CS 4803 / 7643: Deep Learning

Topics:

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

Dhruv Batra

Georgia Tech

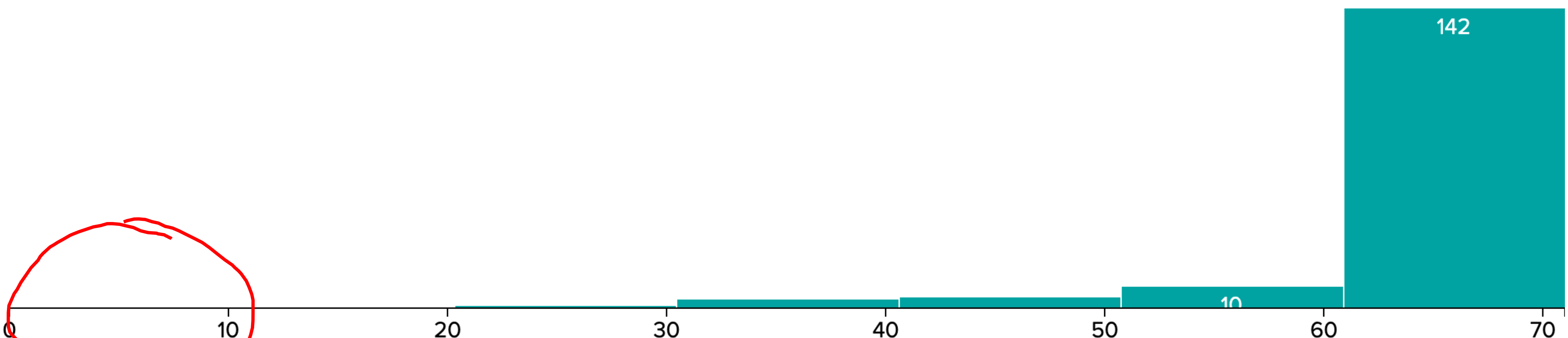
Administrativia

- HW4 Reminder
  - Due: 11/07, 11:55pm
  - Reinforcement Learning
  - Last HW. Focus on project after that.
- Final project
  - No poster session
  - PDF Report submission
    - Details out soon

Administrativia

- HW3 Grades Released
  - Regrade requests close: 11/15, 11:55pm
  - Please check solutions first!

- Grade histogram: 7643
  - Max possible: 71 (regular credit) + 0 (extra credit)

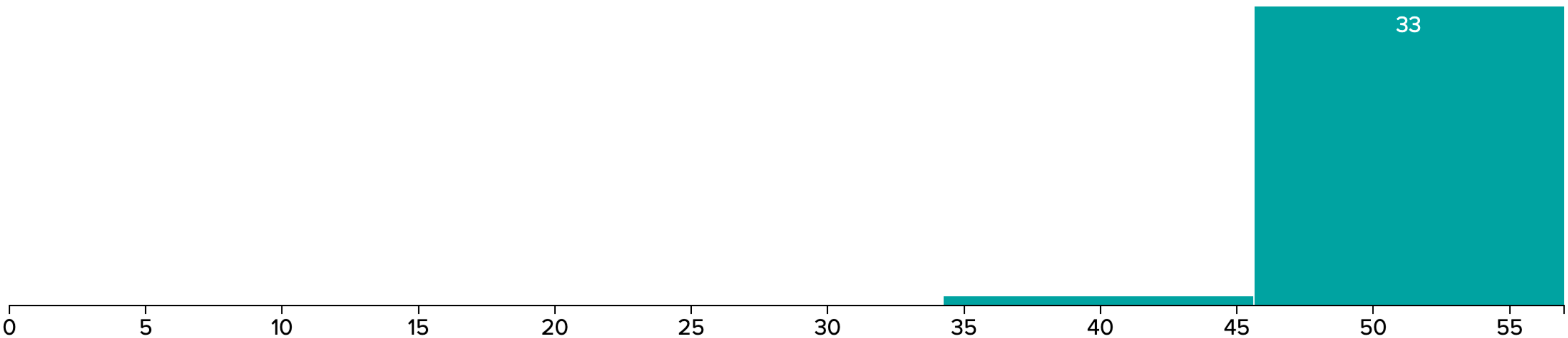


MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
28.0	68.0	71.0	65.51	7.52

Administrativia

- HW3 Grades Released
  - Regrade requests close: 11/15, 11:55pm
  - Please check solutions first!

- Grade histogram: 4803
  - Max possible: 55 (regular) + 14 (extra credit)



MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
40.0	62.0	71.0	61.97	6.77

Recap from ~~last time~~ 2 lectures ago

Supervised vs Reinforcement vs Unsupervised Learning

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

Supervised vs Reinforcement vs Unsupervised Learning

NN

θ

x y

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.

→ Cat

Classification

Supervised vs Reinforcement vs Unsupervised Learning

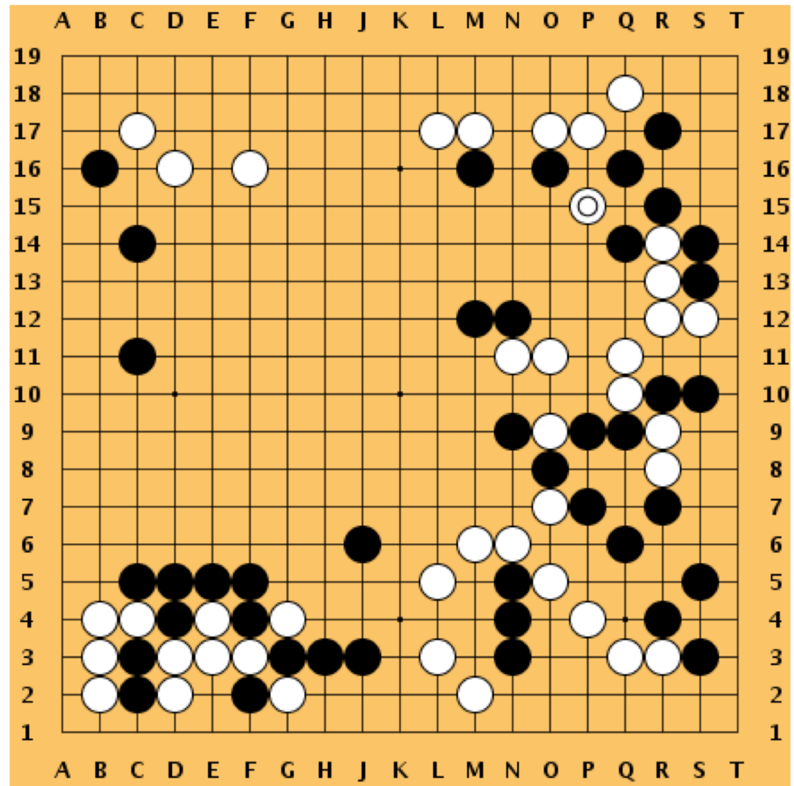
$S \rightarrow a^{st}$

Reinforcement Learning

Given: (e, r)
Environment e, Reward function r (evaluative feedback)

Goal: Maximize expected reward

Examples: Robotic control, video games, board games, 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.

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.

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.

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

Supervised vs Reinforcement vs Unsupervised Learning

$p(\vec{x})$ $p(x_i | x_j)$

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.

Tasks

Supervised Learning

x →	Classification	→ y	Discrete
x →	Regression	→ y	Continuous

Unsupervised Learning

x →	Clustering	→ c	Discrete
x →	Dimensionality Reduction	→ z	Continuous
x →	Density Estimation	→ $\boxed{p(x)}$	On simplex

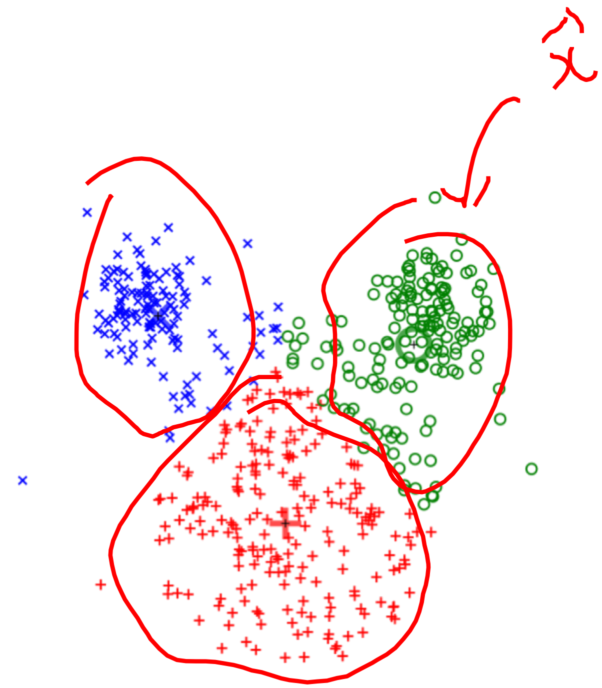
Supervised vs Reinforcement vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

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

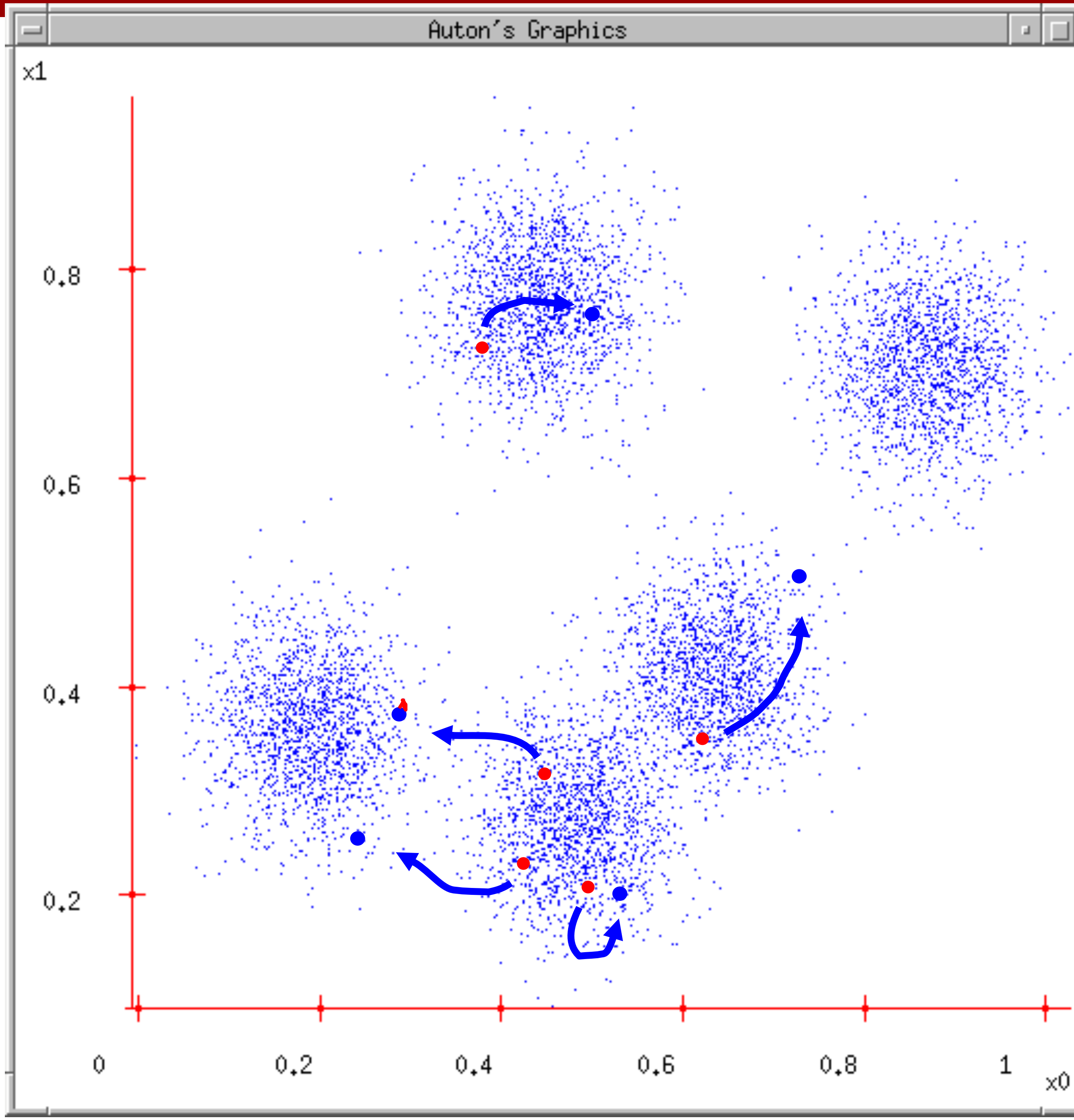
Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



K-means clustering

K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



K-means as Co-ordinate Descent

$a_{ij} = 1$ if $x_i \in$ clust j

- Optimize objective function:

$$\min_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k} \min_{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_N} F(\boldsymbol{\mu}, \boldsymbol{a}) = \min_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k} \min_{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_N} \sum_{i=1}^{N} \sum_{j=1}^{k} a_{ij} \underbrace{||\mathbf{x}_i - \boldsymbol{\mu}_j||^2}_{\text{distortion}}$$

- Fix μ, optimize a
- Fix a, optimize μ

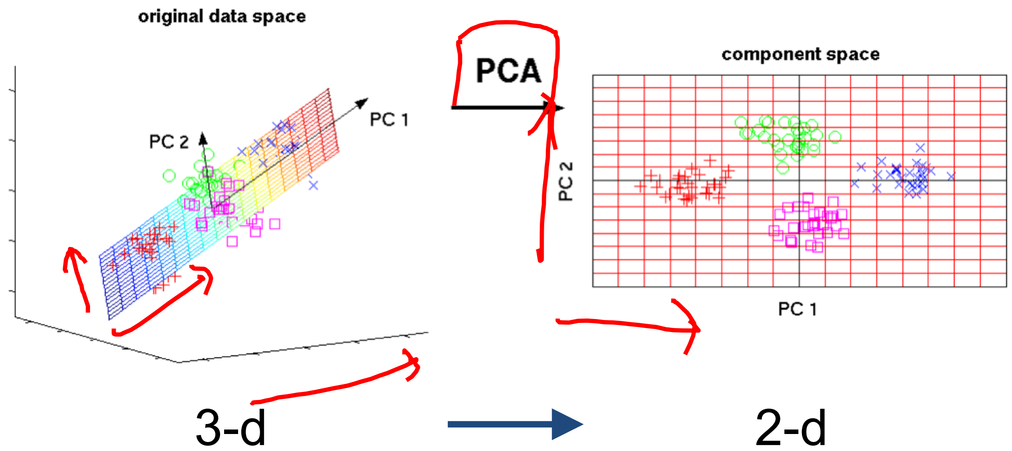
Supervised vs Reinforcement vs Unsupervised Learning

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

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

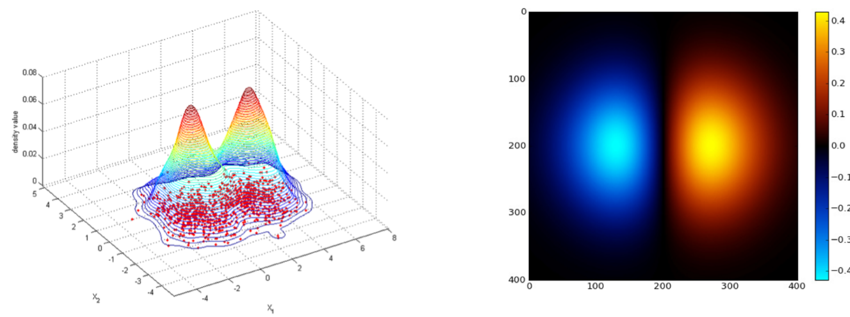
Unsupervised Learning

Data: x
Just data, no labels!

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

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

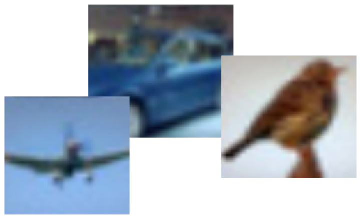
1-d density estimation



2-d density estimation

Generative Models

Given training data, generate new samples from same distribution



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

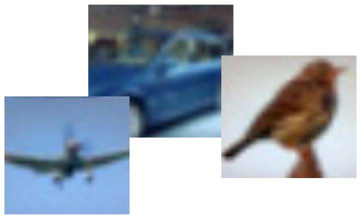
Generated samples ~ $p_{model}(x)$

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

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 ~ $p_{model}(x)$

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

Addresses density estimation, a core problem in unsupervised learning

$p(\vec{x})$

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

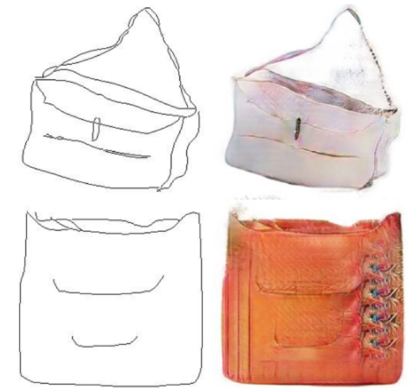
$x \sim p(\vec{x})$

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

Why Generative Models?

p(image | control)

- Realistic samples for artwork, super-resolution, colorization, etc.



- Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!)
- Training generative models can also enable inference of latent representations that can be useful as general features

Taxonomy of Generative Models

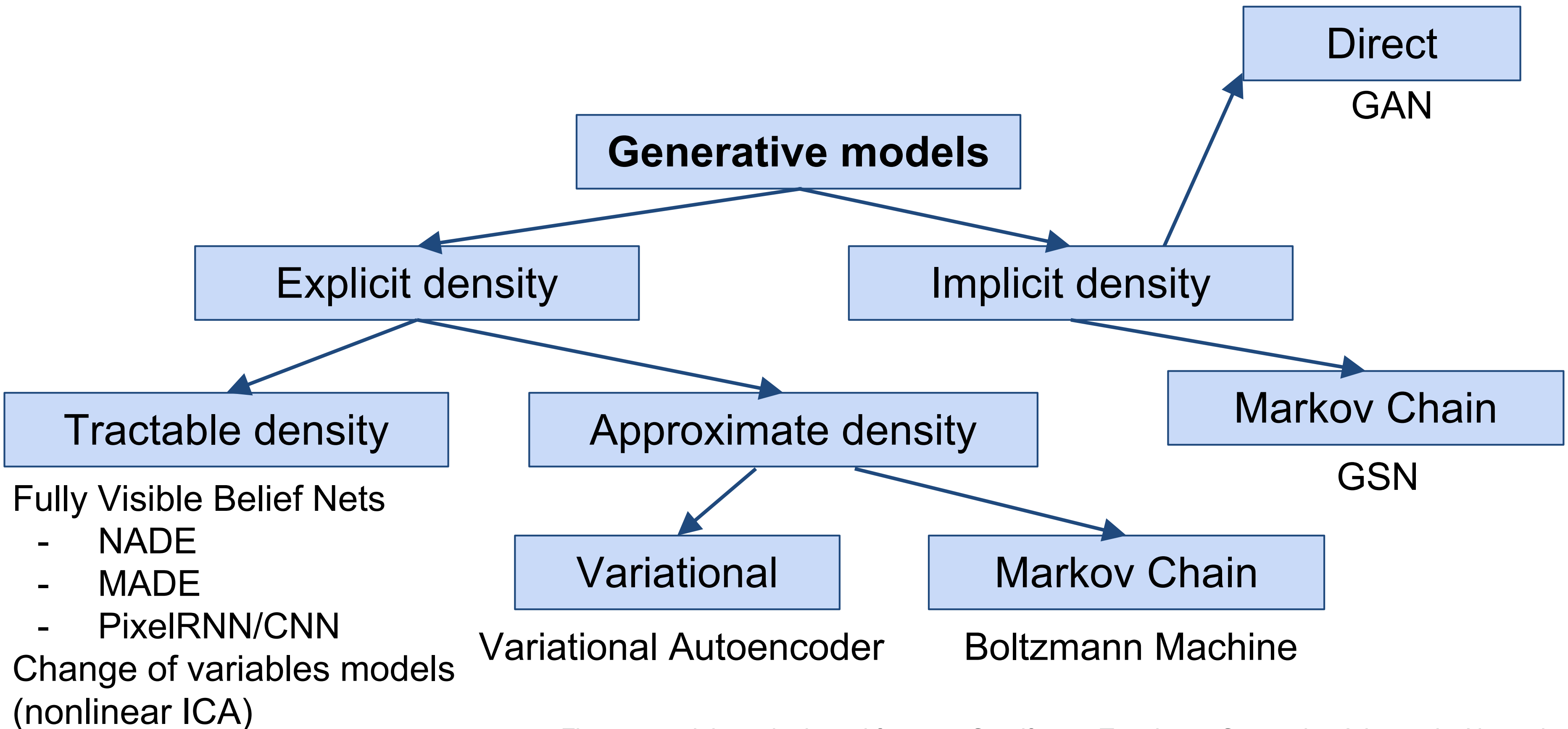


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

We will discuss 3 most popular types of generative models

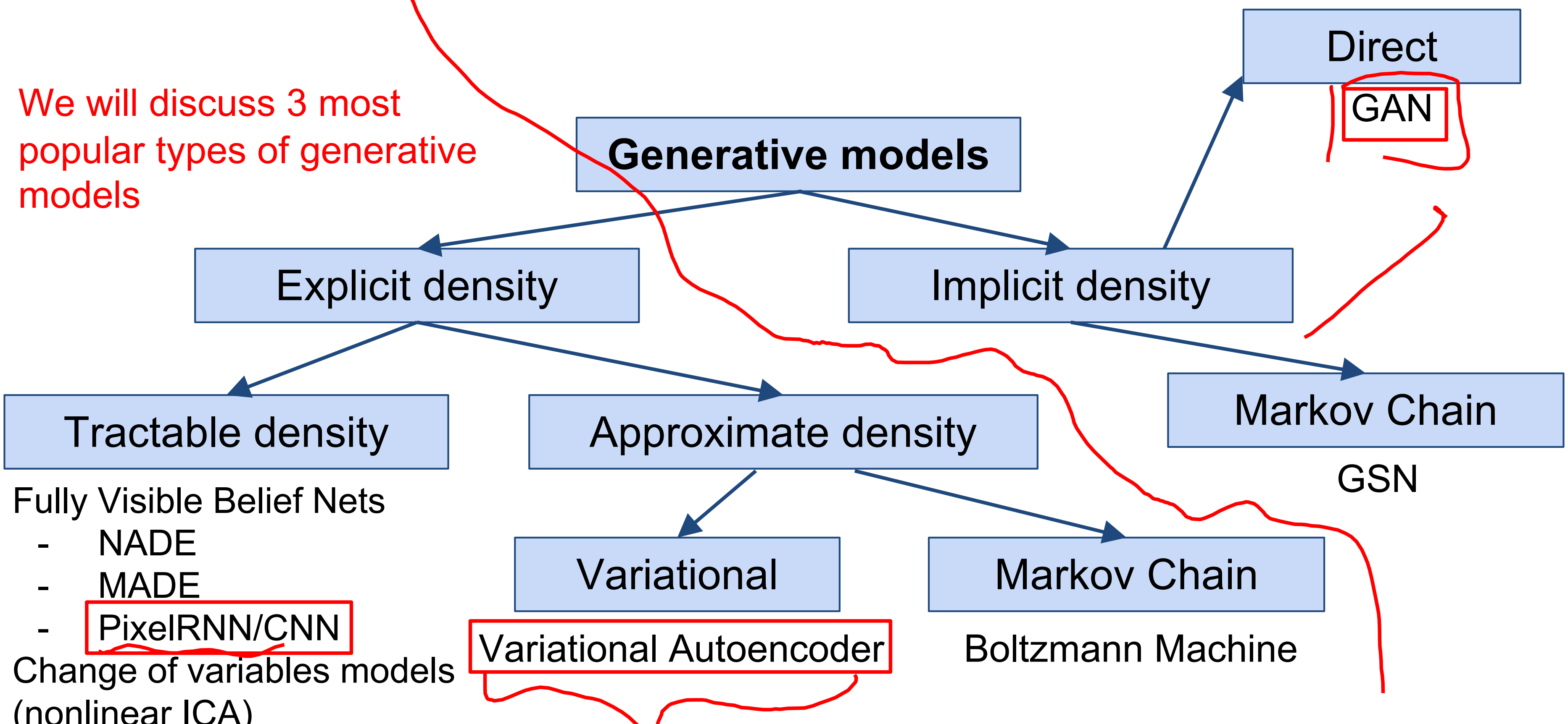


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Fully Observable Model

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

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

$$D = \{\vec{x}^{(i)}\}_{i=1}^{N}$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Then maximize likelihood of training data

$$x_1 \cdots x_{i-1} \to \boxed{\text{NN}}_\theta \to p(x_i)$$

Fully Observable Model

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

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

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Then maximize likelihood of training data

Complex distribution over pixel values
=> Express using a neural network!

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:

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

$P(\vec{x})$ $D = \{\vec{x}\}$

$P(\vec{x}, \boxed{z})$ "latent" "hidden"

$= \underbrace{P(\vec{x} | z)}_{\text{conditional}} \underbrace{P(z)}_{\text{prior}}$

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**:

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

z continuous

$$\sum_z p_\theta(z) p_\theta(x|z)$$

z discrete

So far...

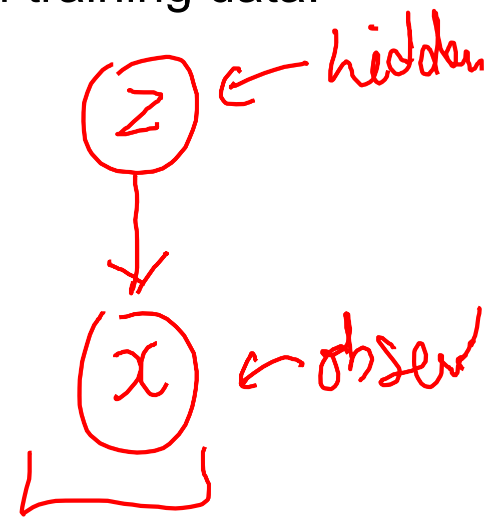
PixelCNNs define tractable density function, optimize likelihood of training data:

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

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

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

$$\sum$$



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

GMM

Gaussian Mixture Model

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

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

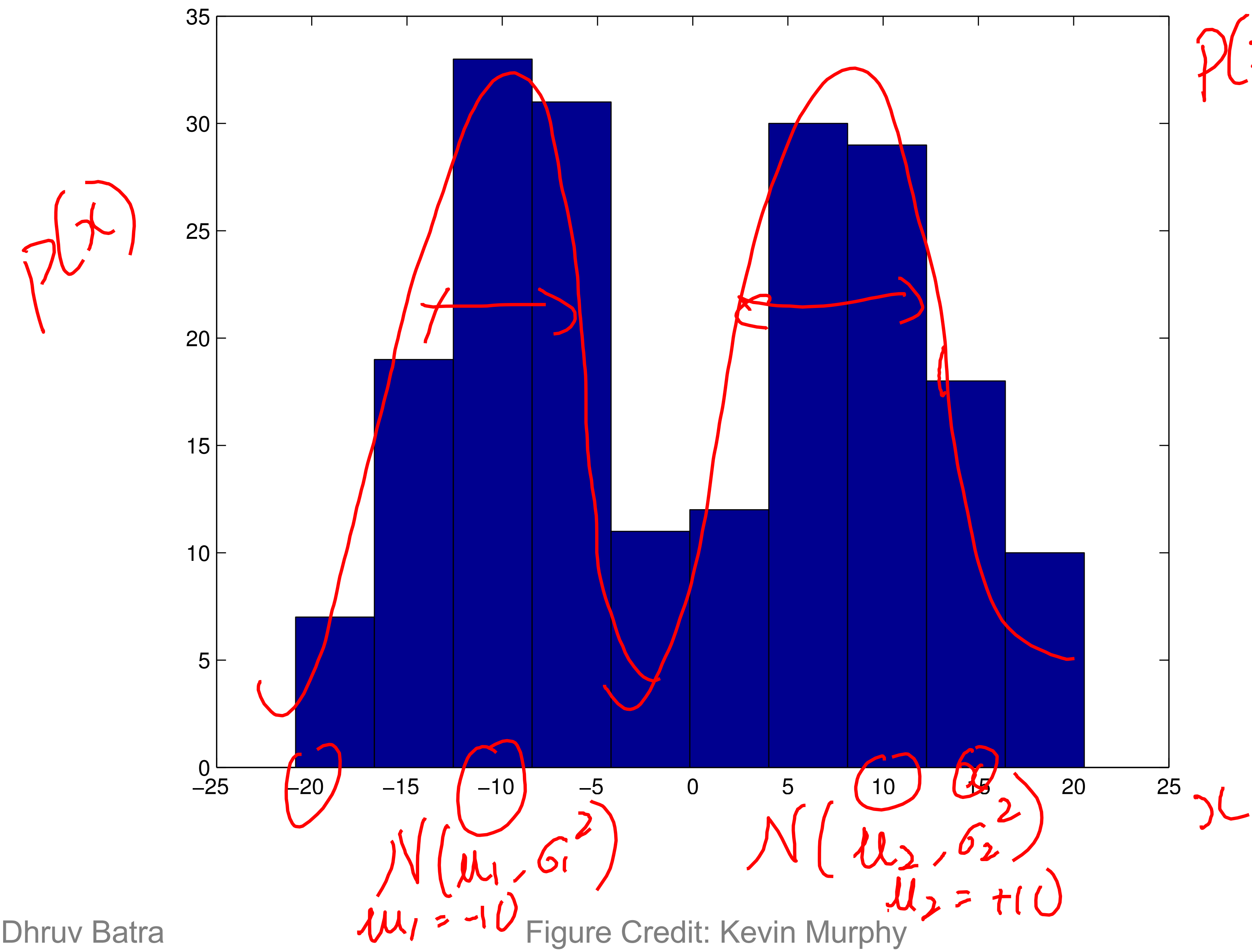
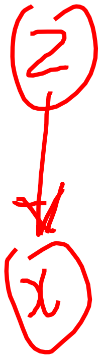


Figure Credit: Kevin Murphy

Gaussian Mixture Model



$$\underline{z \sim \mathrm{Cat}(\vec{\pi})} \qquad \begin{bmatrix} \pi_1 \\ \vdots \\ \pi_k \end{bmatrix} \qquad z \in \{1, \ldots, k\}$$

$$\pi_c = P(z = c)$$

$$X \mid \boxed{z = c} \sim \underline{N(\mu_c, \sigma_c^2)} = \underline{\frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(x-\mu_c)^2}{2\sigma_c^2}}}$$

$$P(\vec{x}) = \underbrace{P(z)}_{\pi_z} \underbrace{P(\vec{x} \mid z)}_{N}$$

Gaussian Mixture Model

$$P(z) = \pi_z$$

$$P(x|z) = N(\mu_z, \sigma_z^2)$$

Available from model

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

$$P(z|\vec{x}) = \frac{P(z,\vec{x})}{P(\vec{x})} = \frac{P(\vec{x}|z)\,P(z)}{\sum (\quad)(\quad)}$$ = "Inference"

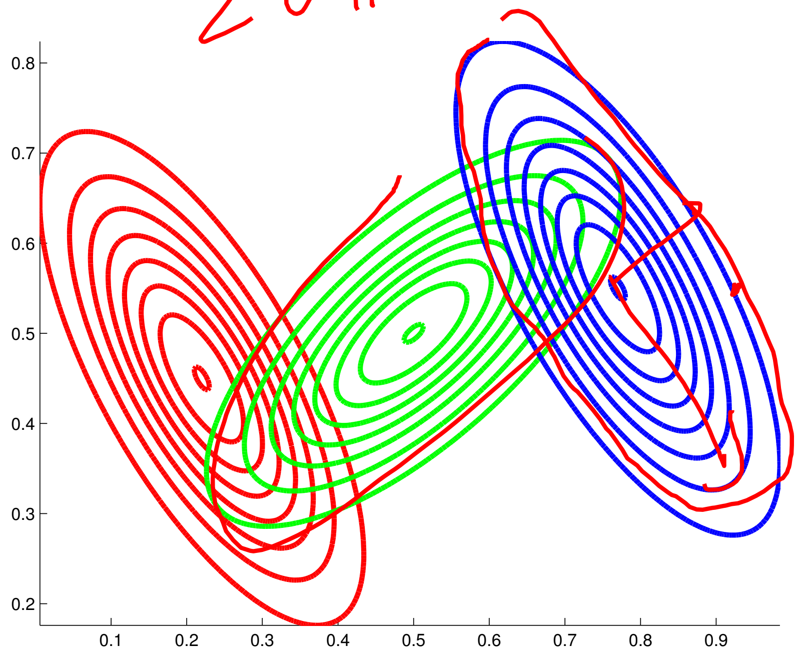
GMM

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

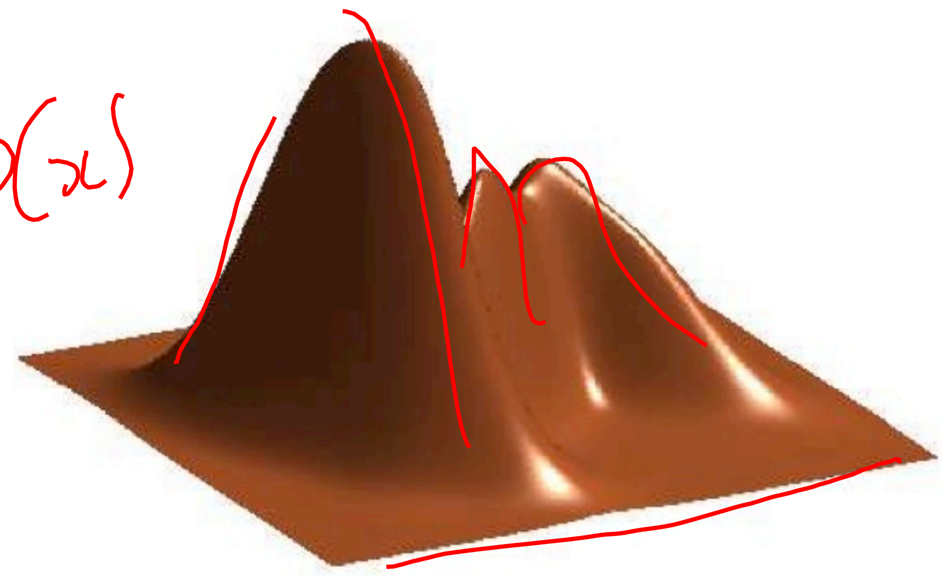
$\vec{\mu} \in \mathbb{R}^d$

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

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



$p(x)$



K-means vs GMM

- K-Means
  - http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

- GMM
  - https://lukapopijac.github.io/gaussian-mixture-model/

Hidden Data Causes Problems #1

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

Parameters: $\{\underbrace{\pi_1 \ldots \pi_k, \mu_1 \ldots \mu_k, \Sigma_1 \ldots \Sigma_k}\} = \theta$

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

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

$$= \sum_i \log P(\vec{x}_i \mid \theta)$$

$$= \sum_i \log \boxed{\sum_{z_i}} P(x_i, z_i \mid \boxed{\theta})$$

$P(x_i \mid z_i)\, P(z_i)$

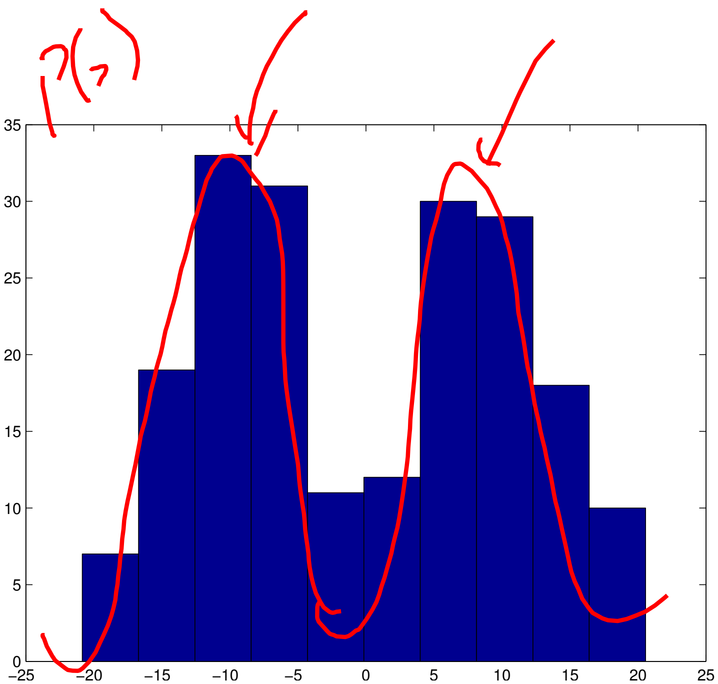
$\log \int$

$P(x_i, z_i \mid \theta)$

$z_i = 1$

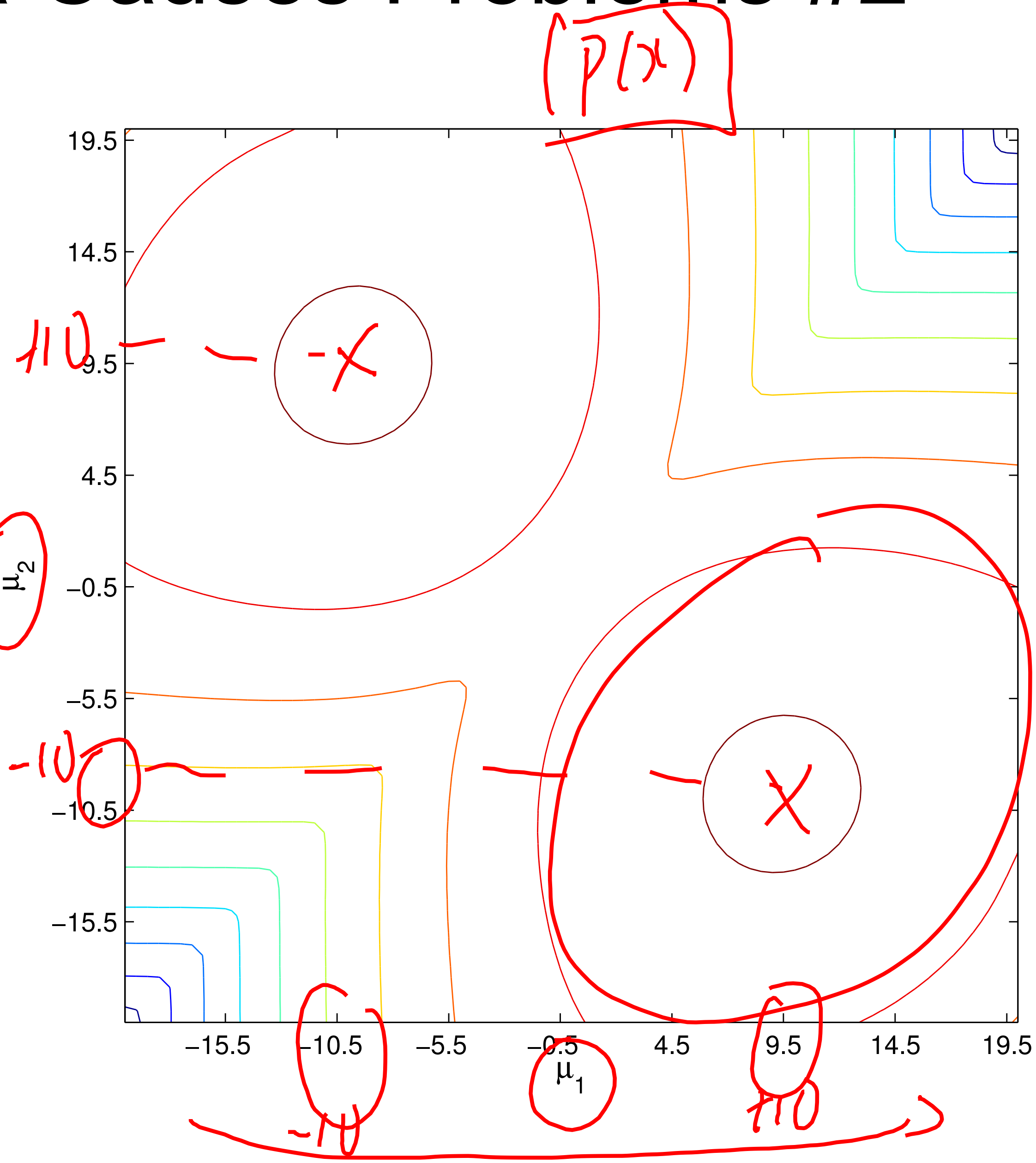
Hidden Data Causes Problems #2

- Identifiability



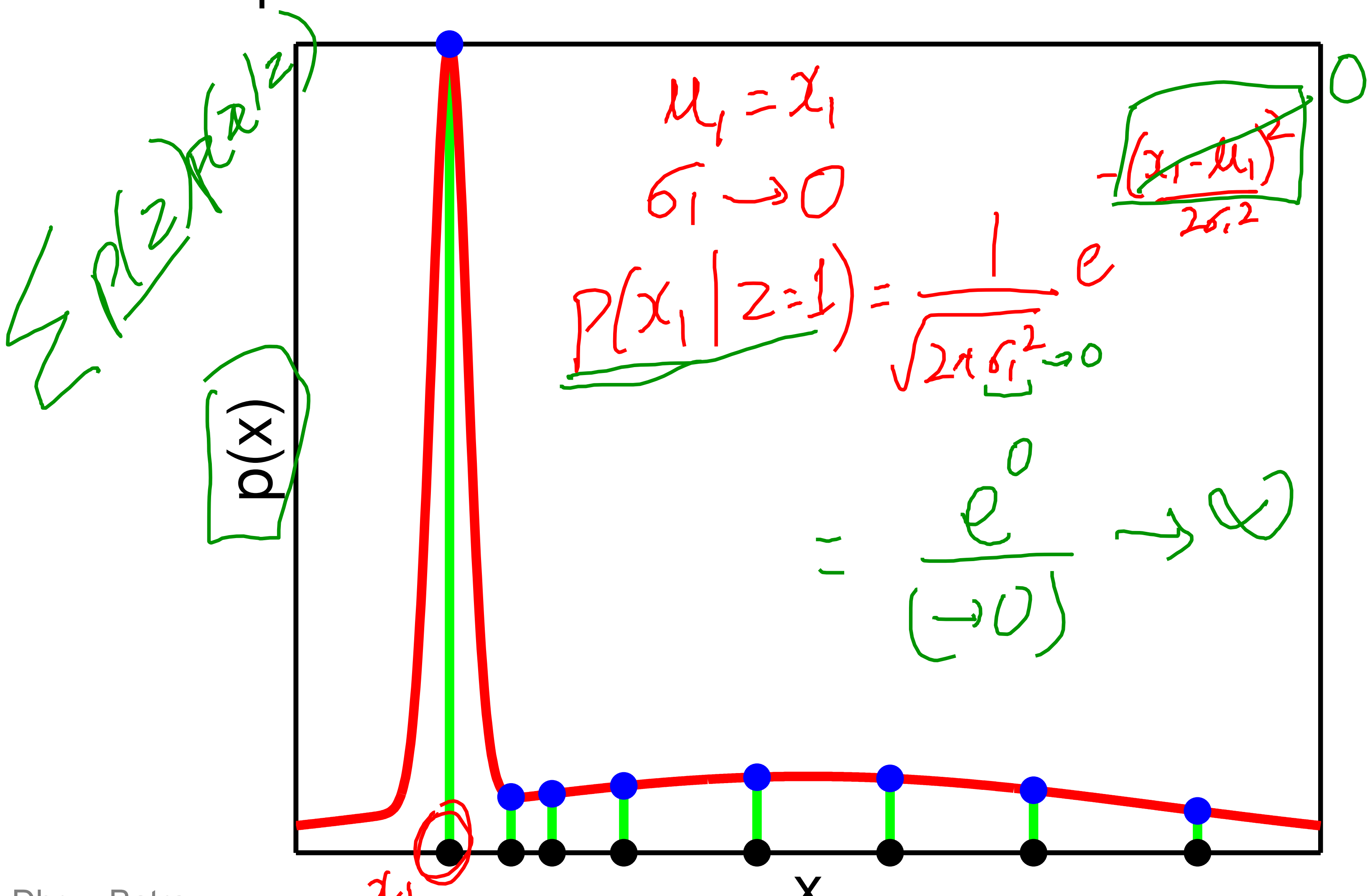
$\mu_1 = -10$
$\mu_2 = +10$

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



Hidden Data Causes Problems #3

- Likelihood has singularities if one Gaussian "collapses"



$z \sim \text{Cat}$

$x \mid z \sim \mathcal{N}$

GMM

Variational Auto Encoders

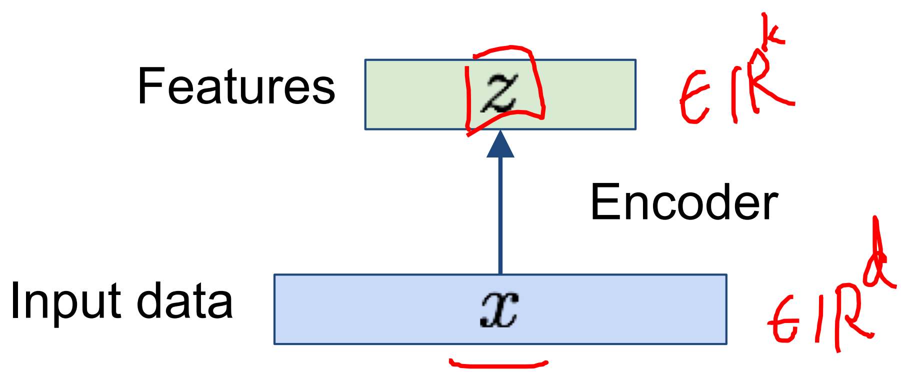
VAEs are a combination of the following ideas: $p(\bar{x})$

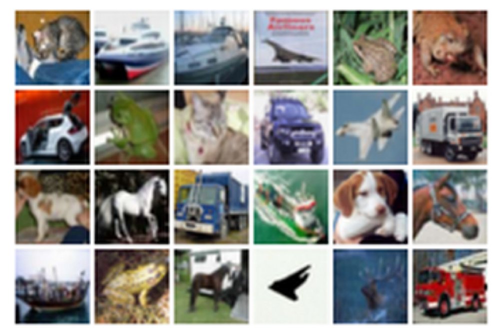
$\sum p(x|z)$
$p(z)$

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

Autoencoders

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

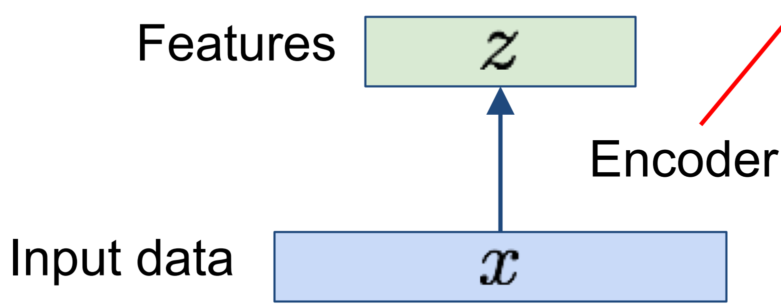


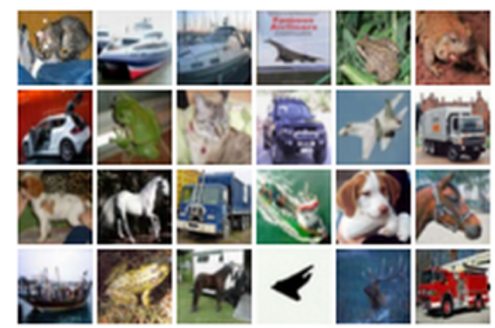


Autoencoders

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

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





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

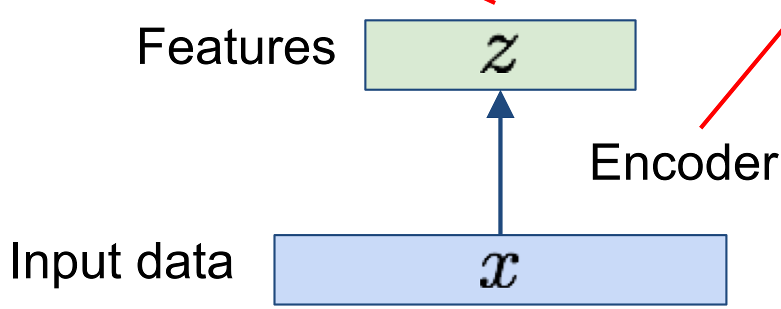
Autoencoders

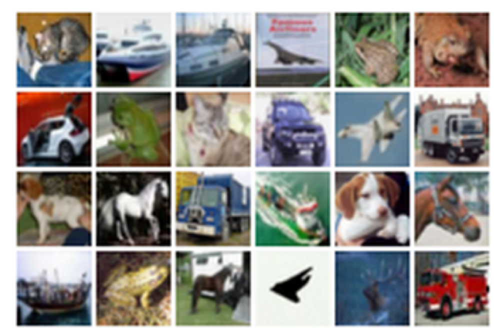
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

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

Q: Why dimensionality reduction?

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





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

Autoencoders

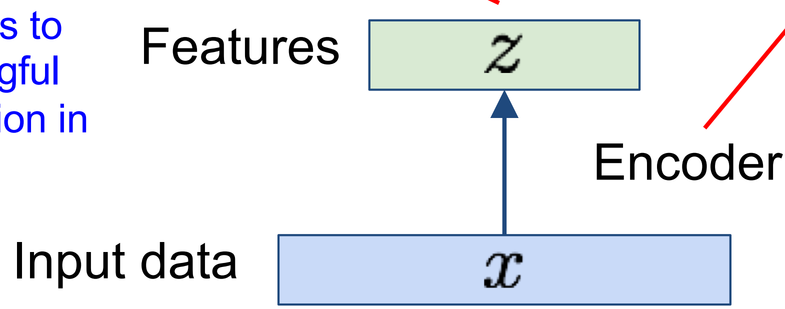
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

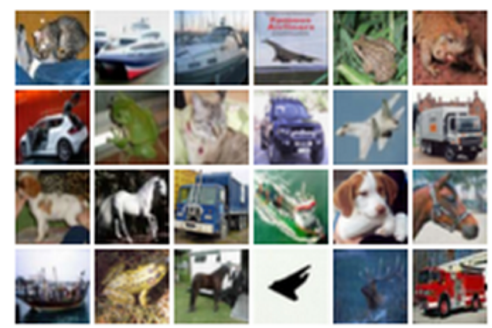
z usually smaller than **x** (dimensionality reduction)

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

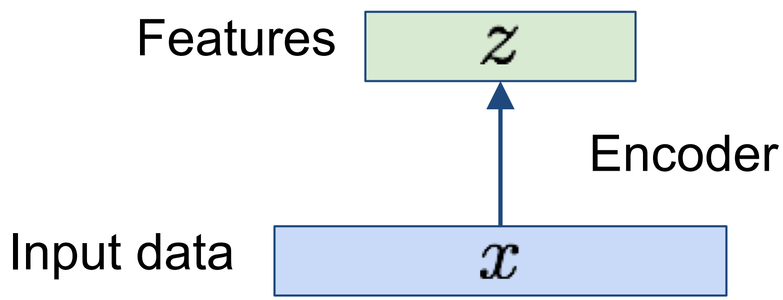


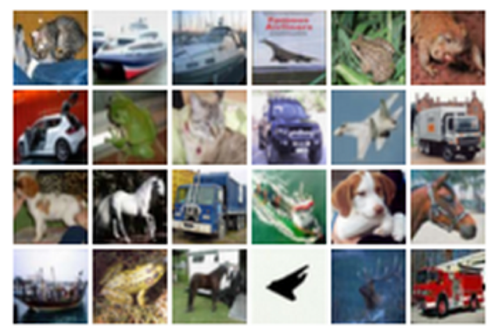


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders

How to learn this feature representation?

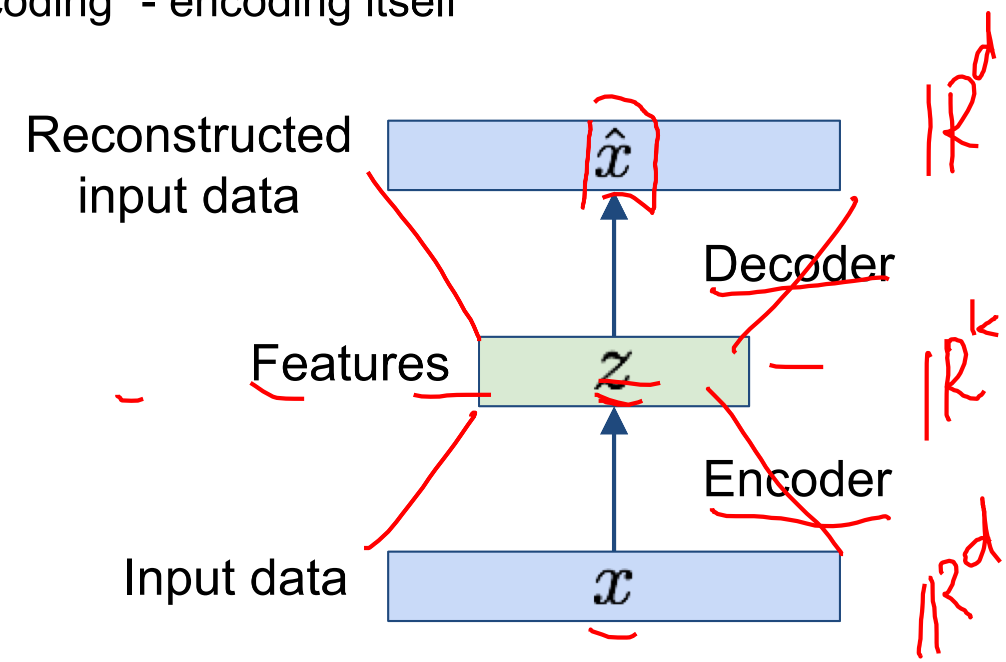


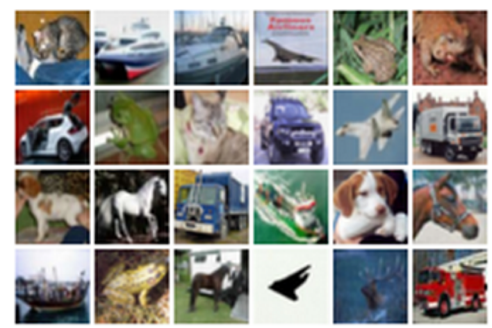


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders

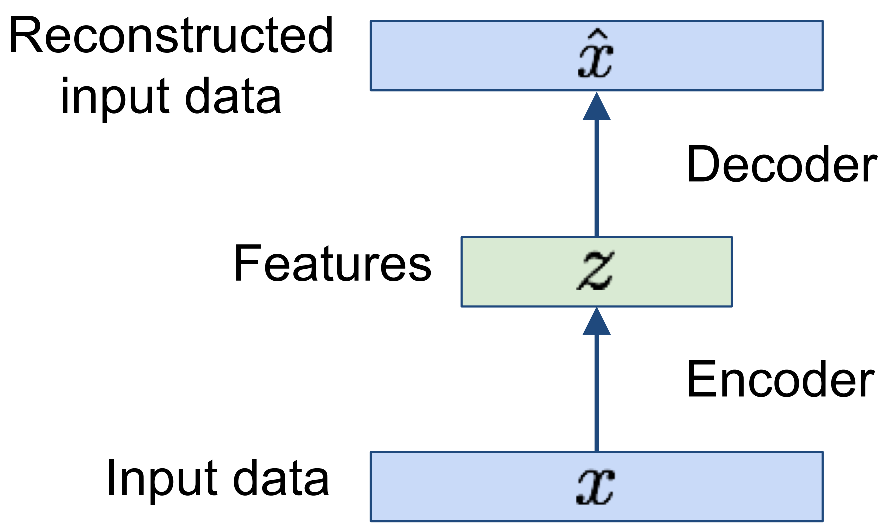
How to learn this feature representation?
Train such that features can be used to reconstruct original data
"Autoencoding" - encoding itself





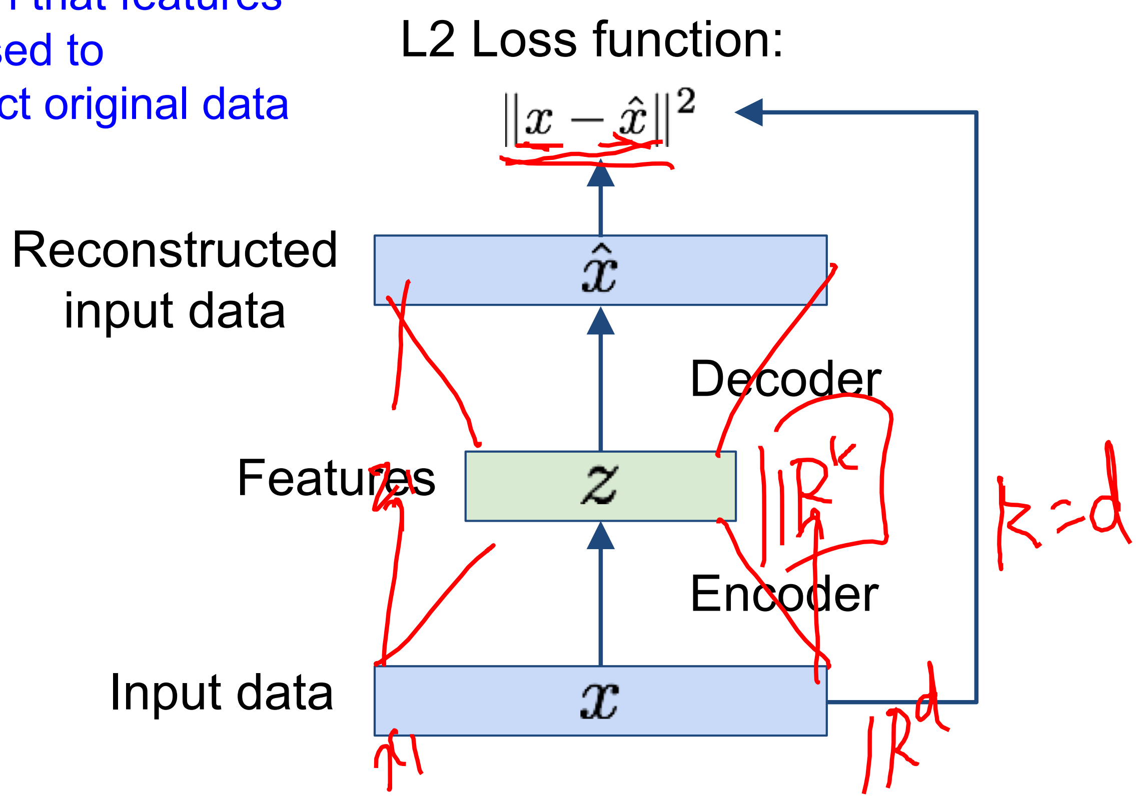
Autoencoders

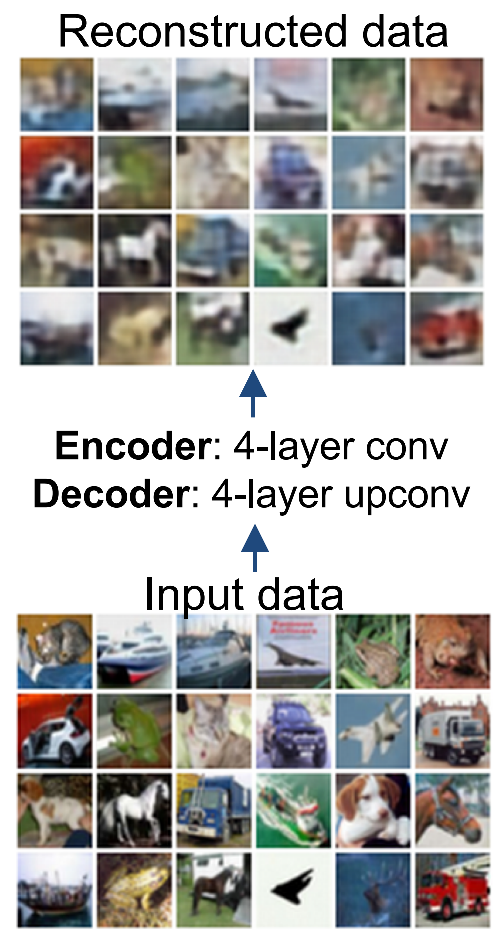
How to learn this feature representation?
Train such that features can be used to reconstruct original data
"Autoencoding" - encoding itself



Autoencoders

Train such that features can be used to reconstruct original data





Autoencoders

Train such that features can be used to reconstruct original data

Doesn't use labels!

L2 Loss function:

$$\|x - \hat{x}\|^2$$

Reconstructed input data $\hat{x}$

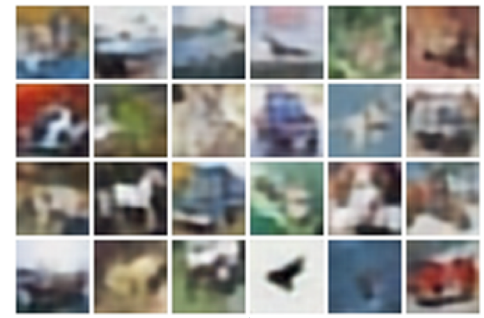
Decoder

Features z

Encoder

Input data x

Reconstructed data



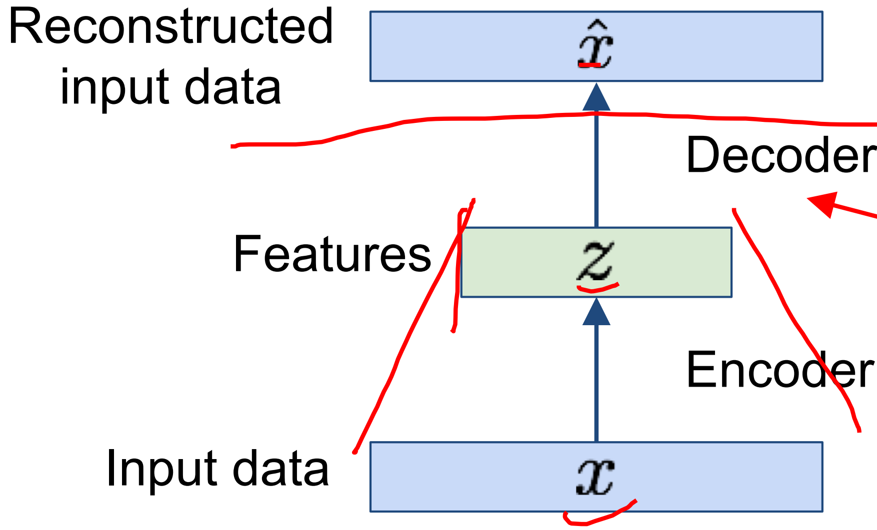
Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data

Autoencoders

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

Autoencoders

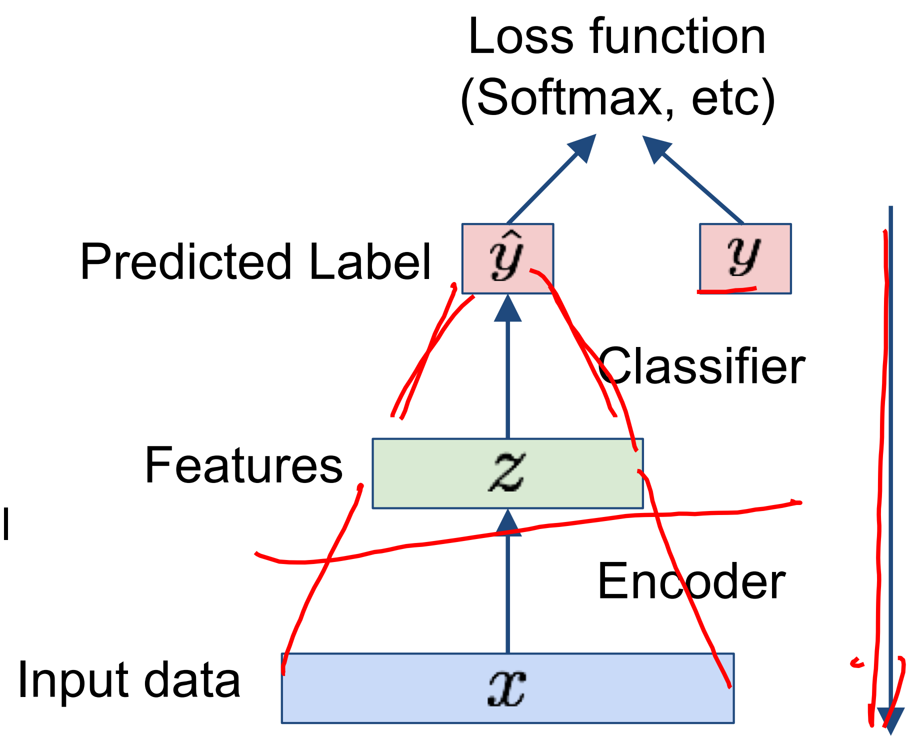


After training, throw away decoder

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

Autoencoders

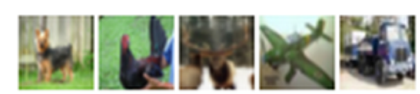
Encoder can be used to initialize a **supervised** model



Fine-tune encoder jointly with classifier

bird plane
dog deer truck

Train for final task (sometimes with small data)



Autoencoders

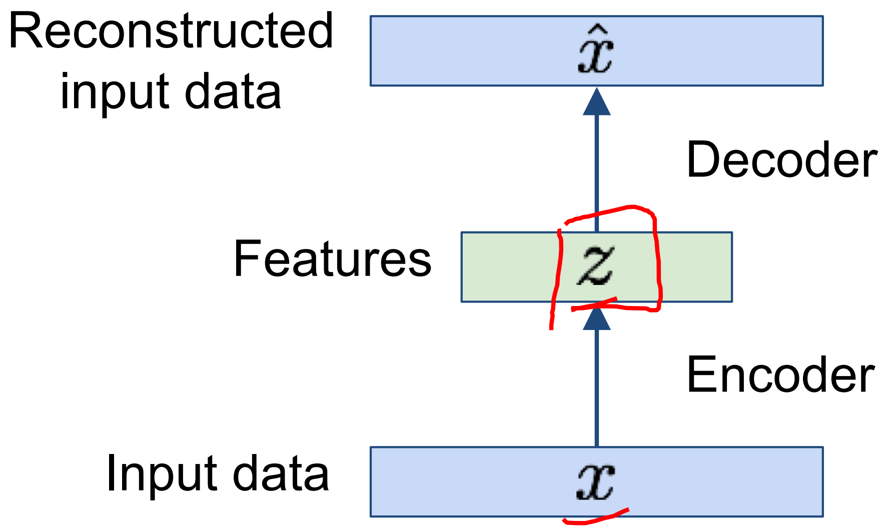
$p(y|x)$

$z = f_\theta(x)$

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

$p(z|x)$

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



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

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

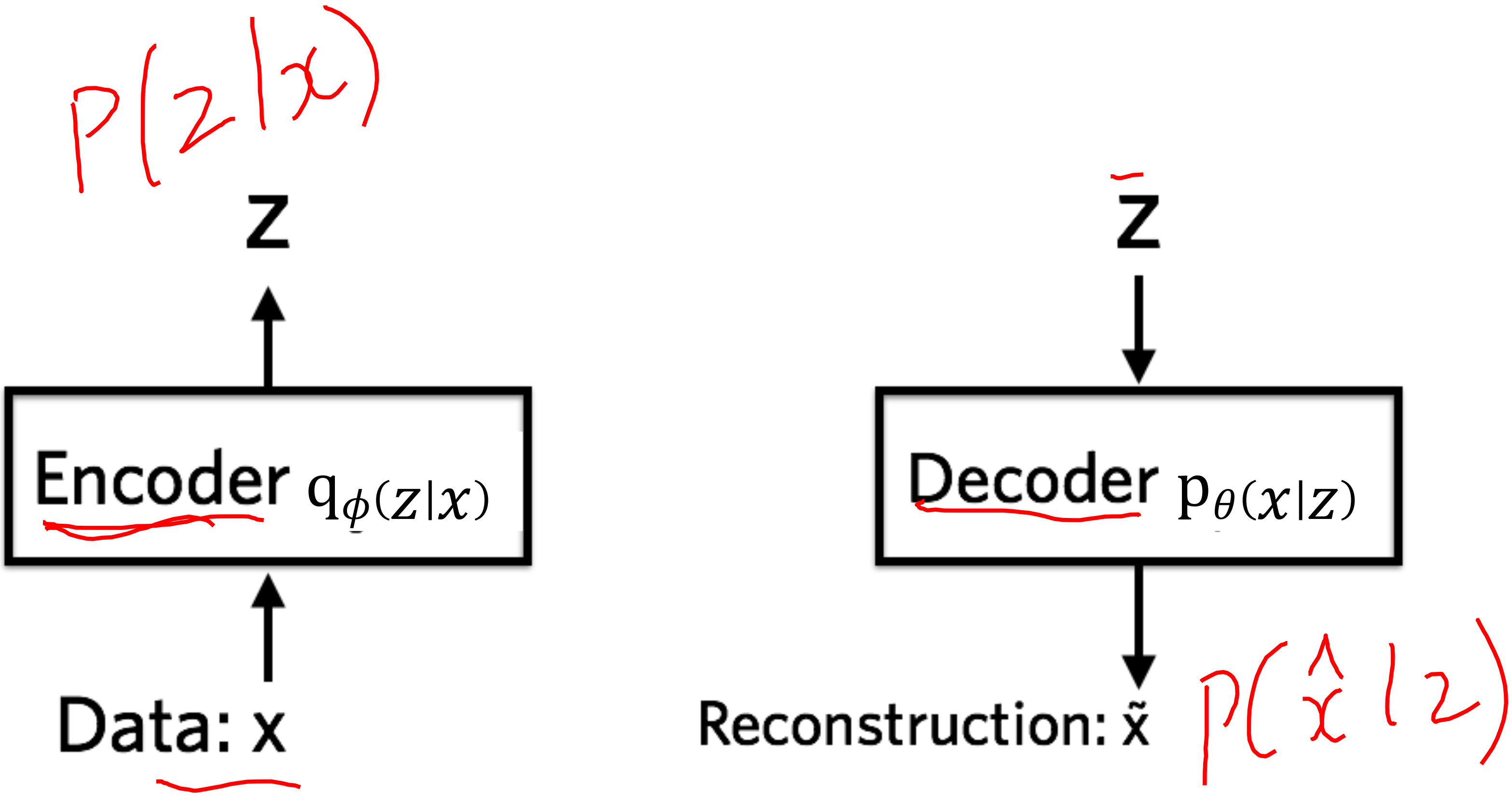


Image Credit: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

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

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

$\int$ Hard

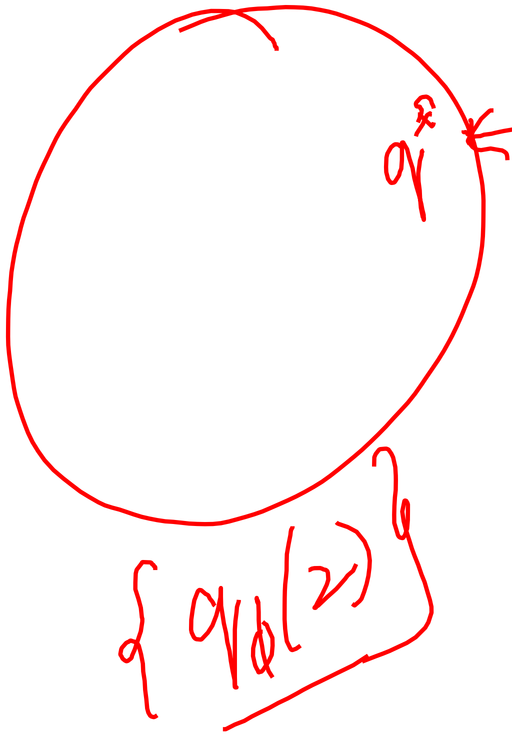
$q(z)$

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



$\boxed{p(z)}$

$$\min_{q \in \{\}} \; d(p, q)$$

$$KL(p(z) \,||\, q(z))$$

$$KL(q(z) \,||\, p(z))$$

$$KL(p \,||\, q) = \sum_z p(z) \log \underbrace{\boxed{\frac{p(z)}{q(z)}}}_{\text{error}} \qquad \sum \underbrace{q(z)}_{\text{support}} \log \boxed{\frac{q(z)}{p(z)}}$$

KL divergence: Distance between distributions

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

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

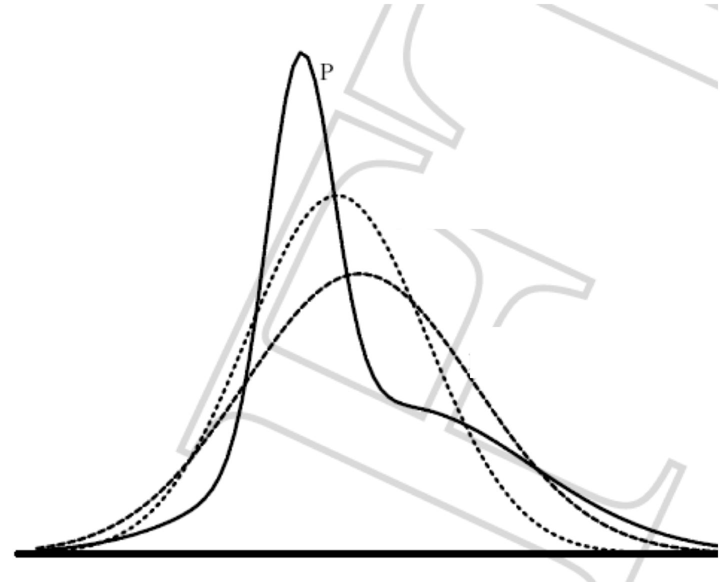
- Not symmetric – p determines where difference is important

Find simple approximate distribution

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

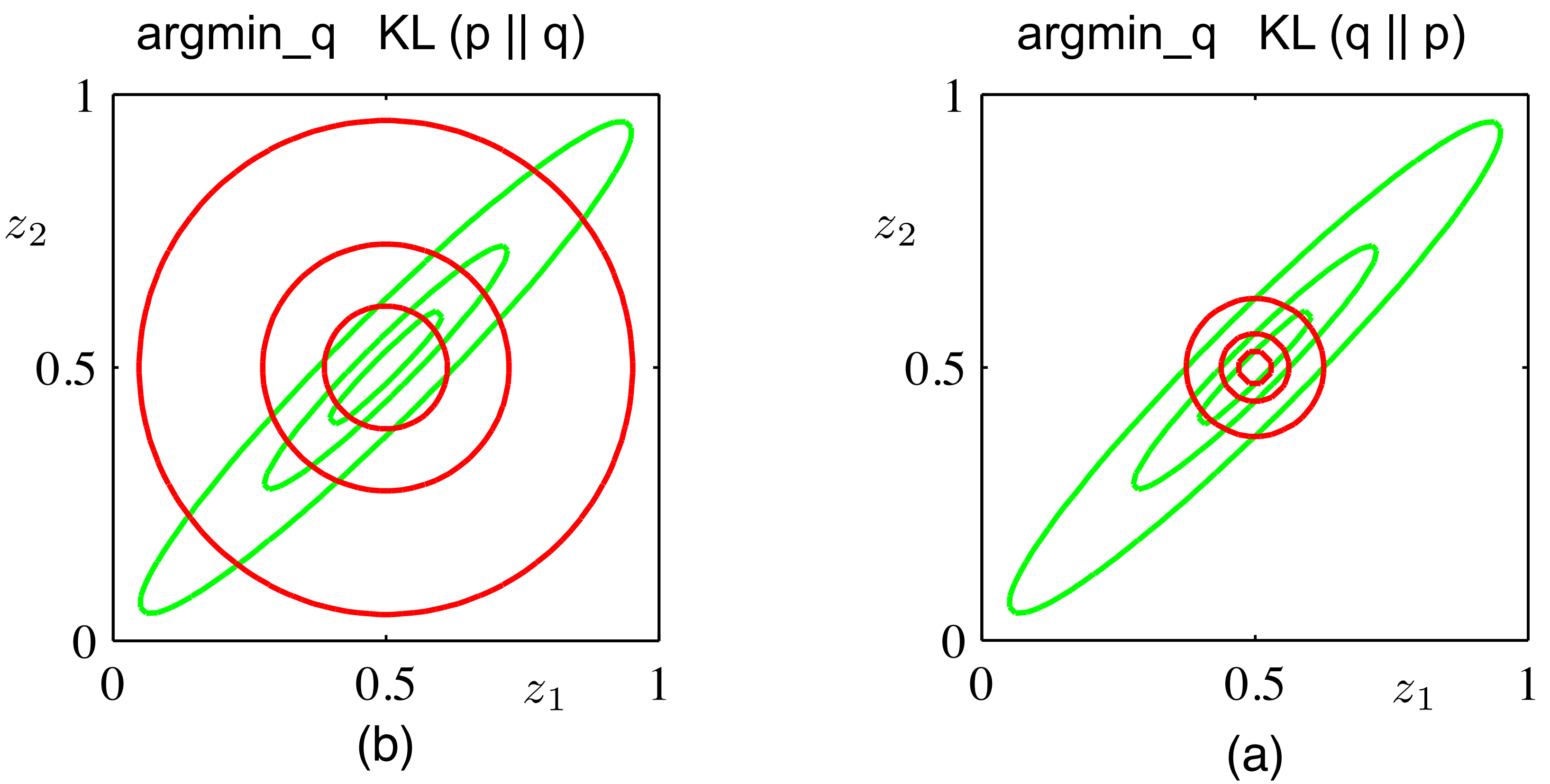
- D(p||q)
  - true distribution p defines support of diff.
  - the “correct” direction
  - will be intractable to compute

- 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



p = Green; q = Red

Example 2

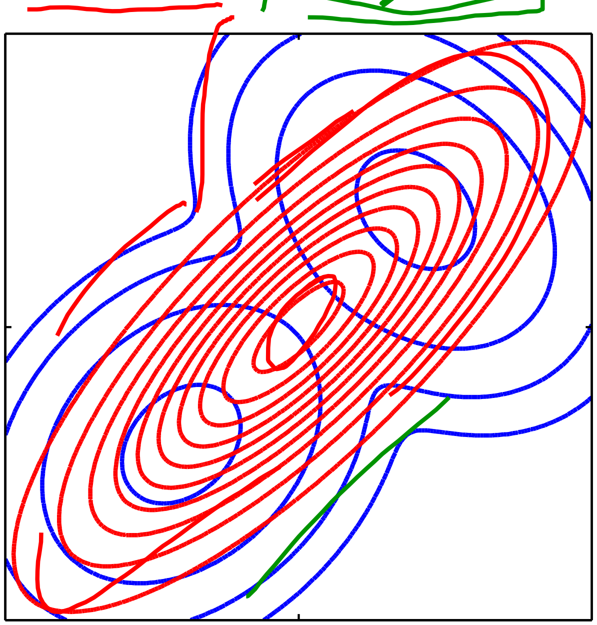
- p = Mixture of Two Gaussians
- q = Single Gaussian   μ, σ

UL

"wrong"

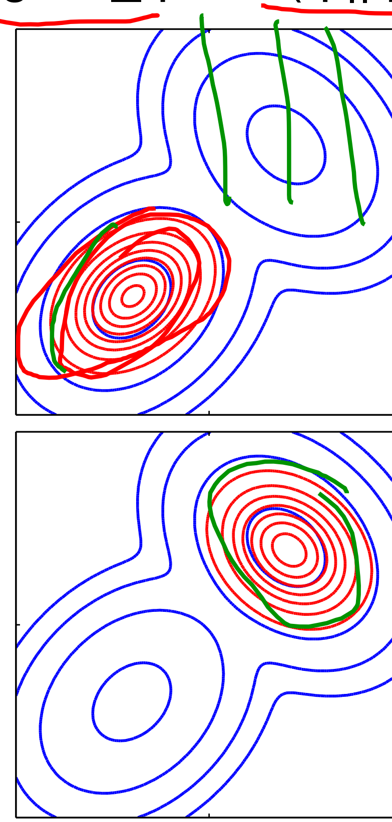
SL → "right"

argmin_q KL (p || q)



argmin_q KL (q || p)

q(z)



p = Blue; q = Red