CS 4803 / 7643: Deep Learning

Topics:

- Variational Auto-Encoders (VAEs)
  - AEs, Variational Inference

Dhruv Batra
Georgia Tech

Administrativia

- Project submission instructions
  - Due: 11/24, 11:59pm
  - Last deliverable in the class
  - Can’t use late days
  - https://www.cc.gatech.edu/classes/AY2021/cs7643_fall/

- Aware of the discussions on Piazza

Recap from ~~last time~~ 2 lectures ago

Types of Learning

$D = \{(\vec{x}_i, \vec{y}_i)\}$

$x \xrightarrow{D} y$

- Supervised learning
  - Learning from a "teacher"
  - Training data includes desired outputs

$s \to a$ $Q(s, a)$

- Reinforcement learning
  - Learning to act under delayed evaluative feedback (rewards)

$D = \{\vec{x}_i\}$

- Unsupervised learning
  - Discover structure in data
  - Training data does not include desired outputs

Supervised vs Reinforcement vs Unsupervised Learning

Unsupervised Learning

Training data is cheap

Data: x
Just data, no labels!

Holy grail: Solve unsupervised learning
=> understand structure of visual world

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised Learning

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

Goal: Learn a *function* to map x → y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

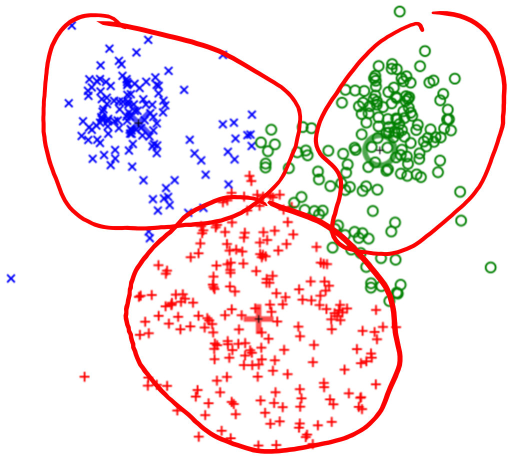
Supervised vs Reinforcement vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



K-means clustering

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

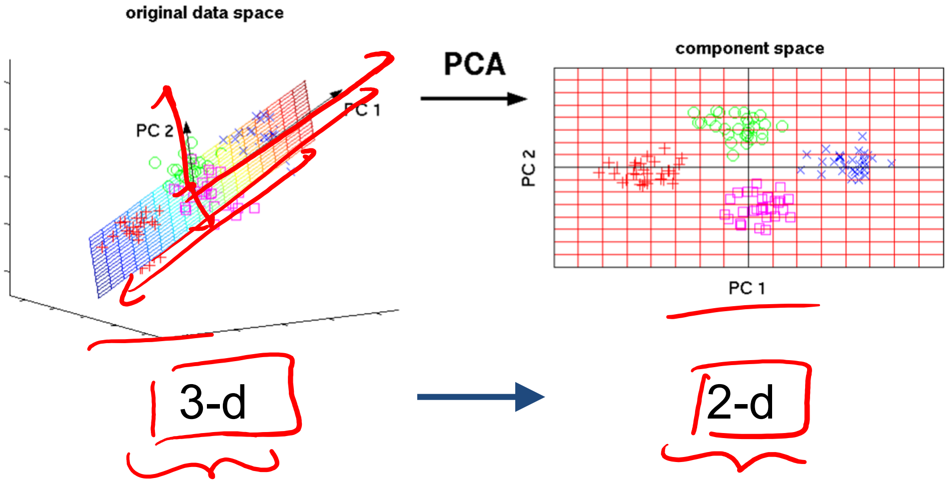
Supervised vs Reinforcement vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



Principal Component Analysis
(Dimensionality reduction)

Supervised vs Reinforcement vs Unsupervised Learning

$x \rightarrow \boxed{p(x)}$

$p(x)$

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

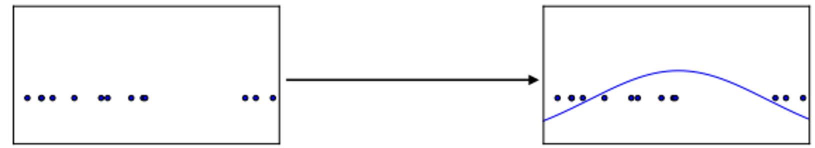
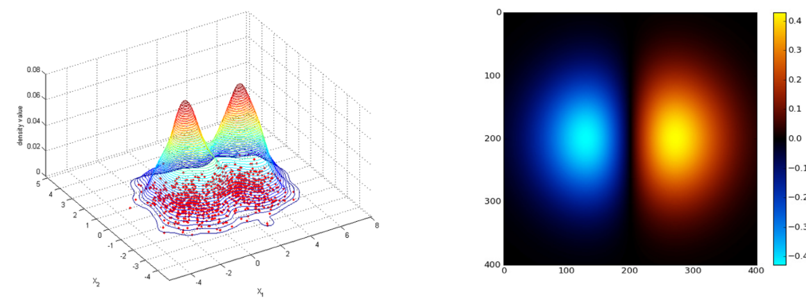


Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation



2-d density estimation

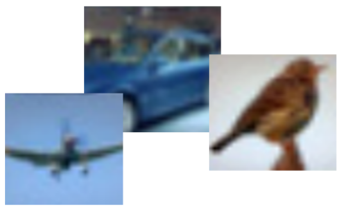
$\vec{x} = [x_1 \; \ldots \; x_d]$

$p(x_i \mid x_j, \ldots, x_{j+n})$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Generative Models

Given training data, generate new samples from same distribution



Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Goal:

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Taxonomy of Generative Models

black box

$x \sim P_{model}(x)$

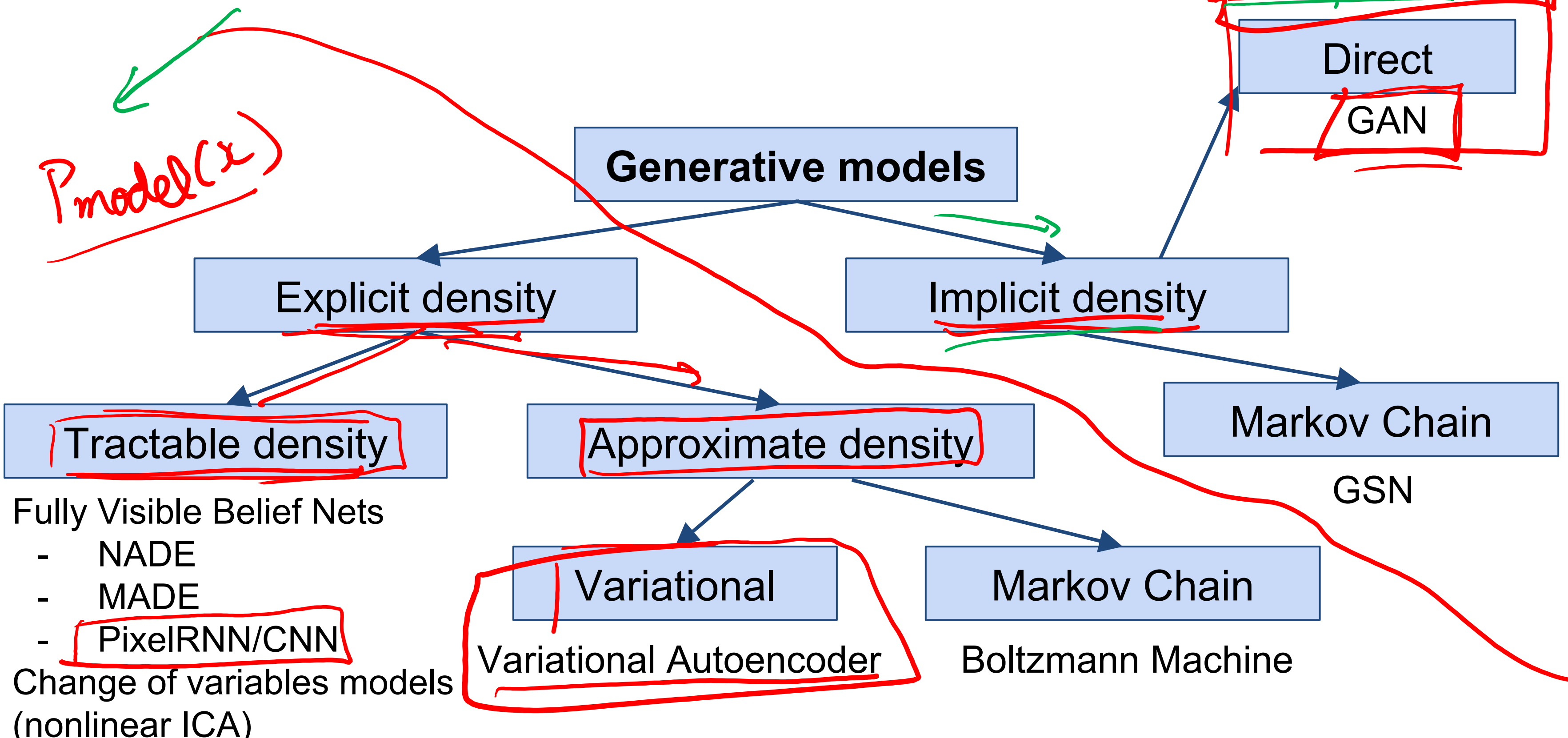


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

We will discuss 3 most popular types of generative models

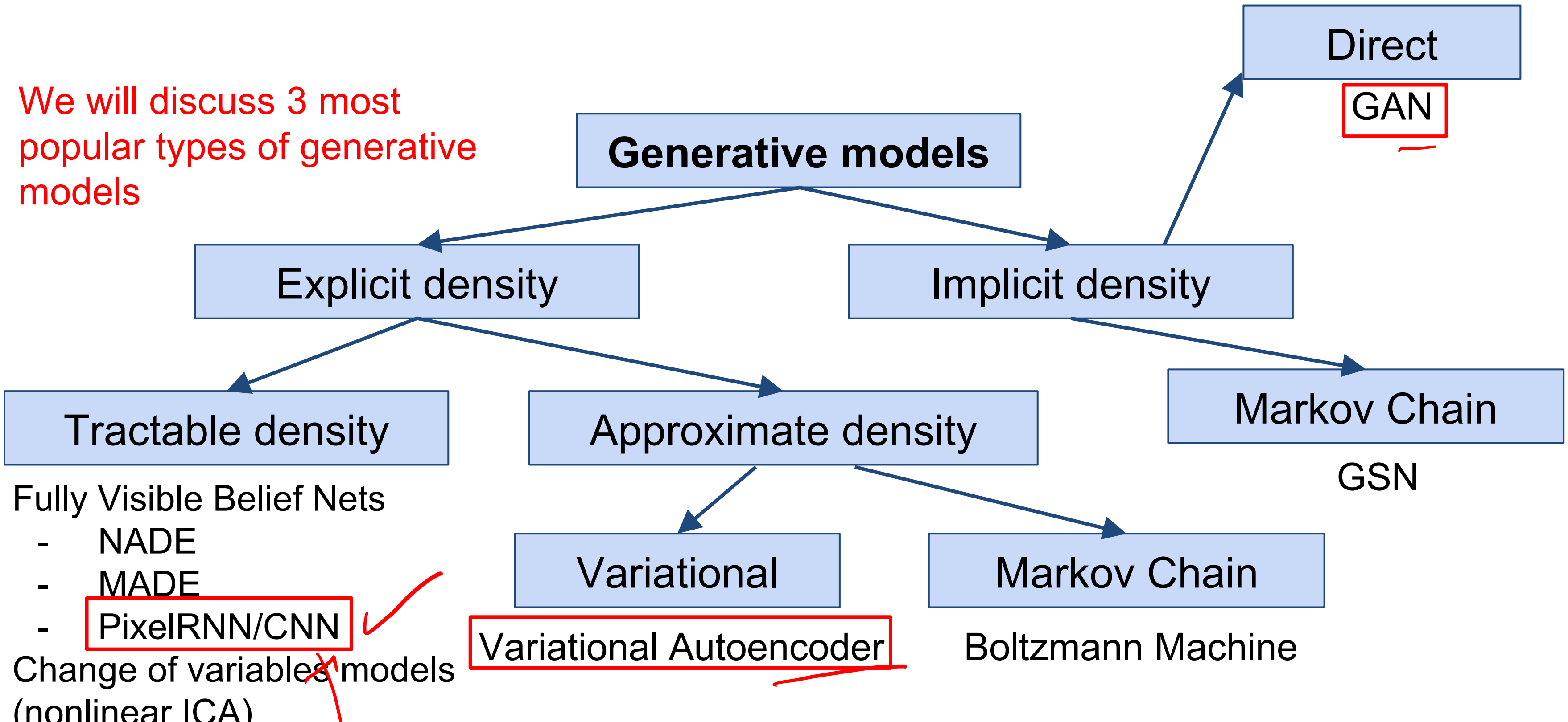


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Plan for Today

- Goal: Variational Autoencoders
- Latent variable probabilistic models
  - Example GMMs
- Autoencodeders
- Variational Inference

Variational Autoencoders (VAE)

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(\vec{x}) = \overset{\text{chain}}{\prod_{i=1}^{n}} p_\theta(x_i | x_1, ..., x_{i-1})$$

$\boxed{D = \{\vec{x}_i\}_{i=1}^{N}}$ N = # samples

"complex" $\rightarrow p(\vec{x})$

R.V

"Latent Variables"

Unobserved RV

$\rightarrow p(\vec{x}, \vec{z})$

$= \underbrace{\boxed{p(\vec{x} | \vec{z})}}_{\text{Conditional}} \underbrace{p(\vec{z})}_{\boxed{\text{Prior}}}$

"simpler"

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

$D = \{ \vec{x}_i \}$

$\boxed{p(\vec{x}, \vec{z})}$

$\boxed{\text{VAEs}}$ define intractable density function with latent **z**:

$$\boxed{p_\theta(\vec{x})} = \boxed{\int} \overbrace{p_\theta(z) p_\theta(x|z)}^{p(x,z)} dz$$

if z is continuous

$\sum_{z} p_\theta(z)\, p(x|z)$ if z is discrete

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead

GMM

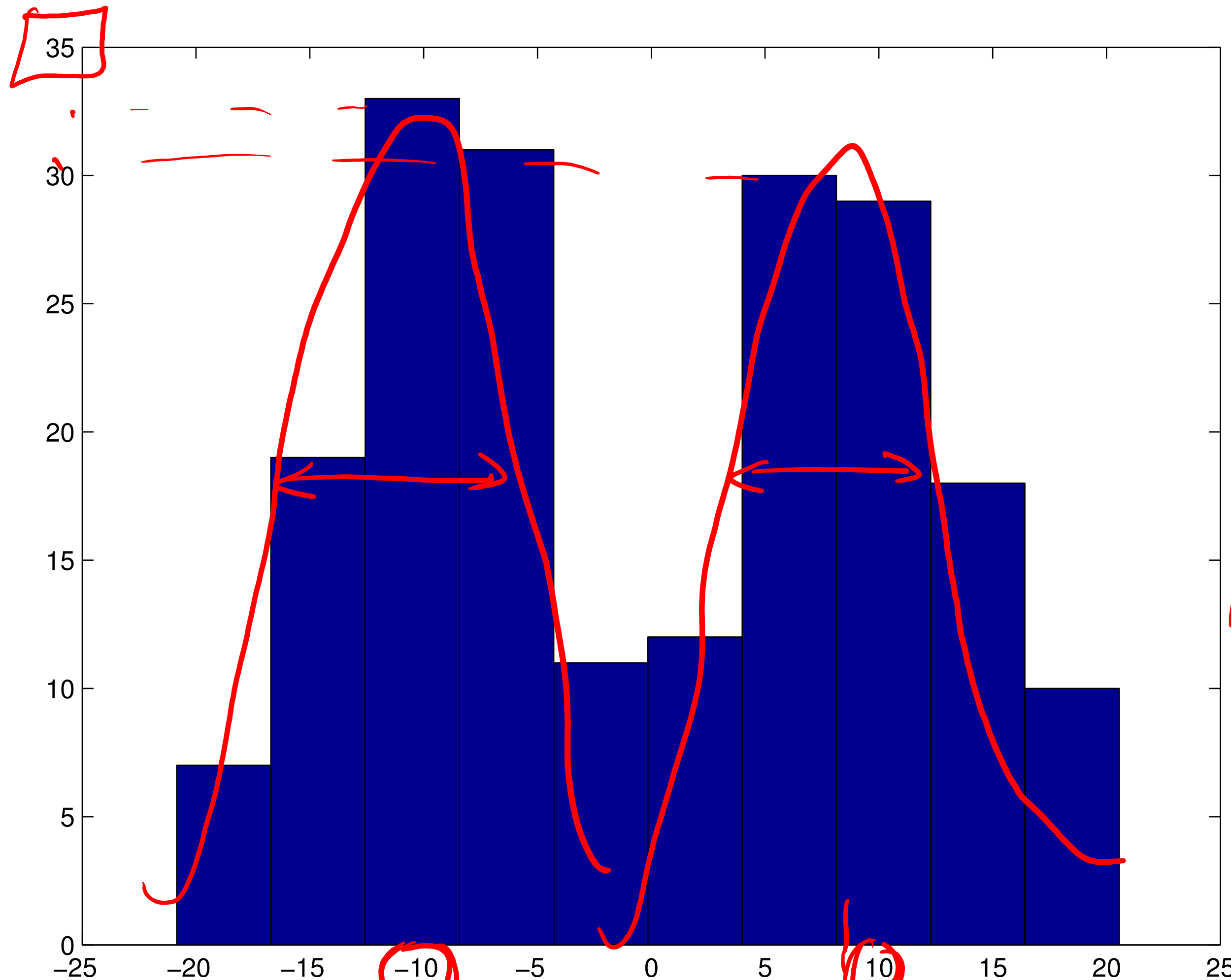
$x \in \mathbb{R}^1$

$z \in \{1, 2\}$

$p(z) = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$

$p(z) = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}$

$\sim p(x)$



$z=1$ $z=2$

$p(x|z) \sim \mathcal{N}(\mu_1, \sigma_1^2)$ $\mu_1 = -10$

$\mathcal{N}(\mu_2, \sigma_2^2)$ $\mu_2 = +10$

Gaussian Mixture Model

$z \in \{1, \ldots, k\}$

$P(x, z)$

z Latent

$\downarrow$

x Observed

$\underline{z} \sim \mathrm{Cat}(\vec{\pi})$ $\begin{bmatrix} \pi_1 \\ \vdots \\ \pi_k \end{bmatrix}$

$\boxed{\boxed{\pi_c} = P_r(z = c)}$

$$\underline{P(\underline{X} \mid z = c)} = \mathcal{N}(\boxed{\mu_c}, \boxed{\sigma_c^2})$$

$$= \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(x-\mu_c)^2}{2\sigma_c^2}}$$

$$P(\underline{x, z}) = P(x \mid z) P(z)$$

Gaussian Mixture Model

$P(z=c) = \pi_c$

$P(x|z) = N(\quad)$

Available from model

$$P(\vec{x}) = \sum_z P(x,z)$$

$$= \sum_z \overline{P(x|z)}\,\overline{P(z)}$$

= Marginalization

$$\boxed{P(z|x)} = \frac{P(z,x)}{P(x)} = \frac{P(x|z)\,P(z)}{\boxed{\sum_z}(\quad)(\quad)}$$

= (Inference)

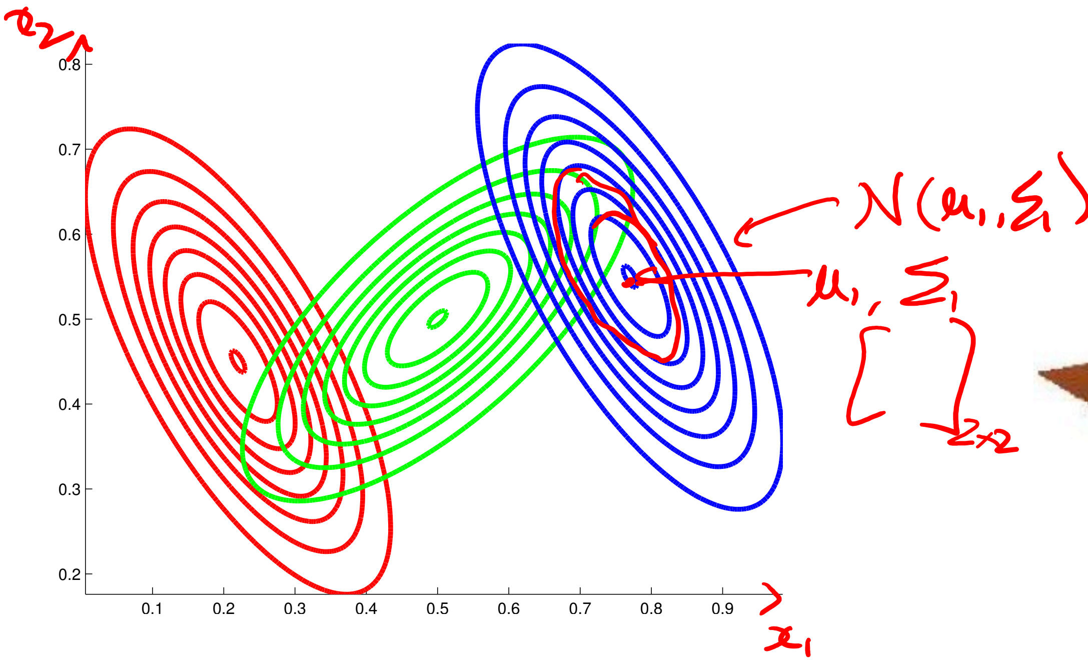
GMM

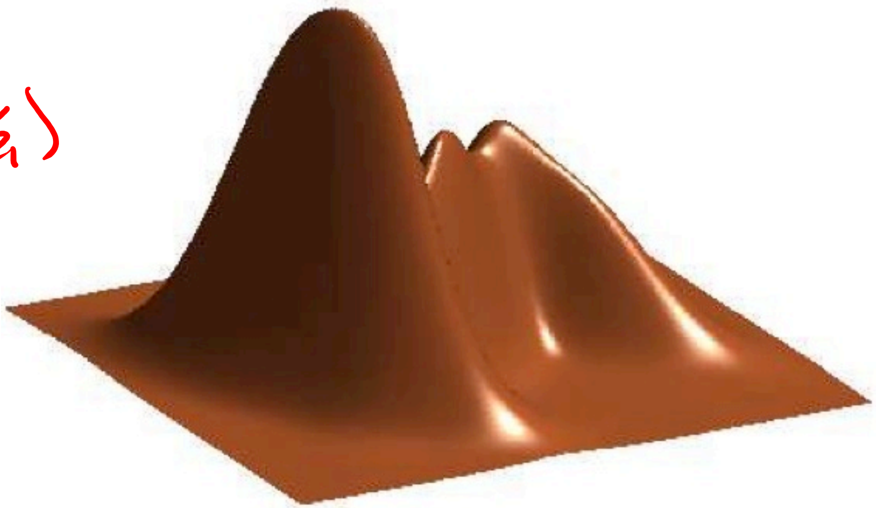
$\vec{x} \in \mathbb{R}^d$

$\vec{\mu} \in \mathbb{R}^d$

$\Sigma \in \mathbb{R}^{d \times d}$

$$N(\vec{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}$$





K-means vs GMM

- K-Means
  - http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html
- GMM
  - https://lukapopijac.github.io/gaussian-mixture-model/

$P(x,z)$

Hidden Data Causes Problems #1

- Fully Observed (Log) Likelihood factorizes
- Marginal (Log) Likelihood doesn't factorize
- All parameters coupled!

Parameters: $\{\pi_1, \ldots, \pi_k, \vec{\mu}_1, \ldots, \vec{\mu}_k, \Sigma_1, \ldots, \Sigma_k\} \equiv \theta$

$$D = \{\vec{x}_i\}_{i=1}^{N} \qquad \vec{x}_i \in \mathbb{R}^d$$

$$\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}} \log P(D \mid \theta)$$

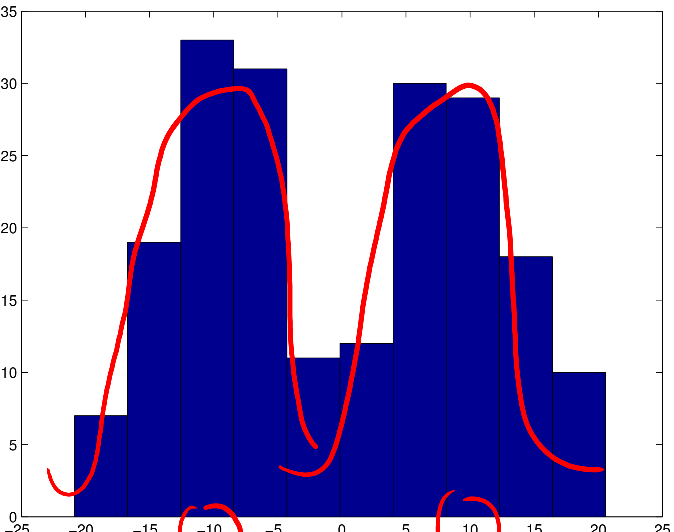
$$\sum_{i=1}^{N} \log \underbrace{P(\underline{\vec{x}_i} \mid \theta)}$$

$$\underbrace{\sum_{i=1}^{N}} \log \boxed{\sum_{\underline{z_i}}} P(x_i, z_i \mid \underline{\theta})$$

$$\int$$

Hidden Data Causes Problems #2

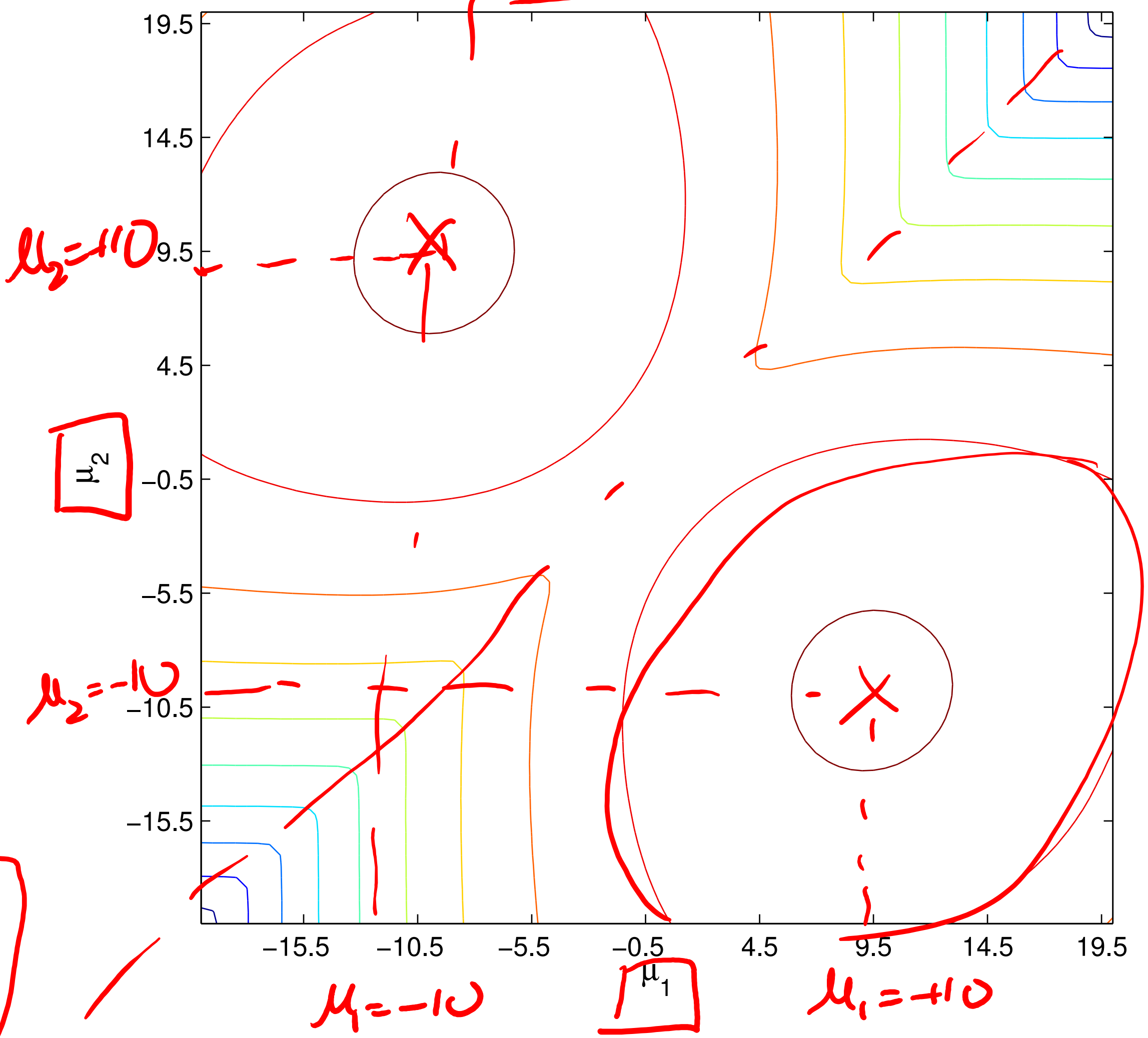
- Identifiability



$\mu_1 = -10$
$\mu_2 = +10$

$\mu_1 = +10$
$\mu_2 = -10$

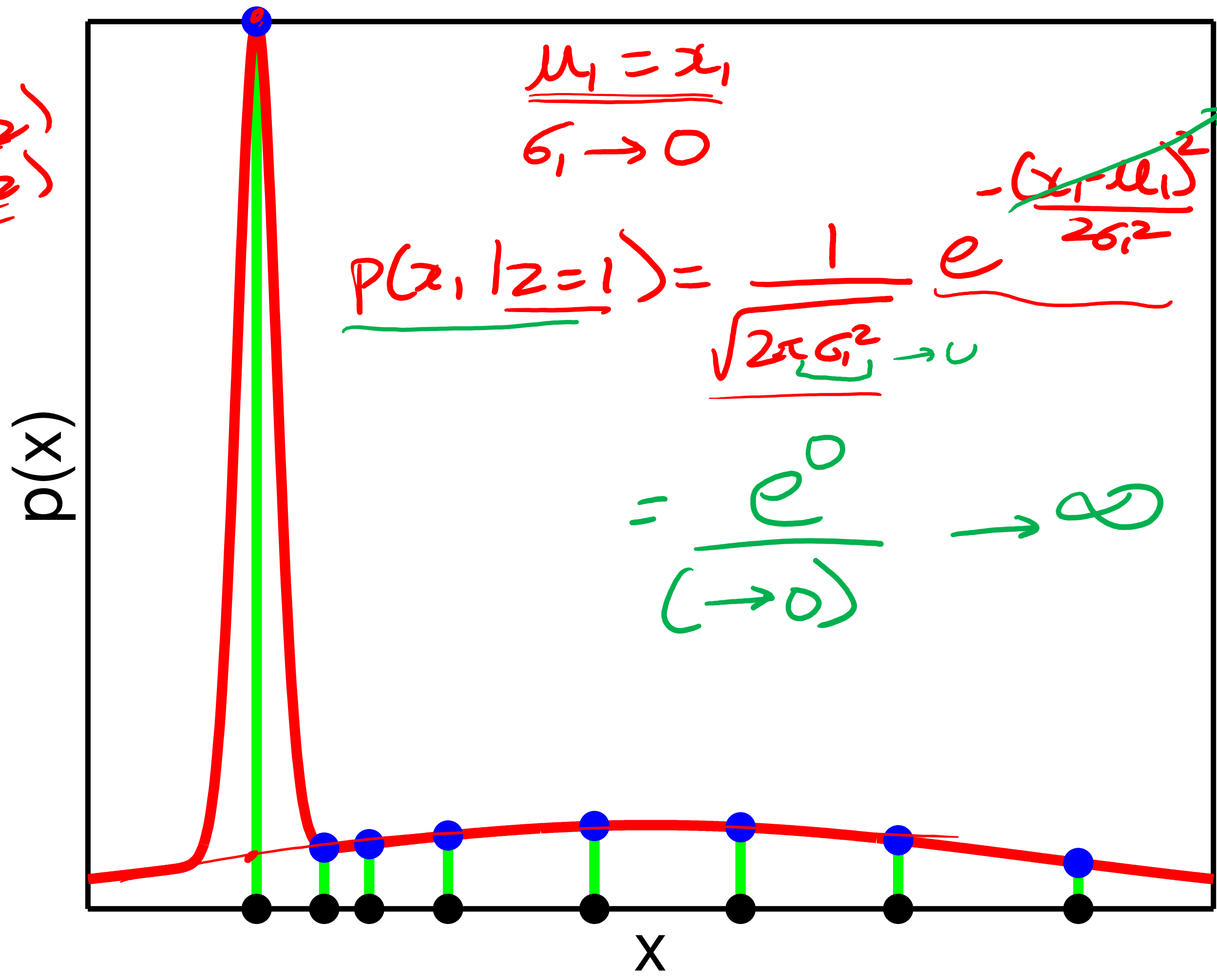
$P(x)$



Hidden Data Causes Problems #3

- Likelihood has singularities if one Gaussian "collapses"

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



$\mu_1 = x_1$

$\sigma_1 \to 0$

$$p(x_1|z=1) = \frac{1}{\underbrace{\sqrt{2\pi\sigma_1^2}}_{\to 0}} e^{\overbrace{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}}^{0}}$$

$$= \frac{e^0}{(\to 0)} \to \infty$$

Variational Auto Encoders

LV

$P(x, z)$

$$\frac{P(z|x)}{P(x|z)}$$

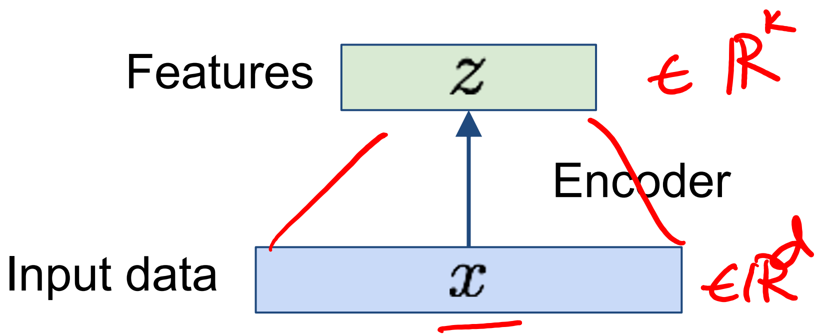
VAEs are a combination of the following ideas:

1. Auto Encoders
2. Variational Approximation
   - Variational Lower Bound / ELBO
3. Amortized Inference Neural Networks
4. "Reparameterization" Trick

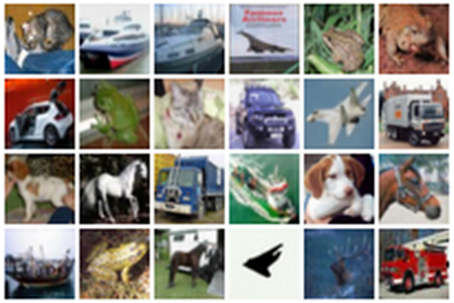
C

Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data



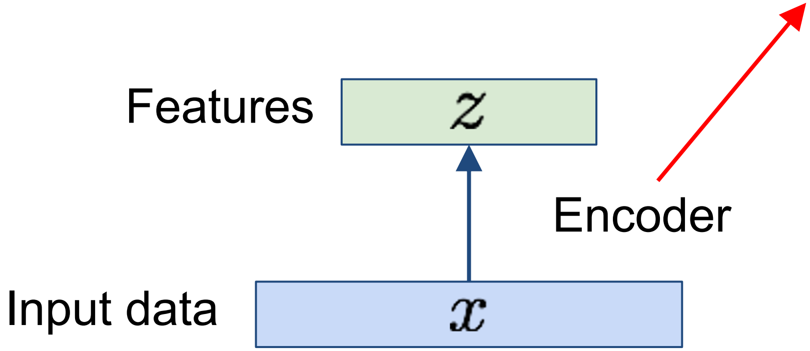
$k < d$

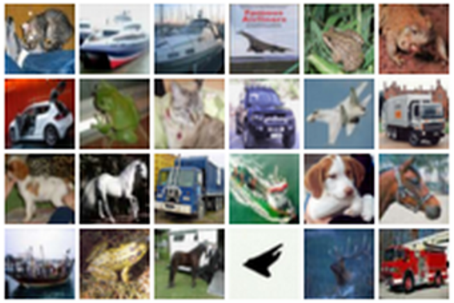


Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN





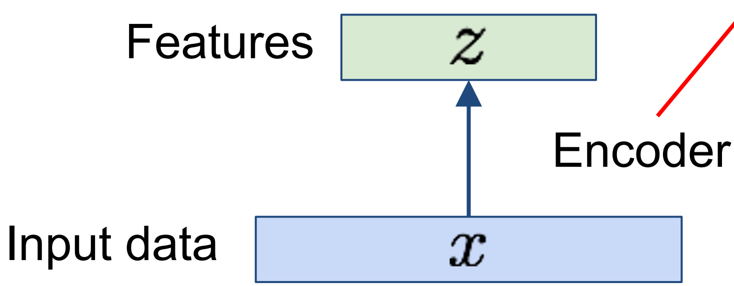
Autoencoders

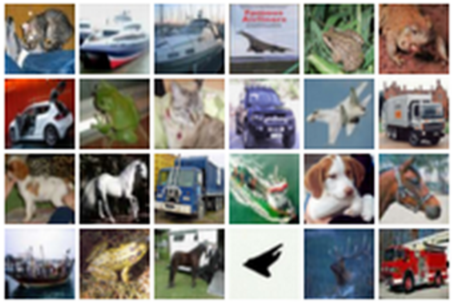
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

Q: Why dimensionality reduction?

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN





Autoencoders

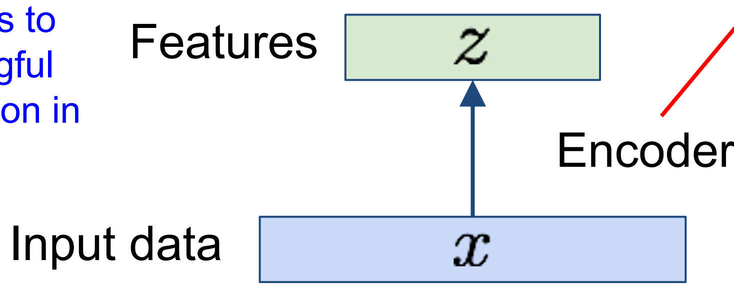
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

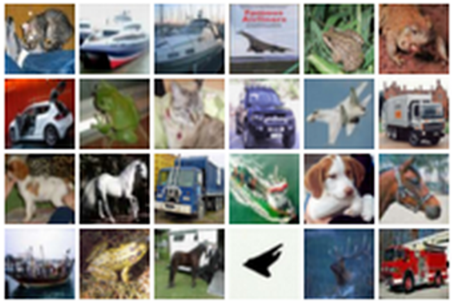
z usually smaller than **x** (dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data

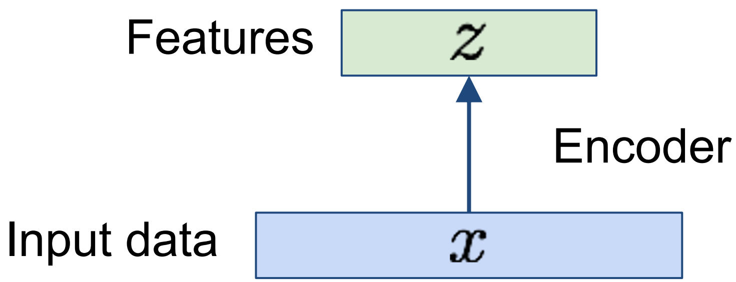
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

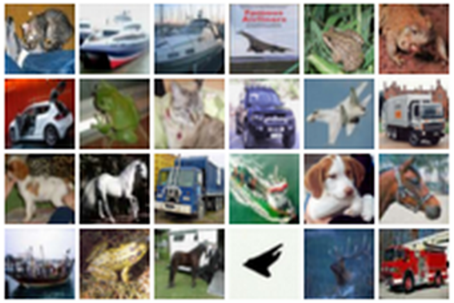




Autoencoders

How to learn this feature representation?



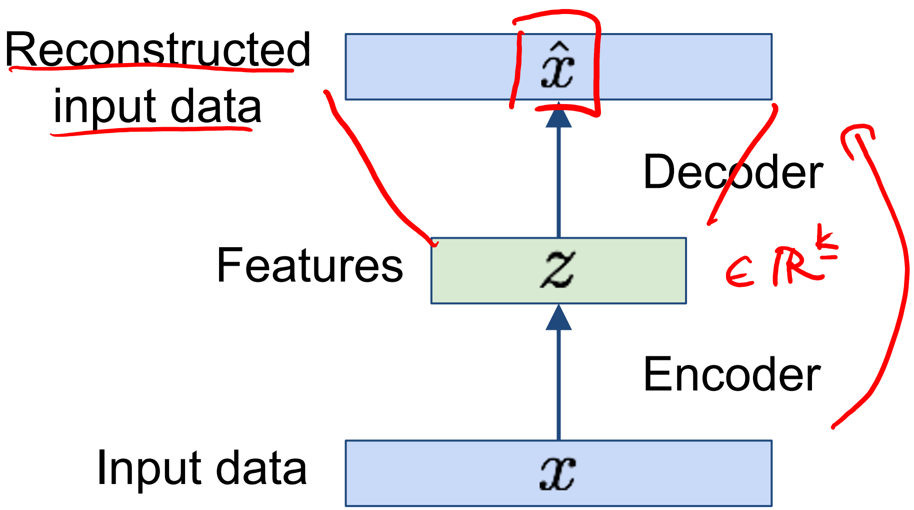


Autoencoders

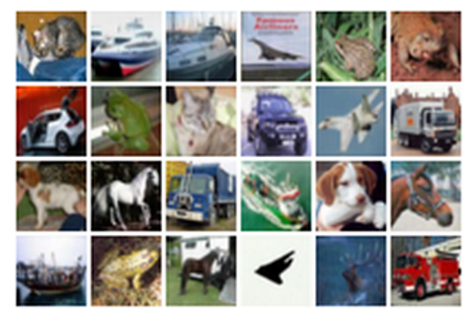
How to learn this feature representation?

Train such that features can be used to reconstruct original data

"Autoencoding" - encoding itself

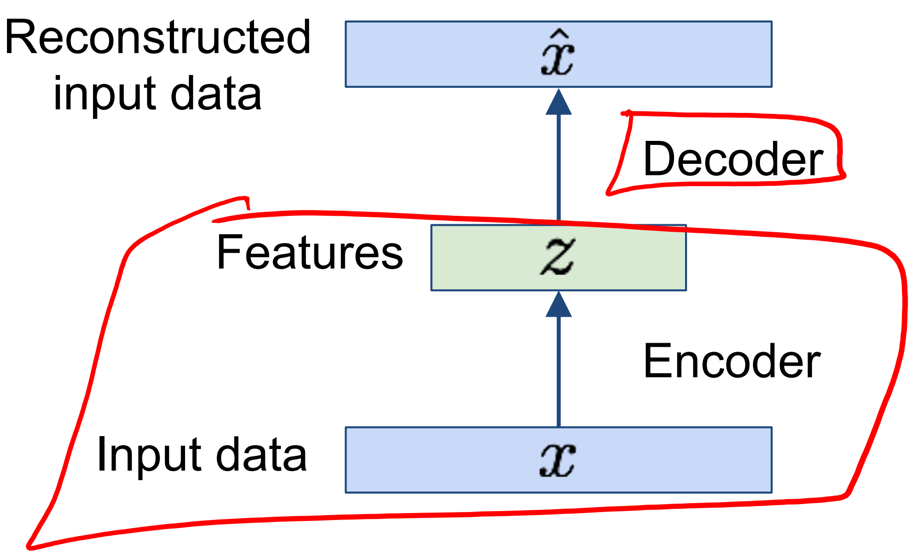


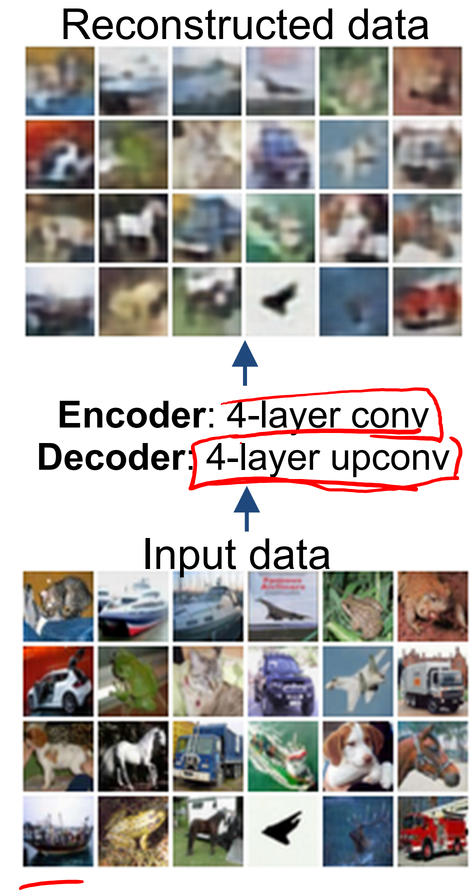
$k < d$



Autoencoders

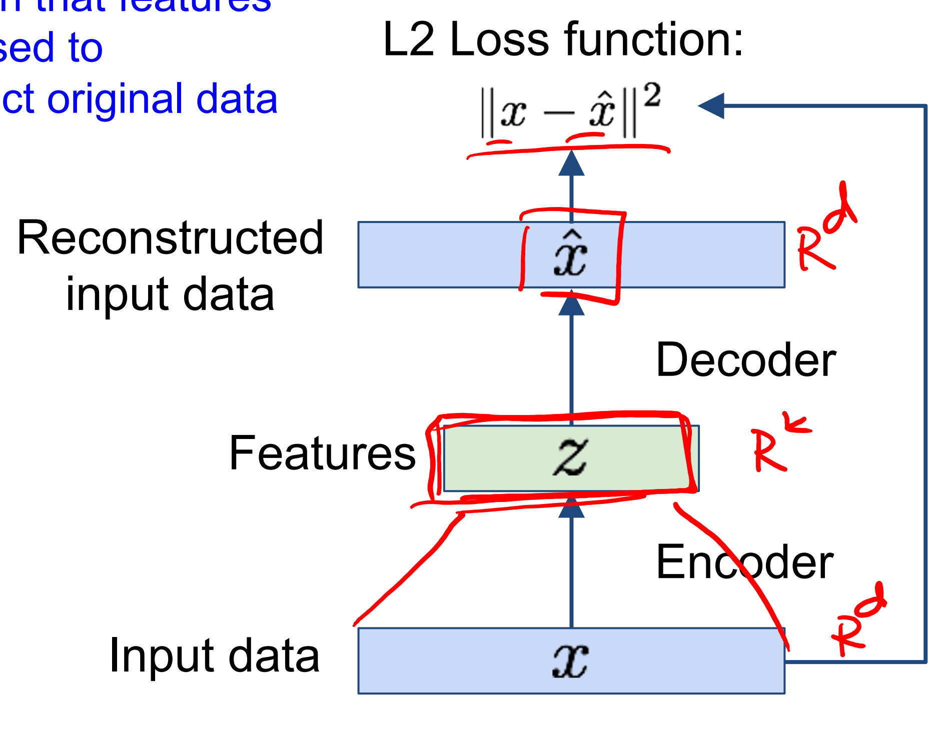
How to learn this feature representation?
Train such that features can be used to reconstruct original data
“Autoencoding” - encoding itself

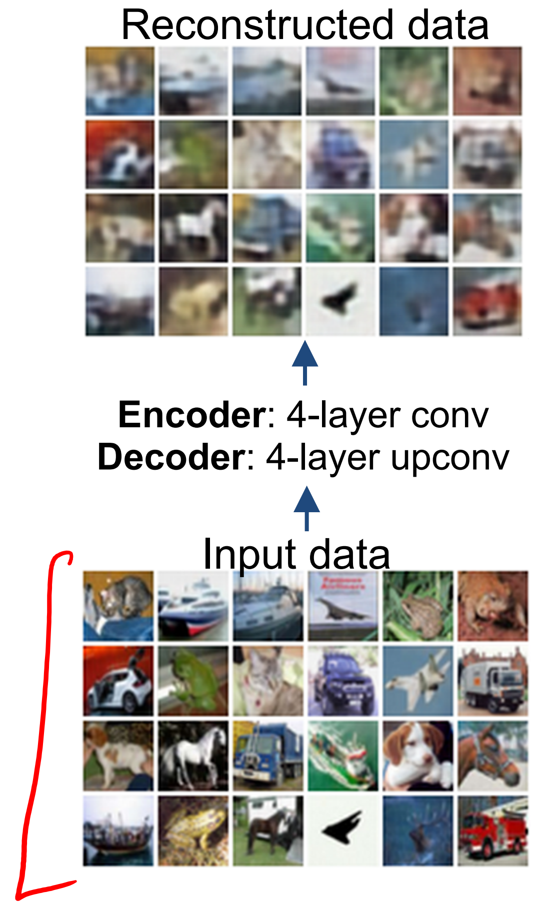




Autoencoders

Train such that features can be used to reconstruct original data



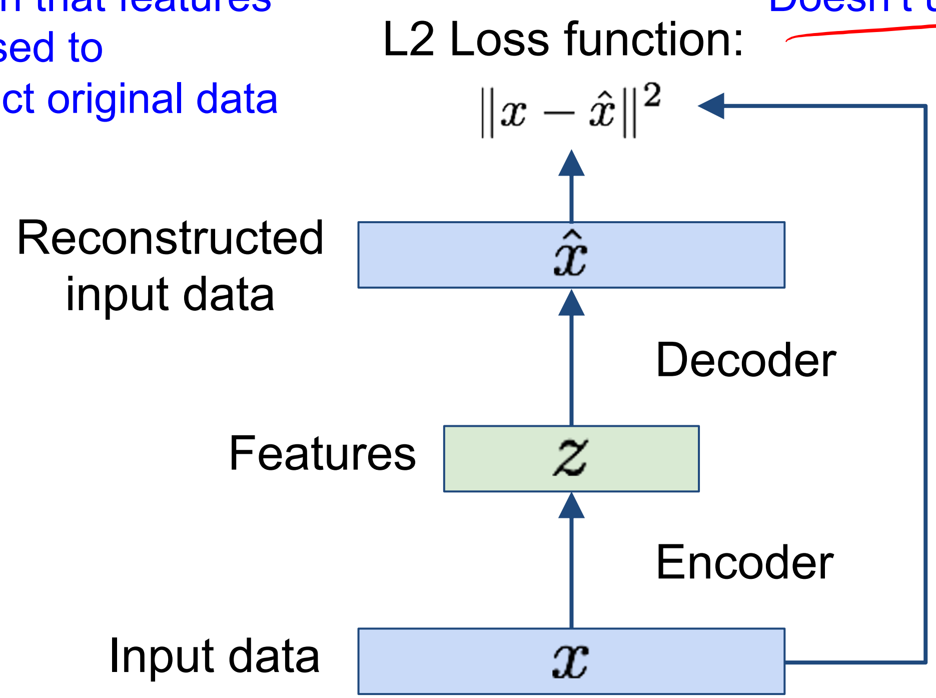


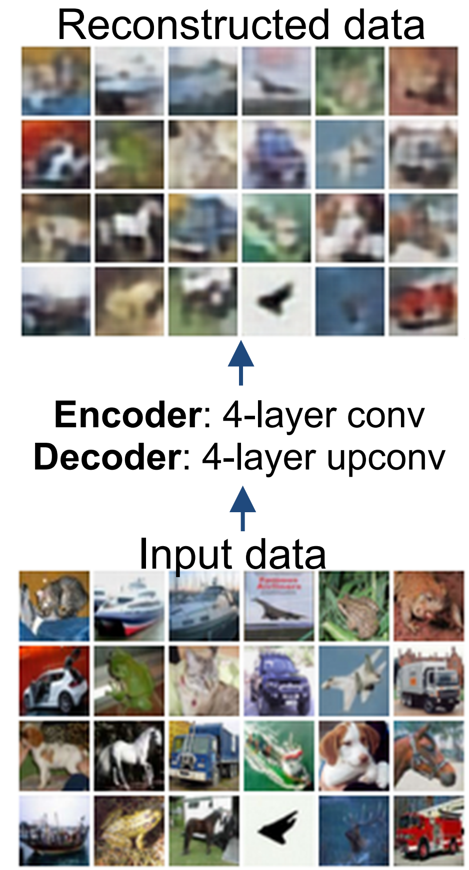
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders

Train such that features can be used to reconstruct original data

Doesn't use labels!

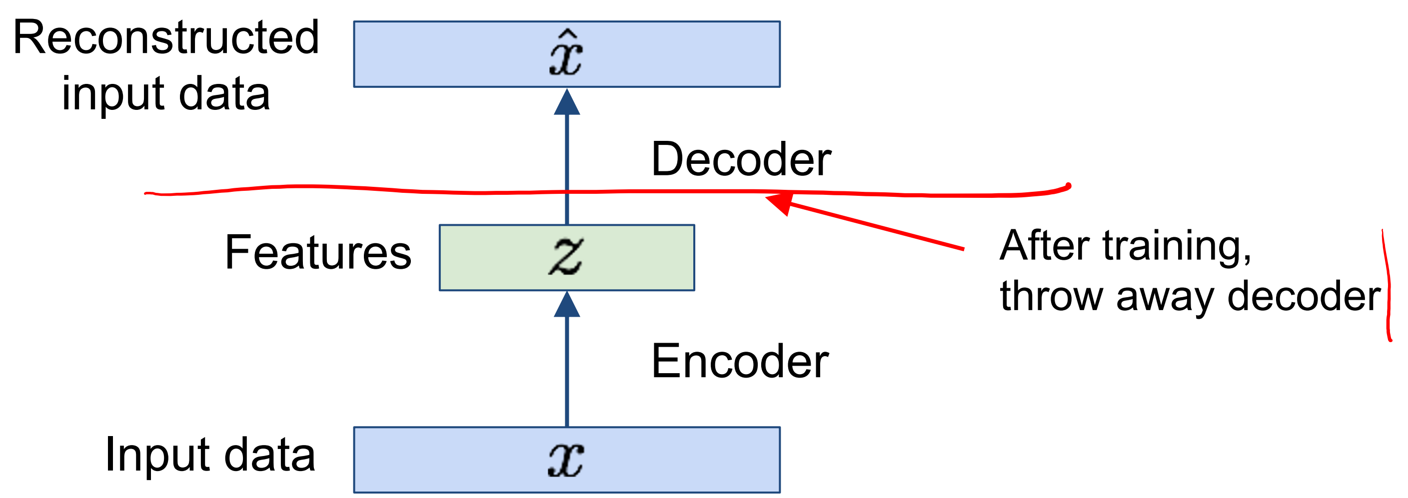




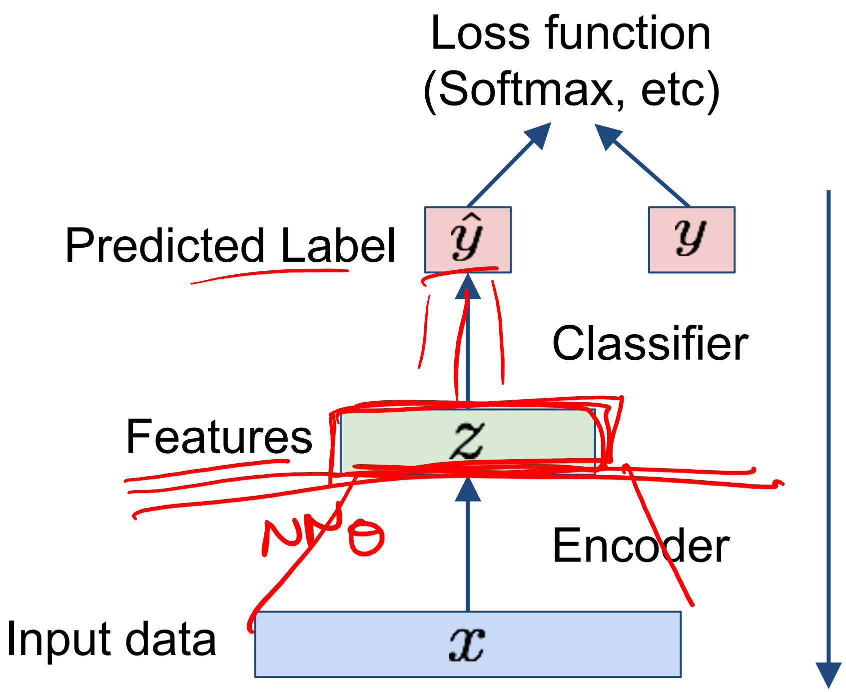
Autoencoders

- Demo
  - https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html

Autoencoders



Autoencoders

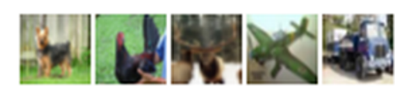


Encoder can be used to initialize a **supervised** model

Fine-tune encoder jointly with classifier

bird plane
dog deer truck

Train for final task (sometimes with small data)



Autoencoders

$z = f_\theta(x)$

$\hat{x} = g_\phi(z)$

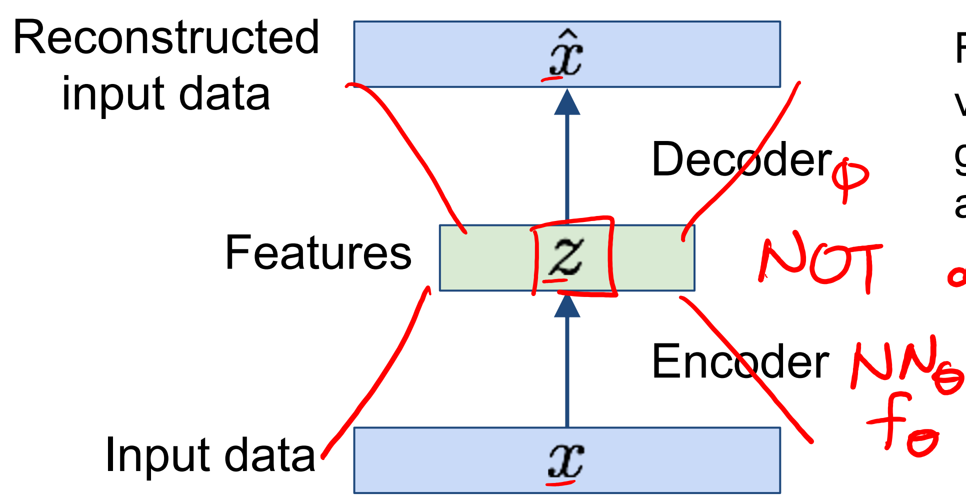
$p(z|x)$

$\sim p(\hat{x}|z)$

VAE

Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Features capture factors of variation in training data. Can we generate new images from an autoencoder?

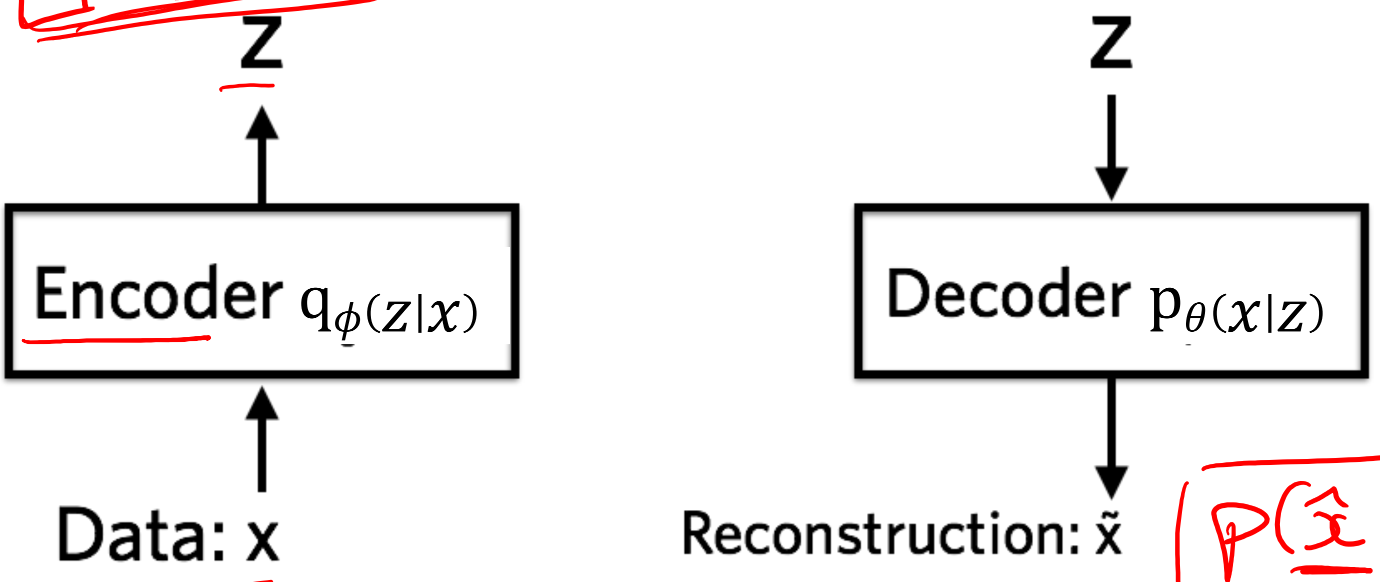


Variational Autoencoders

z

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

$\boxed{p(z|x)}$ Latent



$\boxed{p(\hat{x}|z)}$

Image Credit: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
    - Variational Lower Bound / ELBO

3. Amortized Inference Neural Networks

4. “Reparameterization” Trick

Key problem

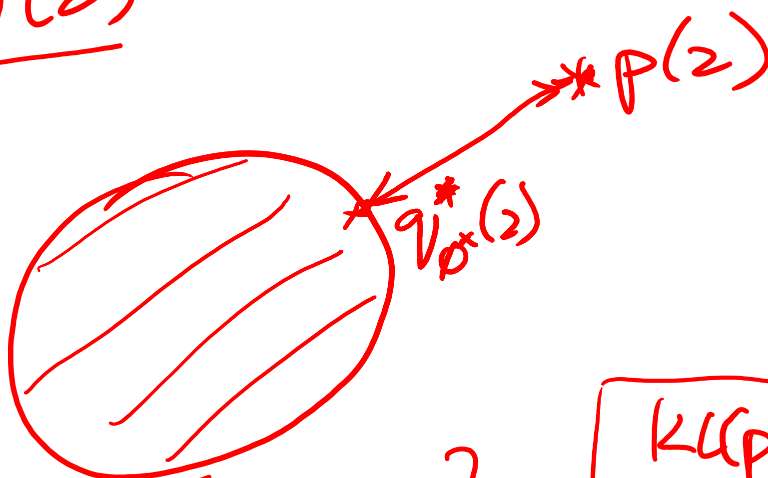
- $\boxed{P(z|x)} = \dfrac{P(z,x)}{P(x)} = \dfrac{P(x|z)\,P(z)}{\boxed{\sum_z} p(x|z)\,p(z)}$

$\int$

"complex" $P(z)$

"simple" $q(z)$

$$\min_{q \in \{q_\phi(z)\}} \underbrace{d(p-q)}_{KL(\quad)}$$



"simple" distributions $\{q_\phi(z)\}$

$KL(p||q)$
"right"
"hard"

$KL(q||p)$
"wrong"
"easy"

What is Variational Inference?

- A class of methods for
  - approximate inference, parameter learning
  - and approximating integrals basically..

- Key idea
  - Reality is complex
  - Instead of performing approximate computation in something complex,
  - Can we perform exact computation in something “simple”?
  - Just need to make sure the simple thing is “close” to the complex thing.

Intuition

KL divergence: Distance between distributions

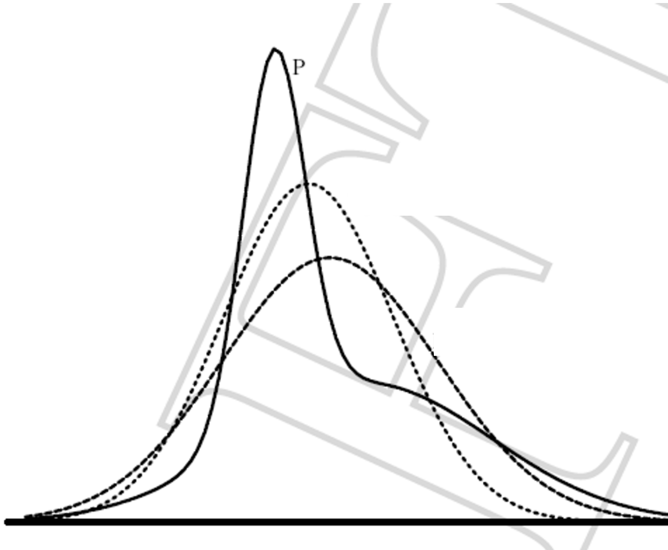
- Given two distributions *p* and *q* KL divergence:

- D(p||q) = 0 iff p=q

- Not symmetric – p determines where difference is important

Find simple approximate distribution

- Suppose *p* is intractable posterior
- Want to find simple *q* that approximates *p*
- KL divergence not symmetric



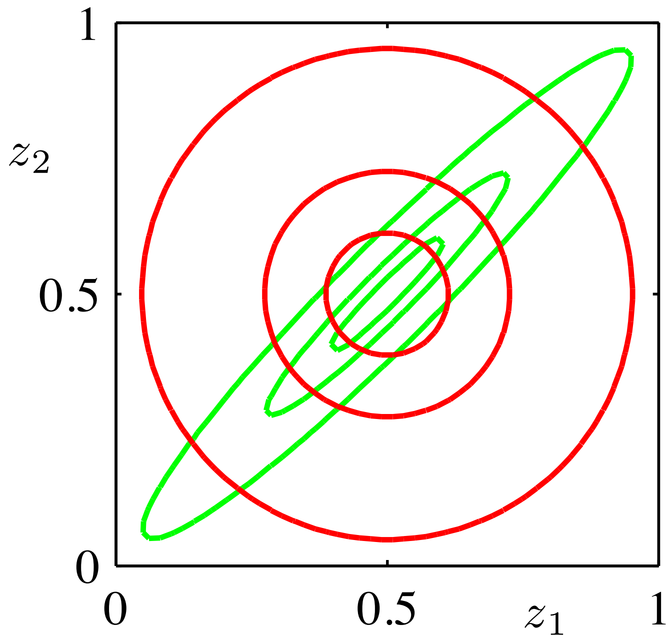
- D(p||q)
  - true distribution p defines support of diff.
  - the “correct” direction
  - will be intractable to compute

- D(q||p)
  - approximate distribution defines support
  - tends to give overconfident results
  - will be tractable

Example 1

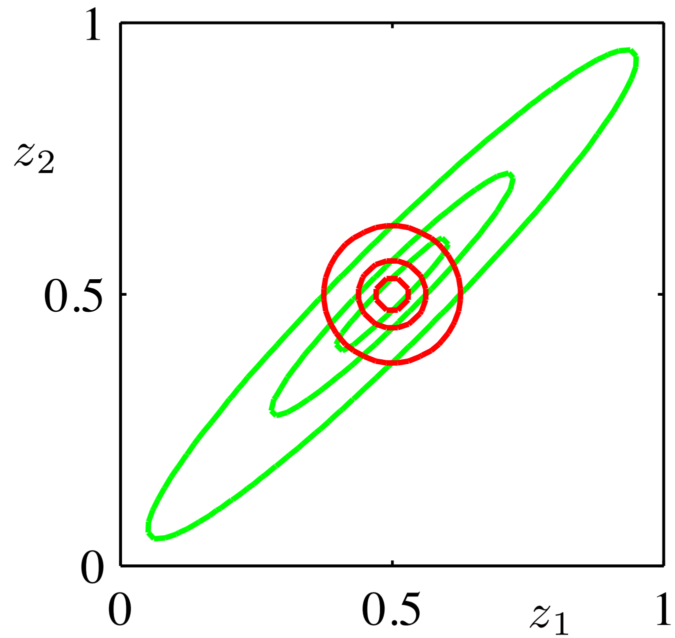
- p = 2D Gaussian with arbitrary co-variance
- q = 2D Gaussian with diagonal co-variance

argmin_q KL (p || q)



(b)

argmin_q KL (q || p)



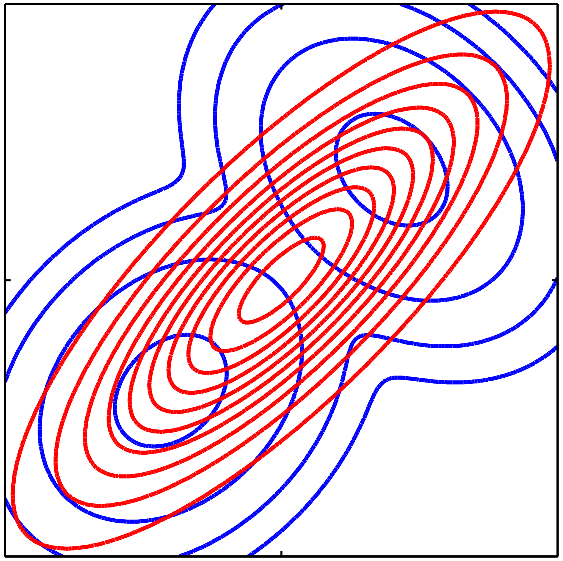
(a)

p = Green; q = Red

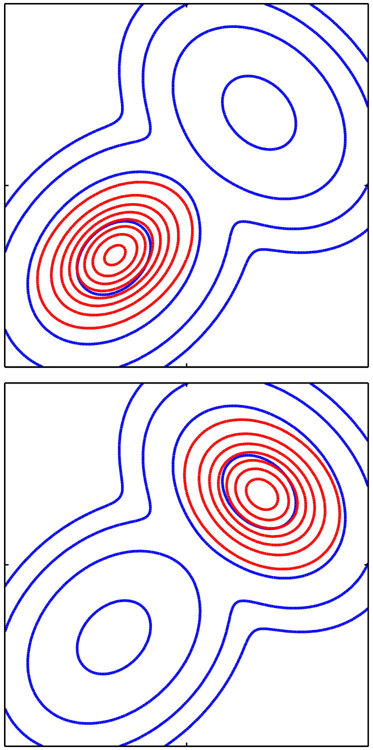
Example 2

- p = Mixture of Two Gaussians
- q = Single Gaussian

argmin_q KL (p || q)



argmin_q KL (q || p)



p = Blue; q = Red