Models?

Modeling ambiguity: If there are many possible outputs of x for an input y, we want to model $p(x|y)$.

Language Modeling: Produce output text x from input text y

Write me a short thyming poem about generative models

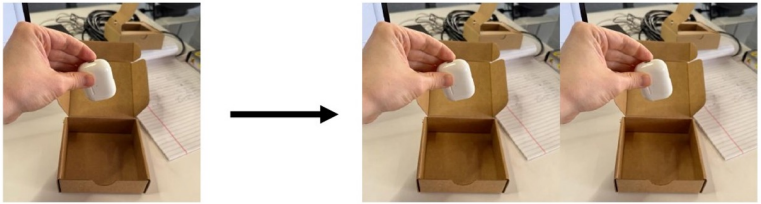


They sample from a learned P, A distribution-structured, free. Each token comes conditionally, on all the ones that used to be.

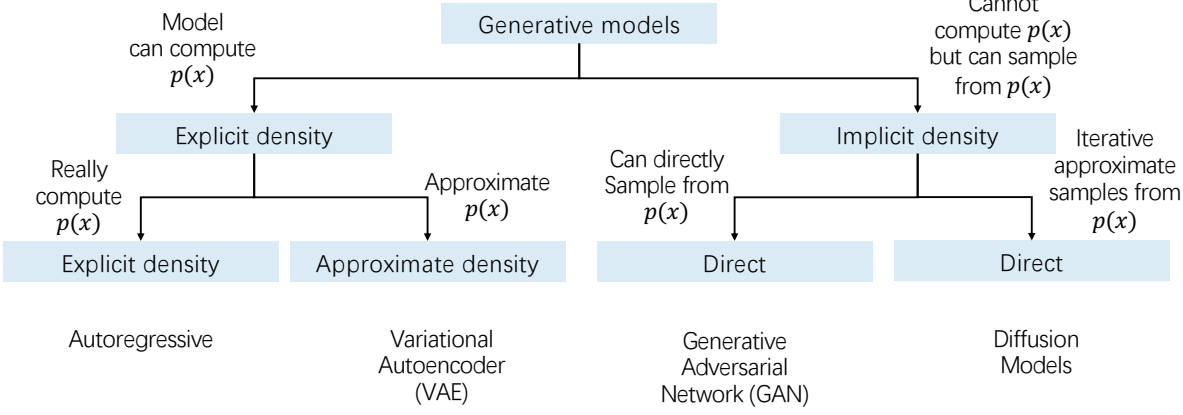
Why Generative Models?

Modeling ambiguity: If there are many possible outputs of x for an input y, we want to model $p(x|y)$.

Image to Video: What happens next?



Taxonomy of Generative Models?



Contents

- Introduction
- **Autoregressive Models**
- Variational Autoencoder
- Generative Adversarial Net
- Diffusion Models



Maximum Likelihood Estimation

Goal: Write down an explicit function for $p(x) = f(x, W)$

Maximum Likelihood Estimation

Goal: Write down an explicit function for $p(x) = f(x, W)$

Given dataset $x^{(1)}, x^{(2)} \dots, x^{(N)}$, train the model by solving:

$$W^* = \arg\max \prod_i p(x^{(i)})$$

Maximize the probability of training data
(Maximum likelihood estimation)

Maximum Likelihood Estimation

Goal: Write down an explicit function for $p(x) = f(x, W)$

Given dataset $x^{(1)}, x^{(2)} \dots, x^{(N)}$, train the model by solving:

$$W^* = \arg\max \prod_i p(x^{(i)})$$

Maximize the probability of training data (Maximum likelihood estimation)

$$= \arg\max \sum_i \log p(x^{(i)})$$

Log trick: Swap product and sum

Maximum Likelihood Estimation

Goal: Write down an explicit function for $p(x) = f(x, W)$

Given dataset $x^{(1)}, x^{(2)} \dots, x^{(N)}$, train the model by solving:

$$W^* = \arg\max \prod_i p(x^{(i)})$$

Maximize the probability of training data (Maximum likelihood estimation)

$$= \arg\max \sum_i \log p(x^{(i)})$$

Log trick: Swap product and sum

$$= \arg\max \sum_i \log f(x^{(i)}, W)$$

This is our loss function.
Maximize it with gradient descent

Autoregressive Models

Goal: Write down an explicit function for $p(x) = f(x, W)$

Assume x is a sequence: $x = (x_1, x_2, \ldots, x_T)$

Autoregressive Models

Goal: Write down an explicit function for $p(x) = f(x, W)$

Assume x is a sequence: $x = (x_1, x_2, \ \dots, \ x_T)$

Use the chain rule of probability:

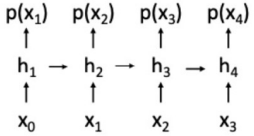
$$\begin{aligned} p(x) &= p(x_1, x_2, x_3, \dots, x_T) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots \\ &= \prod_{t=1}^{T} p(x_t|x_1, \dots, x_{t-1}) \end{aligned}$$

Autoregressive Models

Goal: Write down an explicit function for $p(x) = f(x, W)$

Assume x is a sequence: $x = (x_1, x_2, \ \dots, \ x_T)$

We have already seen this!



Language modeling with RNN

Use the chain rule of probability:

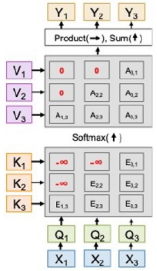
$$
\begin{aligned}
p(x) &= p(x_1, x_2, x_3, \dots, x_T) \\
&= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots \\
&= \prod_{t=1}^{T} p(x_t|x_1, \dots, x_{t-1})
\end{aligned}
$$

Autoregressive Models

Goal: Write down an explicit function for $p(x) = f(x, W)$

Assume x is a sequence: $x = (x_1, x_2, \ldots, x_T)$

Language modeling with masked Transformer



Use the chain rule of probability:

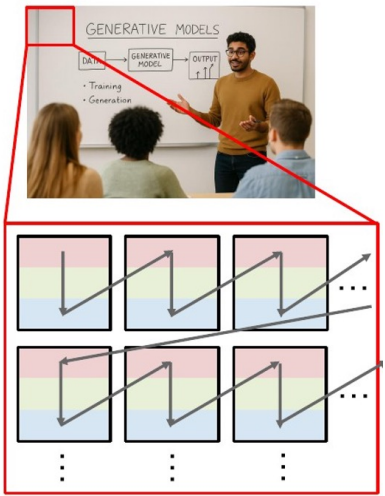
$$
\begin{aligned}
p(x) &= p(x_1, x_2, x_3, \ldots, x_T) \\
&= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \ldots \\
&= \prod_{t=1}^{T} p(x_t|x_1, \ldots, x_{t-1})
\end{aligned}
$$

Autoregressive Models of Images

Treat an image as a sequence of 8-bit subpixel values (scanline order)

Predict each subpixel as a classification among 256 values [0···255]

Model with an RNN or Transformer



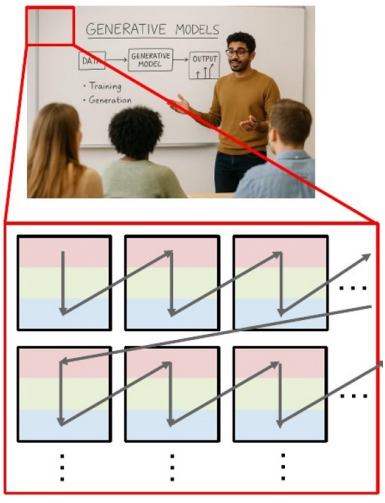
Autoregressive Models of Images

Treat an image as a sequence of 8-bit subpixel values (scanline order)

Predict each subpixel as a classification among 256 values [0···255]

Model with an RNN or Transformer

Problem: Too expensive. 1024x1024 image is a sequence of 3M subpixels.



Autoregressive Models of Images

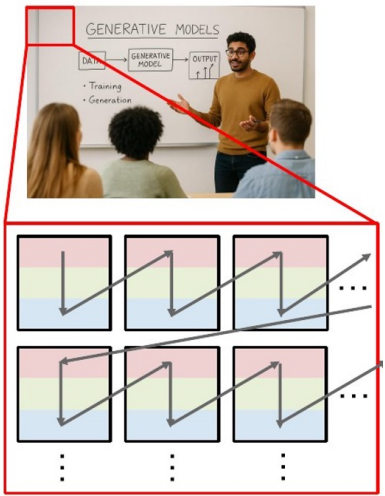
Treat an image as a sequence of 8-bit subpixel values (scanline order)

Predict each subpixel as a classification among 256 values [0···255]

Model with an RNN or Transformer

Problem: Too expensive. 1024x1024 image is a sequence of 3M subpixels.

Solution (jumping ahead): Model as sequence of tiles, not sequence of subpixels.



Contents

- Introduction
- Autoregressive Models
- **Variational Autoencoder**
- Generative Adversarial Net
- Diffusion Models



Variational Autoencoders

PixelRNN / PixelCNN explicitly parameterizes density function with a neural network, so we can train to maximize likelihood of training data:

$$p_W(x) = \prod_{t=1}^{T} p_W(x_t | x_1, \dots, x_{t-1})$$

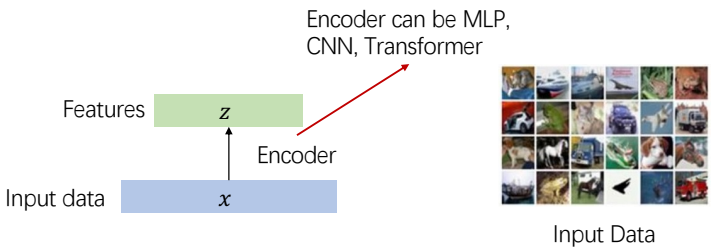
Variational Autoencoders (VAE) define an **intractable density** that we cannot explicitly compute or optimize

But we will be able to directly optimize a **lower bound** on the density

(Non-Variational) Autoencoders

Idea: Unsupervised method for learning to extract feature z from input x, without labels

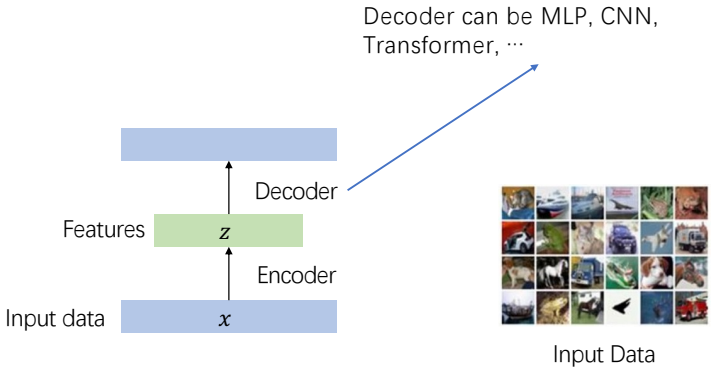
Features should extract useful information (object identity, appearance, scene type, etc) that can be used for downstream tasks.



(Non-Variational) Autoencoders

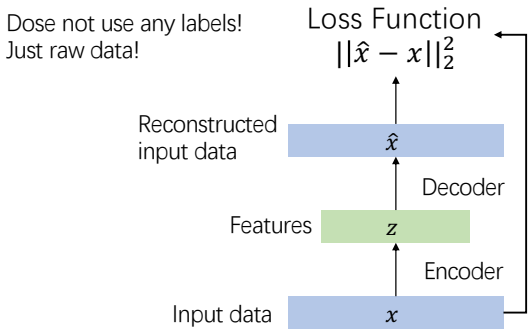
Problem: How can we learn without labels?

Solution: Reconstruct the input data with a decoder.



(Non-Variational) Autoencoders

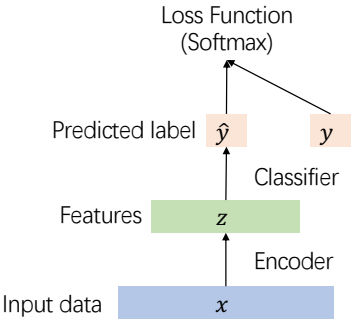
Loss: L2 distance between input and reconstructed data.





(Non-Variational) Autoencoders

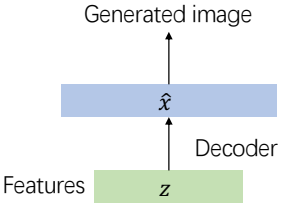
After training, can use the encoder for downstream tasks





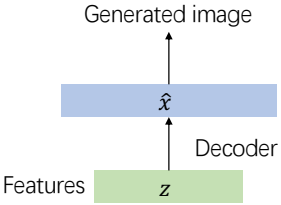
(Non-Variational) Autoencoders

If we could generate new z, could use the decoder to generate images



(Non-Variational) Autoencoders

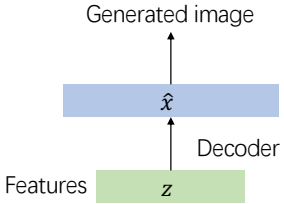
If we could generate new z, could use the decoder to generate images



Problem: Generating new z is not any easier than generating new x

(Non-Variational) Autoencoders

If we could generate new z, could use the decoder to generate images



Problem: Generating new z is not any easier than generating new x

Solution: What if we force all z to come from a known distribution?

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

Intuition: x is an image, z is latent factors used to generate x: attributes, orientation, etc.

Variational Autoencoders

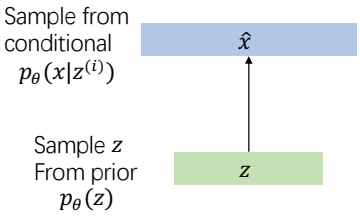
Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

After training, sample new data like this:

Intuition: x is an image, z is latent factors used to generate x: attributes, orientation, etc.



Variational Autoencoders

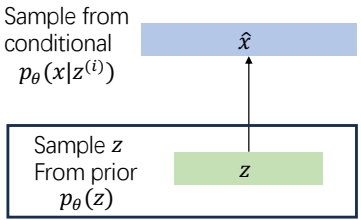
Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

After training, sample new data like this:

Intuition: x is an image, z is latent factors used to generate x: attributes, orientation, etc.



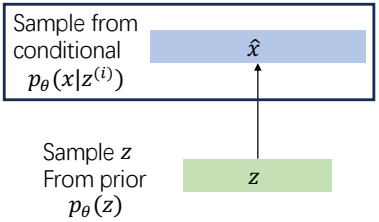
Assume simple prior $p(z)$, e.g. Gaussian

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

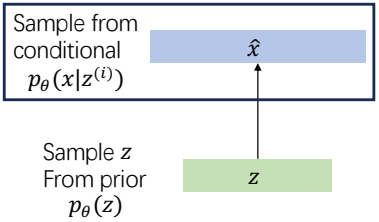
If we had a dataset of (x, z) then train a conditional generative model $p(x|z)$

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

If we had a dataset of (x, z) then train a conditional generative model $p(x|z)$

We don't observe z, so **marginalize**:

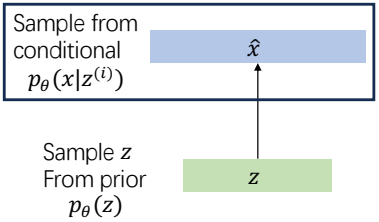
$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)\, p_\theta(z)dz$$

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

If we had a dataset of (x, z) then train a conditional generative model $p(x|z)$

We don't observe z, so **marginalize**:

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

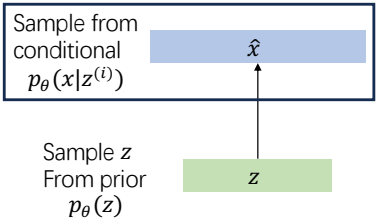
Ok, we can compute this with the decoder

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

If we had a dataset of (x, z) then train a conditional generative model $p(x|z)$

We don't observe z, so **marginalize**:

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

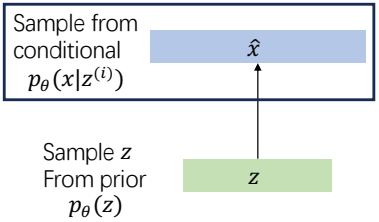
Ok, we assumed Gaussian prior for z

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

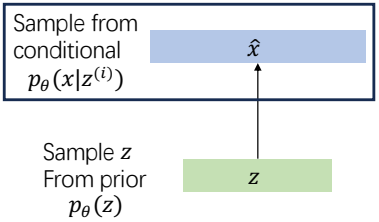
$$p_\theta(x) = \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z)$$

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

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

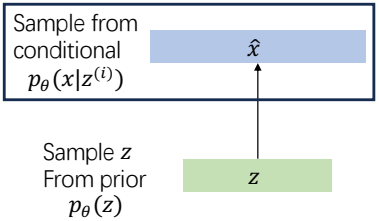
Ok, we can compute this with the decoder

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

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

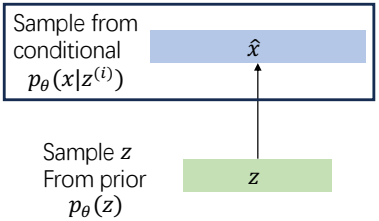
Ok, we assumed Gaussian prior for z

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

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

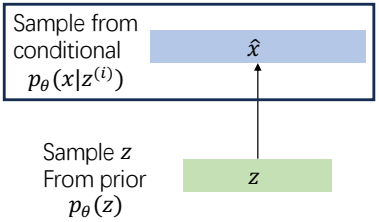
Problem: no way to compute this

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

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

Problem: no way to compute this

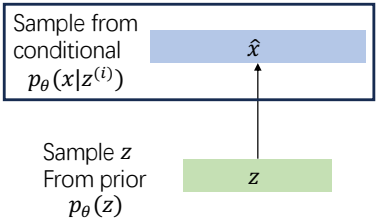
Solution: Train another network that learns $q_\phi(z|x) \approx p_\theta(z|x)$

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

$$p_\theta(x) = \frac{p_\theta(x|z)}{p_\theta(z|x)}p(z) \approx \frac{p_\theta(x|z)}{q_\phi(z|x)}p(z)$$

Solution: Train another network that learns $q_\phi(z|x) \approx p_\theta(z|x)$

Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

Encoder Network:
Input latent x,
Output distribution over latent codes z

If we can ensure that
$q_\phi(z|x) \approx p_\theta(x|z)$,

then we can approximate
$p_\theta(x) \approx \frac{p_\theta(x|z)}{q_\phi(z|x)} p(z)$,

Idea: Jointly train both encoder and decoder.



Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

Encoder Network:
Input latent x,
Output distribution over latent codes z

If we can ensure that
$q_\phi(z|x) \approx p_\theta(x|z)$,

then we can approximate
$p_\theta(x) \approx \frac{p_\theta(x|z)}{q_\phi(z|x)} p(z)$,

Idea: Jointly train both encoder and decoder.

Aside: How to output probability distributions from neural networks?

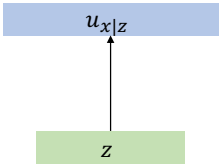
Networks outputs mean (and std) of a (diagonal) distribution.



Variational Autoencoders

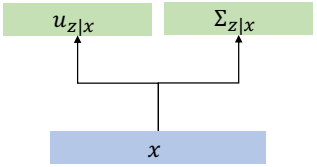
Decoder Network:
Input latent code z,
Output distribution over data x

$p_\theta(x|z) = N(u_{x|z}, \sigma^2)$



Encoder Network:
Input latent x,
Output distribution over latent codes z

$q_\phi(z|x) = N(u_{z|x}, \Sigma_{z|x})$



If we can ensure that
$q_\phi(z|x) \approx p_\theta(x|z)$,

then we can approximate
$p_\theta(x) \approx \frac{p_\theta(x|z)}{q_\phi(z|x)} p(z)$,

Idea: Jointly train both encoder and decoder.

Aside: How to output probability distributions from neural networks?

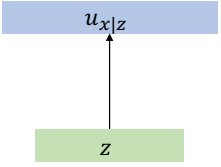
Networks outputs mean (and std) of a (diagonal) distribution.

Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

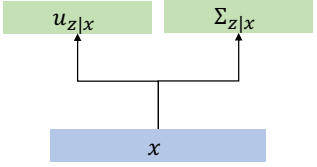
$$p_\theta(x|z) = N(u_{x|z}, \sigma^2)$$

$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2}||x - u||_2^2 + C_2$$



Encoder Network:
Input latent x,
Output distribution over latent codes z

$$q_\phi(z|x) = N(u_{z|x}, \Sigma_{z|x})$$



If we can ensure that
$q_\phi(z|x) \approx p_\theta(x|z)$,

then we can approximate
$p_\theta(x) \approx \frac{p_\theta(x|z)}{q_\phi(z|x)} p(z)$,

Idea: Jointly train both encoder and decoder.

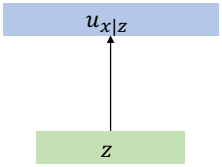
Maximizing $\log p_\theta(x|z)$ is equivalent to minimizing L_2 distance between x and network output!

Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

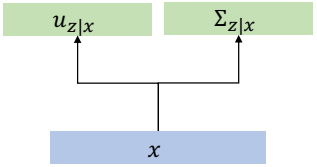
$$p_\theta(x|z) = N(u_{x|z}, \sigma^2)$$

$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2}||x - u||_2^2 + C_2$$



Encoder Network:
Input latent x,
Output distribution over latent codes z

$$q_\phi(z|x) = N(u_{z|x}, \Sigma_{z|x})$$



If we can ensure that
$q_\phi(z|x) \approx p_\theta(x|z)$,

then we can approximate
$p_\theta(x) \approx \frac{p_\theta(x|z)}{q_\phi(z|x)} p(z)$,

Idea: Jointly train both encoder and decoder.

Question: What's out training objective?

Variational Autoencoders (ELBO)

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

Bayes' Rules

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x)}{p_\theta(x|z) q_\phi(z|x)},$$

Multiply top and bottom by $q_\phi(z|x)$

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x) p(z)}{p_\theta(z|x) q_\phi(z|x)},$$
$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)},$$

Logarithms + rearranging

Variational Autoencoders (ELBO)

$$\begin{aligned}\log p_\theta(x) &= \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x) p(z)}{p_\theta(z|x) q_\phi(z|x)}, \\ &= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}, \\ &= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]\end{aligned}$$

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$

We can wrap in an expectation since it doesn't depend on z.

Variational Autoencoders (ELBO)

$$
\begin{aligned}
\log p_\theta(x) &= \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x) p(z)}{p_\theta(z|x) q_\phi(z|x)}, \\
&= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}, \\
&= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] \\
&= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}\left(q_\phi(z|x), p_\theta(z|x)\right)
\end{aligned}
$$

Variational Autoencoders (ELBO)

$$
\begin{aligned}
\log p_\theta(x) &= \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x) p(z)}{p_\theta(z|x) q_\phi(z|x)}, \\
&= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}, \\
&= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] \\
&= E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}\left(q_\phi(z|x), p_\theta(z|x)\right)
\end{aligned}
$$

Data reconstruction: x => encoder => decoder should reconstruct x.
Can compute in closed form for Gaussians.

Variational Autoencoders (ELBO)

$$
\begin{aligned}
\log p_\theta(x) &= \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x) p(z)}{p_\theta(z|x) q_\phi(z|x)}, \\
&= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}, \\
&= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] \\
&= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}\left(q_\phi(z|x), p_\theta(z|x)\right)
\end{aligned}
$$

Prior: Encoder output should match prior over z.
Can compute in closed form for Gaussians.

Variational Autoencoders (ELBO)

$$
\begin{aligned}
\log p_\theta(x) &= \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x) p(z)}{p_\theta(z|x) q_\phi(z|x)}, \\
&= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}, \\
&= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] \\
&= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}\left(q_\phi(z|x), p_\theta(z|x)\right)
\end{aligned}
$$

Posterior Approximation: Encoder output $q_\phi(z|x)$ should match $p_\theta(z|x)$.
We cannot compute this for Gaussians.

Variational Autoencoders (ELBO)

$$
\begin{aligned}
\log p_\theta(x) &= \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x) p(z)}{p_\theta(z|x) q_\phi(z|x)}, \\
&= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}, \\
&= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] \\
&= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}\left(q_\phi(z|x), p_\theta(z|x)\right)
\end{aligned}
$$

Posterior Approximation: Encoder output $q_\phi(z|x)$ should match $p_\theta(z|x)$.
KL is always >= 0, so we can drop it to get a lower bound on likelihood.

Variational Autoencoders (ELBO)

$$
\begin{aligned}
\log p_\theta(x) &= \log \frac{p_\theta(x|z)}{p_\theta(z|x)} p(z) = \log \frac{p_\theta(x|z) q_\phi(z|x) p(z)}{p_\theta(z|x) q_\phi(z|x)}, \\
&= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}, \\
&= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] \\
&= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}\left(q_\phi(z|x), p_\theta(z|x)\right)
\end{aligned}
$$

$$
\boxed{\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)}
$$

This is our VAE training objective!

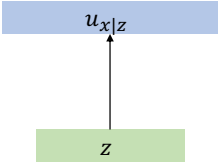
Variational Autoencoders (ELBO)

Jointly train encoder q and decoder p to maximize the variational lower bound on the data likelihood.
Also called Evidence Lower Bound (ELBo)

$$\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

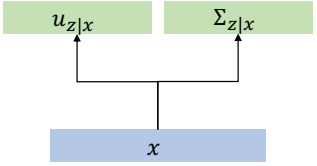
Decoder Network

$p_\theta(x|z) = N(u_{x|z}, \sigma^2)$



Encoder Network

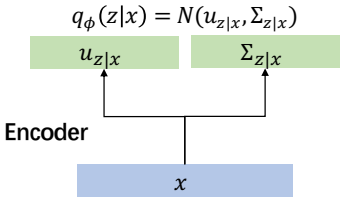
$q_\phi(z|x) = N(u_{z|x}, \Sigma_{z|x})$



Variational Autoencoders: Training

Train by maximizing the **variational lower bound**

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

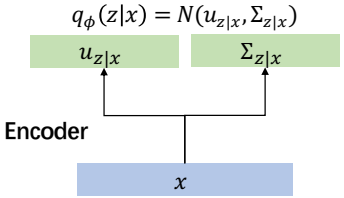


Variational Autoencoders: Training

Train by maximizing the **variational lower bound**

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Run input data through **encoder** to get distribution over z.

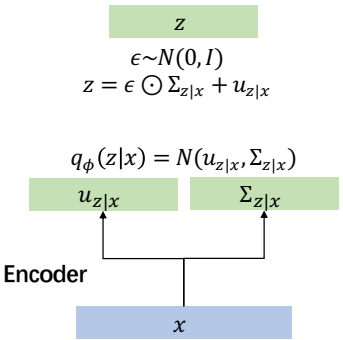


Variational Autoencoders: Training

Train by maximizing the **variational lower bound**

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Run input data through **encoder** to get distribution over z.
2. Prior loss: Encoder output should be unit Gaussian (zero mean, unit variance)
3. Sample z from encoder output $q_\phi(z|x)$ (Reparameterization trick)

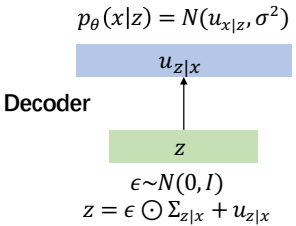


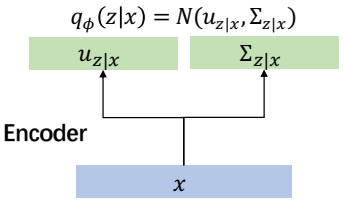
Variational Autoencoders: Training

Train by maximizing the **variational lower bound**

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Run input data through **encoder** to get distribution over z.
2. Prior loss: Encoder output should be unit Gaussian (zero mean, unit variance)
3. Sample z from encoder output $q_\phi(z|x)$ (Reparameterization trick)
4. Run z through **decoder** to get predicted data mean.



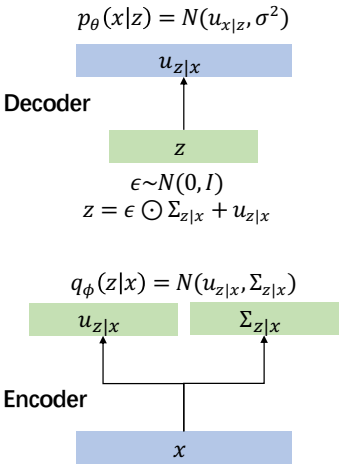


Variational Autoencoders: Training

Train by maximizing the **variational lower bound**

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Run input data through **encoder** to get distribution over z.
2. Prior loss: Encoder output should be unit Gaussian (zero mean, unit variance)
3. Sample z from encoder output $q_\phi(z|x)$ (Reparameterization trick)
4. Run z through **decoder** to get predicted data mean.
5. Reconstruction loss: predicted mean should match x in L_2.



Variational Autoencoders: Training

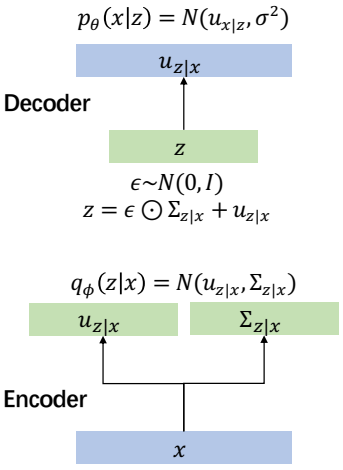
Train by maximizing the **variational lower bound**

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

The loss terms fight against each other!

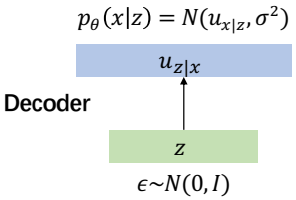
Reconstruction loss wants $\Sigma_{z|x} = 0$ and $u_{z|x}$ to be unique for each x, so decoder can deterministically reconstruct x.

Prior loss wants $\Sigma_{z|x} = I$ and $u_{z|x} = 0$ so encoder output is always Gaussian.



Variational Autoencoders: Sampling

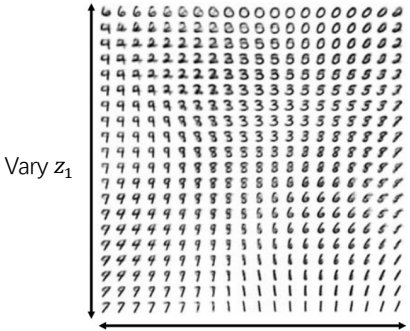
1. Sample z from the prior
2. Run through decoder to get an image



Variational Autoencoders: Sampling

The diagonal prior on $p(z)$ causes dimensions of z to be independent.

"Disentangling factors of variation"



Recap: Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: (x, y) Learn a function to map x -> y

Examples: Classification, regression, Object detection, semantic Segmentation, image captioning, etc.

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn hidden structure in data

Examples: Clustering, dimensionality Reduction, density estimation, etc.

Recap: Generative vs Discriminative Models

Discriminative Model:
Learn a probability distribution $p(y|x)$

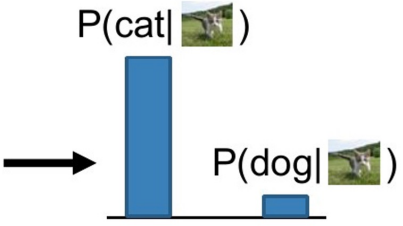
Generative Model:
Learn a probability distribution $p(x)$

Conditional Generative:
Learn a probability distribution $p(x|y)$

Data: x



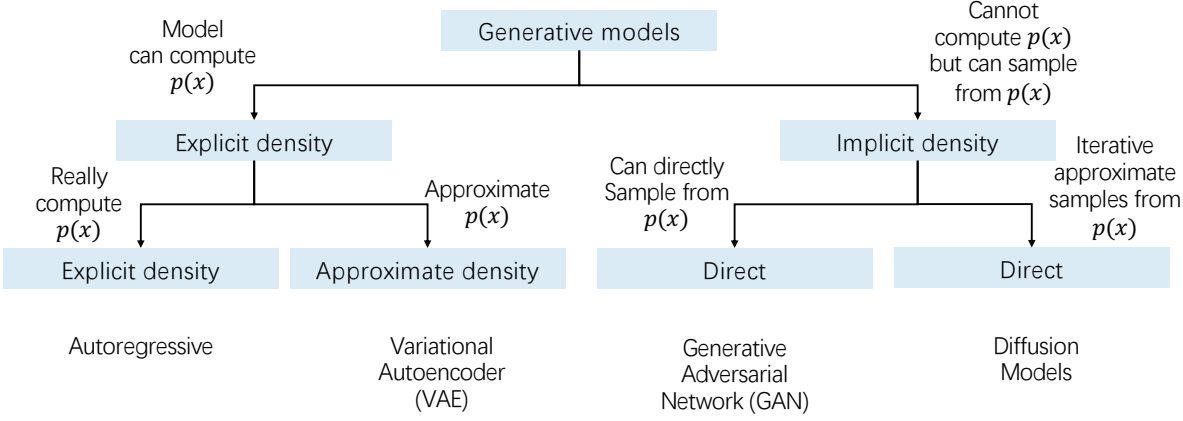
Label: y
Cat



$$\int_X p(x)dx = 1$$

Different values of x **compete** for density.

Recap: Taxonomy of Generative Models?



Thanks

Welcome to join US

Sherrylone.github.io