Machine Learning
Deep Generative Models

Rajesh Ranganath

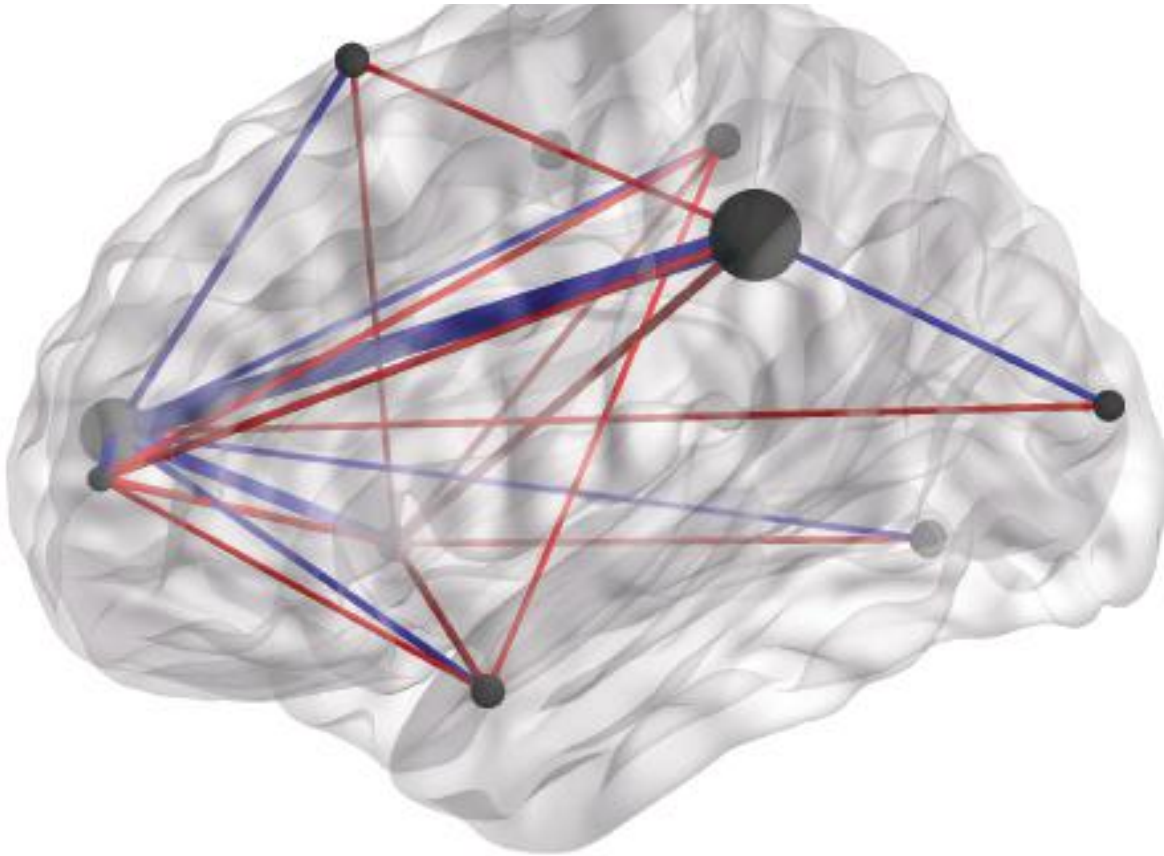
Last Class

We talked about building models

$$\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = p(\mathbf{x})$$

Models can be used to find hidden structure

Neuroscience

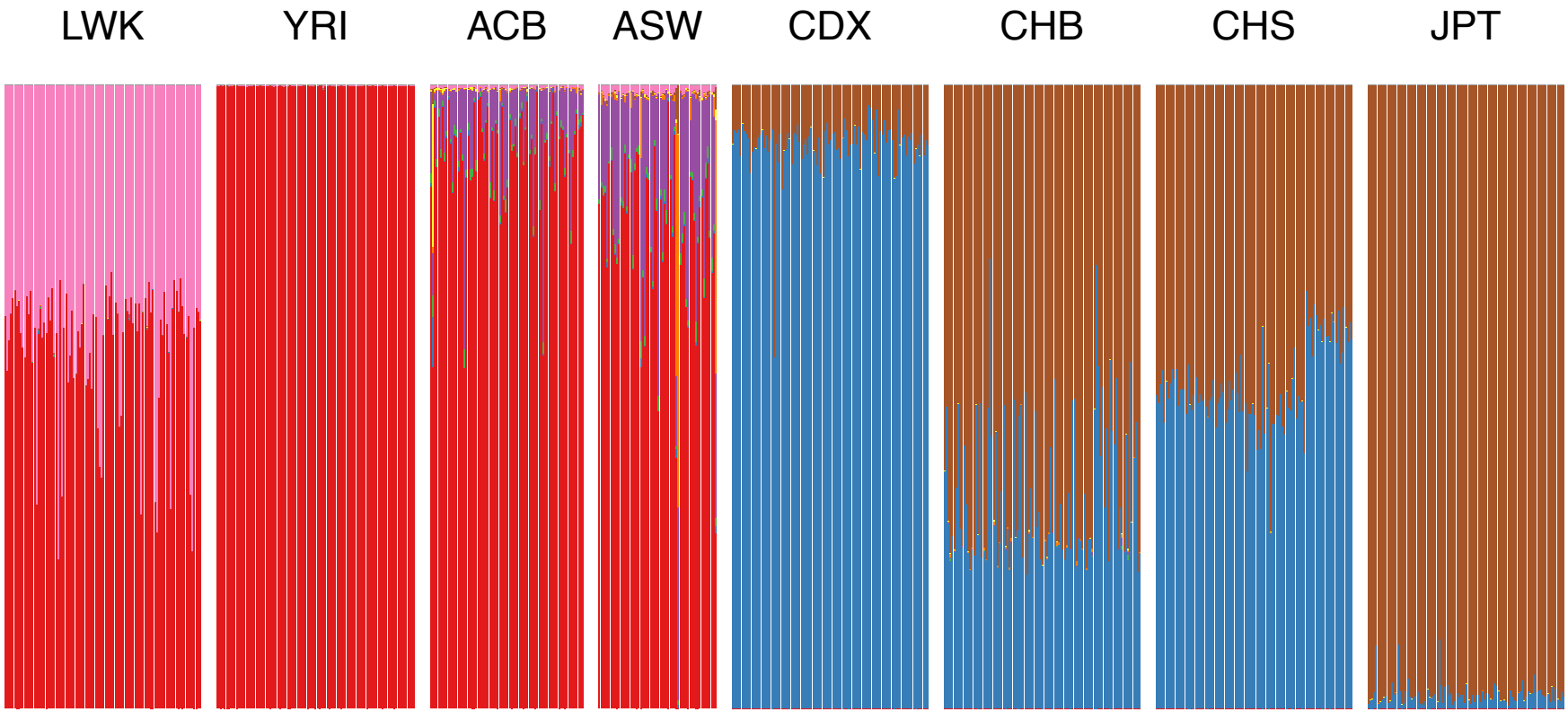


[Manning+ 2014]

Astrophysics

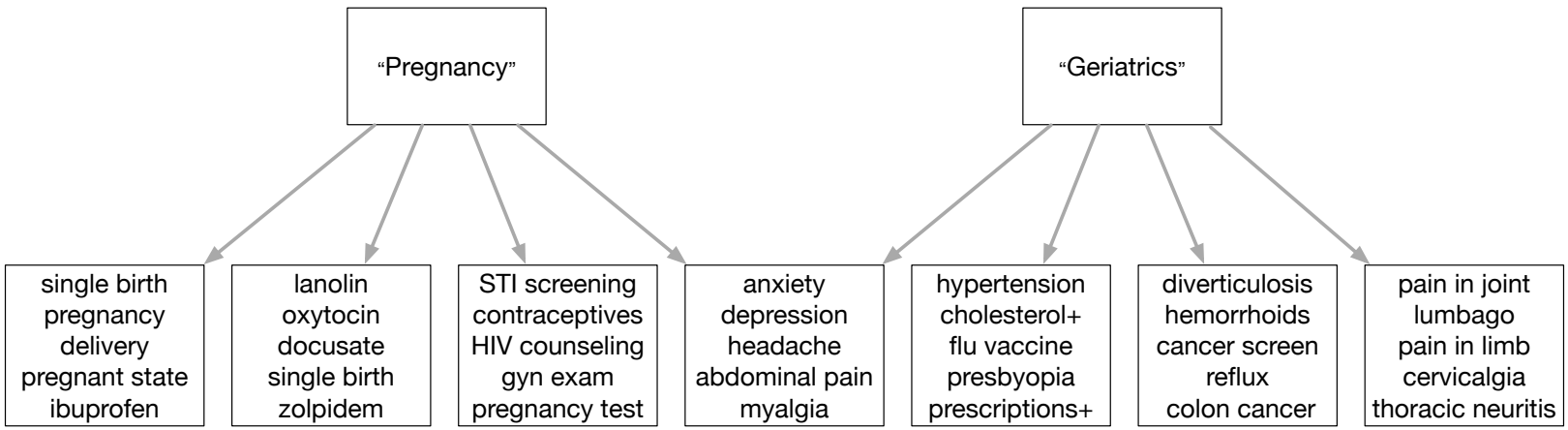
[Regier+ 2015]

Genetics



[Gopalan+ 2016]

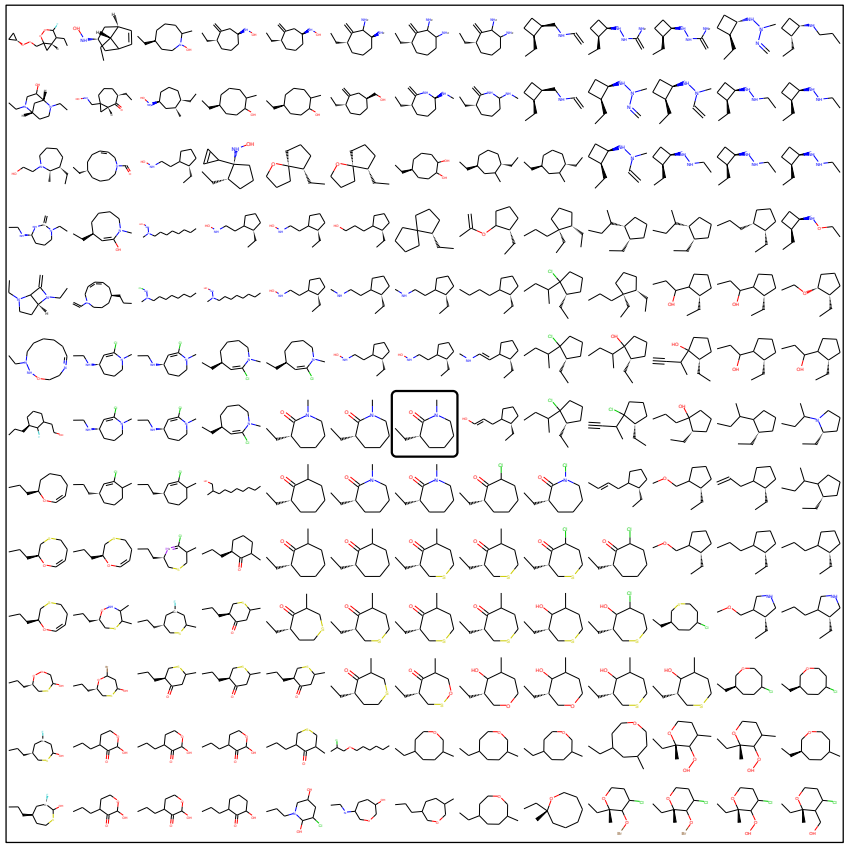
Medicine

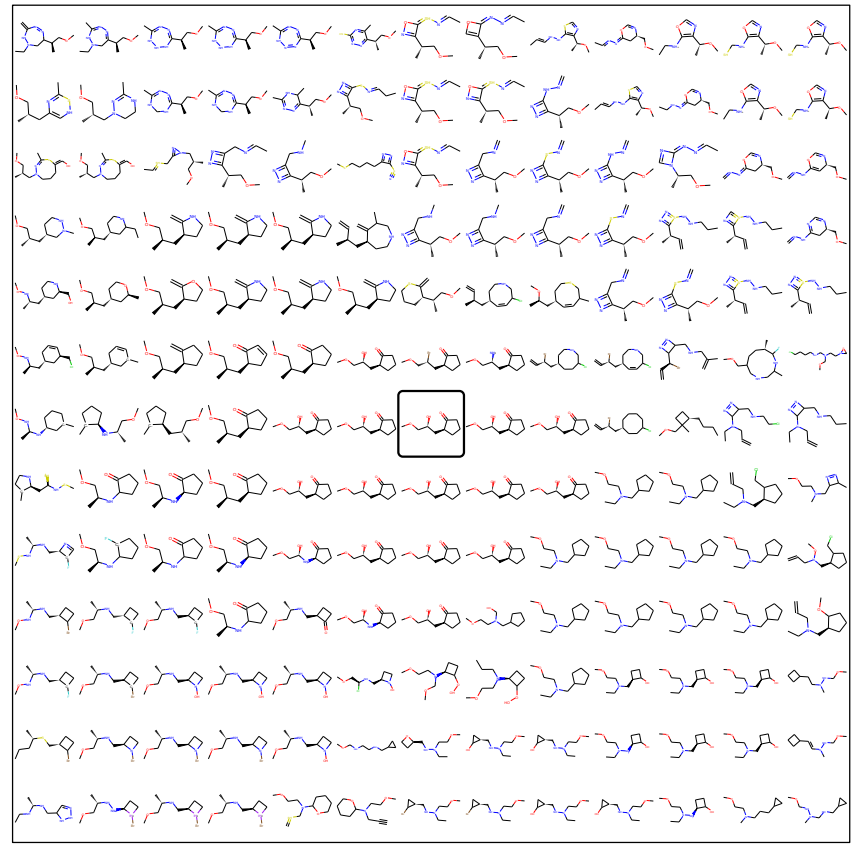

“Pregnancy”
“Geriatrics”
single birth
pregnancy
delivery
pregnant state
ibuprofen
lanolin
oxytocin
docusate
single birth
zolpidem
STI screening
contraceptives
HIV counseling
gyn exam
pregnancy test
anxiety
depression
headache
abdominal pain
myalgia
hypertension
cholesterol+
flu vaccine
presbyopia
prescriptions+
diverticulosis
hemorrhoids
cancer screen
reflux
colon cancer
pain in joint
lumbago
pain in limb
cervicalgia
thoracic neuritis

[R+]

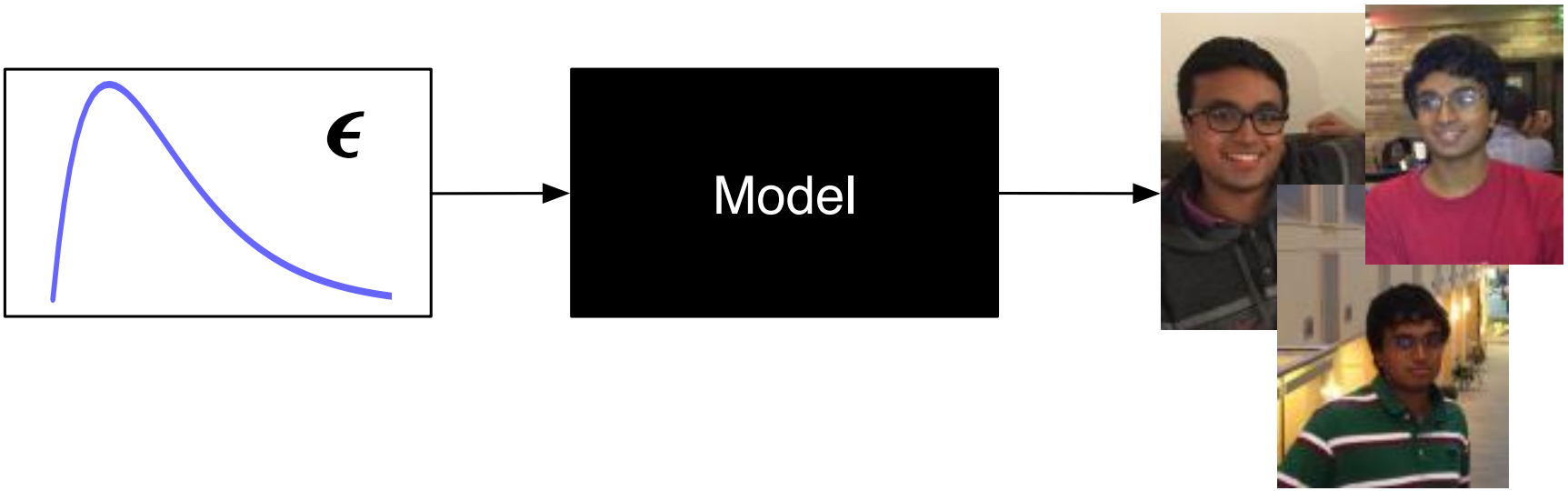
What about sampling from $p(\mathbf{x})$ directly?

Sampling Molecules





[Kusner+ 2017]

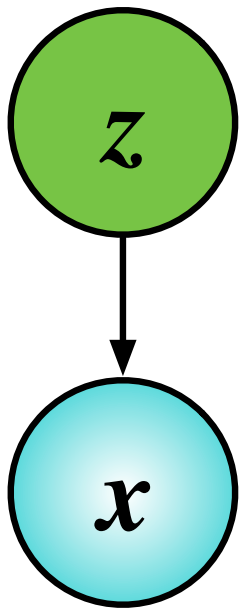

ϵ
Model

Roadmap

- **Variational Autoencoders**
- **Direct Likelihood Models**
- **Generative Adversarial Networks**

Variational Autoencoders

Variational Autoencoder (VAE)



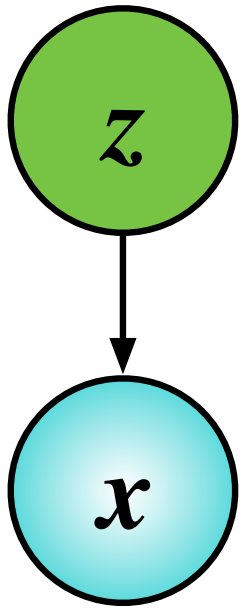
$$p(\mathbf{z}) = \text{Normal}(0, 1)$$

$$p(\mathbf{x}|\mathbf{z}) = \text{Normal}(\mu_\beta(\mathbf{z}), \sigma^2_\beta(\mathbf{z}))$$

μ and σ^2 are deep networks with parameters $\boldsymbol{\beta}$.

[Kingma+ 2014; Rezende+ 2014]

Variational Autoencoder (VAE)

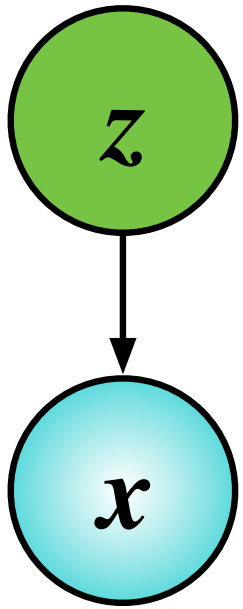


$$p(\mathbf{z}) = \text{Normal}(0, 1)$$

$$p(\mathbf{x}|\mathbf{z}) = \text{Normal}(\mu_\beta(\mathbf{z}), \sigma^2_\beta(\mathbf{z}))$$

Need to learn parameters $\boldsymbol{\beta}$ How?

Variational Autoencoder (VAE)



$$p(\mathbf{z}) = \text{Normal}(0, 1)$$

$$p(\mathbf{x}|\mathbf{z}) = \text{Normal}(\mu_\beta(\mathbf{z}), \sigma^2_\beta(\mathbf{z}))$$

Need to learn parameters $\boldsymbol{\beta}$ How? Maximum likelihood

$$\sum_{i=1}^{n} \log p_{\boldsymbol{\beta}}(\mathbf{x}_i) = \sum_{i=1}^{n} \log \int p_{\boldsymbol{\beta}}(\mathbf{x}_i, \mathbf{z}_i) d\mathbf{z}_i$$

is intractable.

Use Variational Inference!

Evidence Lower Bound (ELBO)

For a single point, would like to

$$\max_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{x}) = \max_{\boldsymbol{\beta}} \log \int p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Integral is intractable

$$\log p_{\boldsymbol{\beta}}(\mathbf{x}) \quad = \quad \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\log p_{\boldsymbol{\beta}}(\mathbf{x})] = \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})]$$

Evidence Lower Bound (ELBO)

For a single point, would like to

$$\max_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{x}) = \max_{\boldsymbol{\beta}} \log \int p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Integral is intractable

$$\begin{aligned} \log p_{\boldsymbol{\beta}}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\log p_{\boldsymbol{\beta}}(\mathbf{x})] = \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})] \\ &= \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) + \log q(\mathbf{z}; \boldsymbol{\lambda}) - \log q(\mathbf{z}; \boldsymbol{\lambda}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})] \end{aligned}$$

Evidence Lower Bound (ELBO)

For a single point, would like to

$$\max_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{x}) = \max_{\boldsymbol{\beta}} \log \int p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Integral is intractable

$$\begin{aligned}
\log p_{\boldsymbol{\beta}}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\log p_{\boldsymbol{\beta}}(\mathbf{x})] = \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) + \log q(\mathbf{z}; \boldsymbol{\lambda}) - \log q(\mathbf{z}; \boldsymbol{\lambda}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] + \mathbb{E}_q[\log q(\mathbf{z}; \boldsymbol{\lambda}) - \log p(\mathbf{z} \mid \mathbf{x})]
\end{aligned}$$

Evidence Lower Bound (ELBO)

For a single point, would like to

$$\max_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{x}) = \max_{\boldsymbol{\beta}} \log \int p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Integral is intractable

$$\begin{aligned}
\log p_{\boldsymbol{\beta}}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\log p_{\boldsymbol{\beta}}(\mathbf{x})] = \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) + \log q(\mathbf{z}; \boldsymbol{\lambda}) - \log q(\mathbf{z}; \boldsymbol{\lambda}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] + \mathbb{E}_q[\log q(\mathbf{z}; \boldsymbol{\lambda}) - \log p(\mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] + KL(q(\mathbf{z}; \boldsymbol{\lambda}) || p(\mathbf{z} \mid \mathbf{x}))
\end{aligned}$$

Evidence Lower Bound (ELBO)

For a single point, would like to

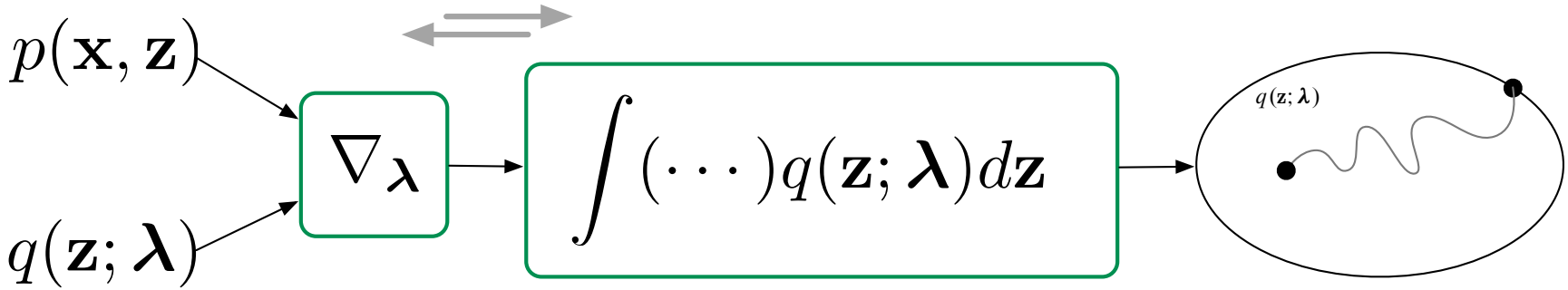
$$\max_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{x}) = \max_{\boldsymbol{\beta}} \log \int p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Integral is intractable

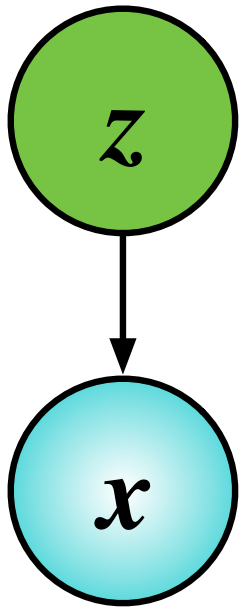
$$\begin{aligned}
\log p_{\boldsymbol{\beta}}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\log p_{\boldsymbol{\beta}}(\mathbf{x})] = \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_q[\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) + \log q(\mathbf{z}; \boldsymbol{\lambda}) - \log q(\mathbf{z}; \boldsymbol{\lambda}) - \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] + \mathbb{E}_q[\log q(\mathbf{z}; \boldsymbol{\lambda}) - \log p(\mathbf{z} \mid \mathbf{x})] \\
&= \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] + KL(q(\mathbf{z}; \boldsymbol{\lambda}) || p(\mathbf{z} \mid \mathbf{x})) \\
&\geq \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})]
\end{aligned}$$

Optimize lower bound on $\log p$ with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$

The New VI Recipe



Variational Autoencoder (VAE)



$$p(\mathbf{z}) = \text{Normal}(0, 1)$$

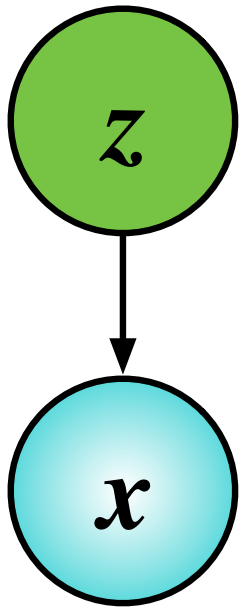
$$p(\mathbf{x}|\mathbf{z}) = \text{Normal}(\mu_\beta(\mathbf{z}), \sigma^2_\beta(\mathbf{z}))$$

Joint:

$$p(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{n} p(\mathbf{x}_i, \mathbf{z}_i) = \prod_{i=1}^{n} p(\mathbf{z}_i) p(\mathbf{x}_i \,|\, \mathbf{z}_i)$$

- $\mathbf{z}_i \sim \text{Normal}(0, 1)$
- $\mathbf{x}_i \,|\, \mathbf{z}_i \sim \text{Normal}(\mu_{\boldsymbol{\beta}}(\mathbf{z}_i), \sigma_{\boldsymbol{\beta}}(\mathbf{z}_i))$

Variational Autoencoder (VAE)



$$p(\mathbf{z}) = \text{Normal}(0, 1)$$

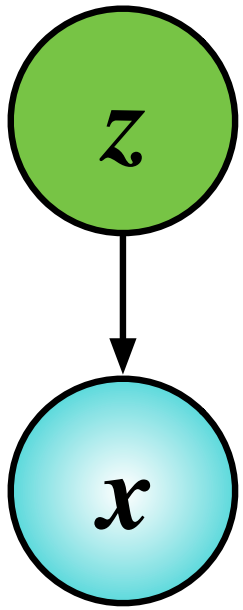
$$p(\mathbf{x}|\mathbf{z}) = \text{Normal}(\mu_\beta(\mathbf{z}), \sigma^2_\beta(\mathbf{z}))$$

Variational approximation:

$$q(\mathbf{z}) = \prod_{i=1}^{n} q(\mathbf{z}_i; \mathbf{m}_i, \mathbf{s}_i)$$

- $q(\mathbf{z}_i; \mathbf{m}_i, \mathbf{s}_i) = \text{Normal}(\mathbf{m}_i, \mathbf{s}_i)$
- Each data point has separate variational parameters

Variational Autoencoder (VAE)



$$p(\mathbf{z}) = \text{Normal}(0, 1)$$

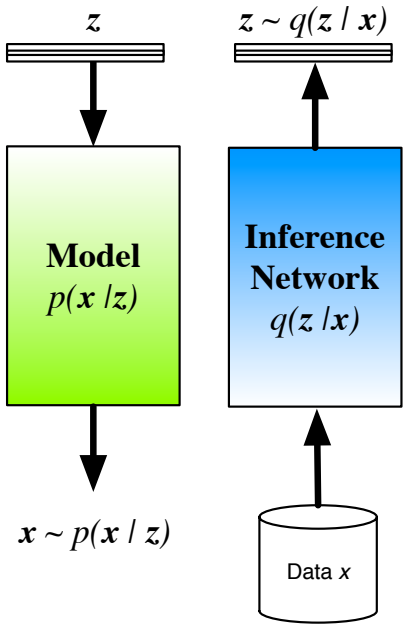
$$p(\mathbf{x}|\mathbf{z}) = \text{Normal}(\mu_\beta(\mathbf{z}), \sigma^2_\beta(\mathbf{z}))$$

Variational Objective:

$$\mathscr{L}(q, \boldsymbol{\beta}) = \mathbb{E}_q[\log p - \log q] = \sum_{i=1}^{n} \mathbb{E}_q[\log p(\mathbf{z}_i, \mathbf{x}_i) - \log q(\mathbf{z}_i; \mathbf{m}_i, \mathbf{s}_i)]$$

- Use reparameterization gradients
- Can subsample data!
- Any problems?

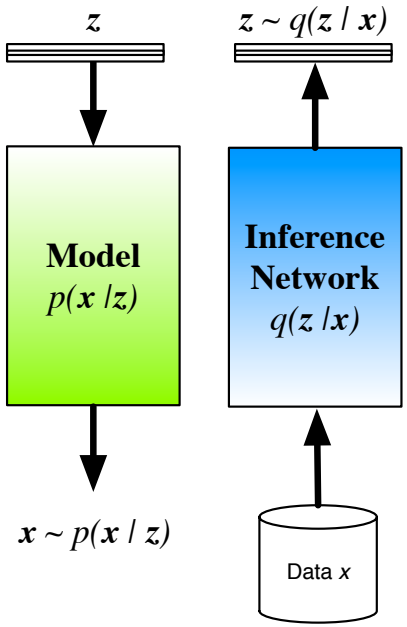
Variational Autoencoder (VAE): Inference Networks



$$q(\mathbf{z}|\mathbf{x}) = \text{Normal}(f_\theta^\mu(\mathbf{x}), f_\theta^{\sigma^2}(\mathbf{x}))$$

All functions are deep networks

Variational Autoencoder (VAE): Inference Networks



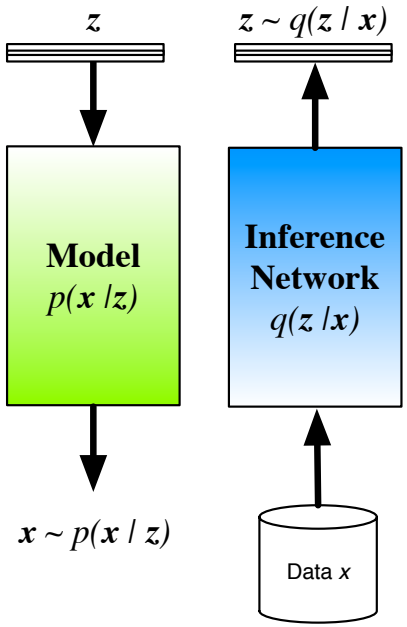
$$q(\mathbf{z}|\mathbf{x}) = \text{Normal}(f_\theta^\mu(\mathbf{x}), f_\theta^{\sigma^2}(\mathbf{x}))$$

Variational Objective:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\theta}) &= \mathbb{E}_q[\log p - \log q] \\ &= \sum_{i=1}^{n} \mathbb{E}_q[\log p(\mathbf{z}_i, \mathbf{x}_i) - \log q(\mathbf{z}_i; \mathbf{m}_i = f_{\boldsymbol{\theta}}^\mu(\mathbf{x}_i), \mathbf{s}_i = f_{\boldsymbol{\theta}}^\sigma(\mathbf{x}_i))]\end{aligned}$$

Converts each local parameter into neural network parameters

Variational Autoencoder (VAE): Inference Networks



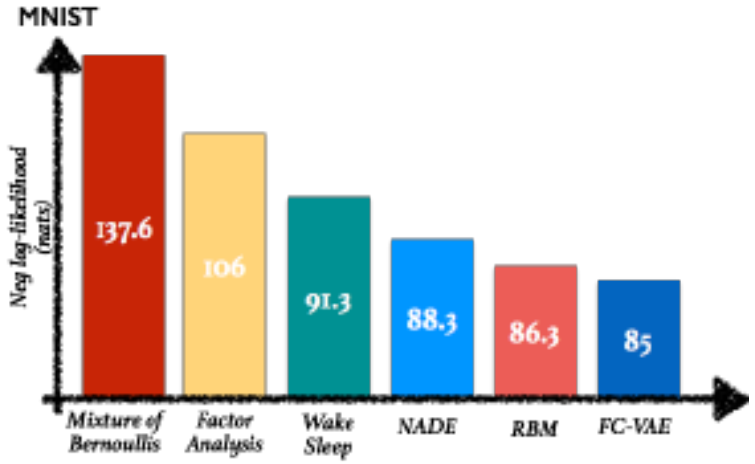
$$q(\mathbf{z}|\mathbf{x}) = \text{Normal}(f_\theta^\mu(\mathbf{x}), f_\theta^{\sigma^2}(\mathbf{x}))$$

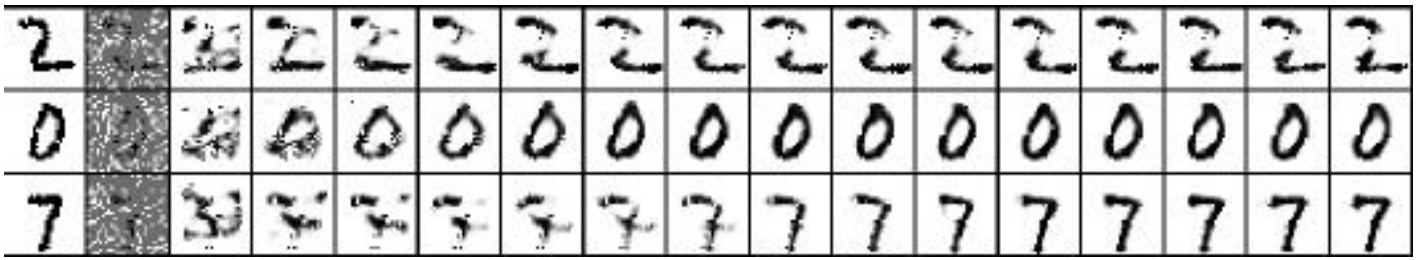
Variational Objective:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\theta}) &= \mathbb{E}_q[\log p - \log q] \\ &= \sum_{i=1}^{n} \mathbb{E}_q[\log p(\mathbf{z}_i, \mathbf{x}_i) - \log q(\mathbf{z}_i; \mathbf{m}_i = f_{\boldsymbol{\theta}}^\mu(\mathbf{x}_i), \mathbf{s}_i = f_{\boldsymbol{\theta}}^\sigma(\mathbf{x}_i))]\end{aligned}$$

Optimize using reparameterization gradients

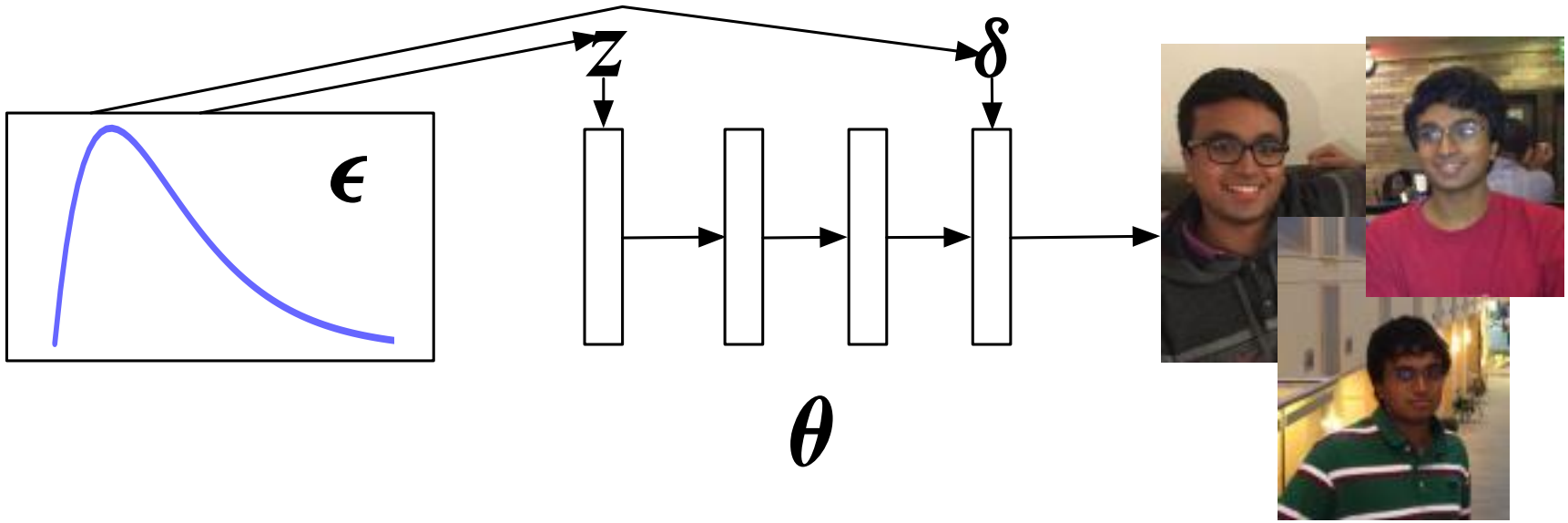
Variational Autoencoder (VAE)





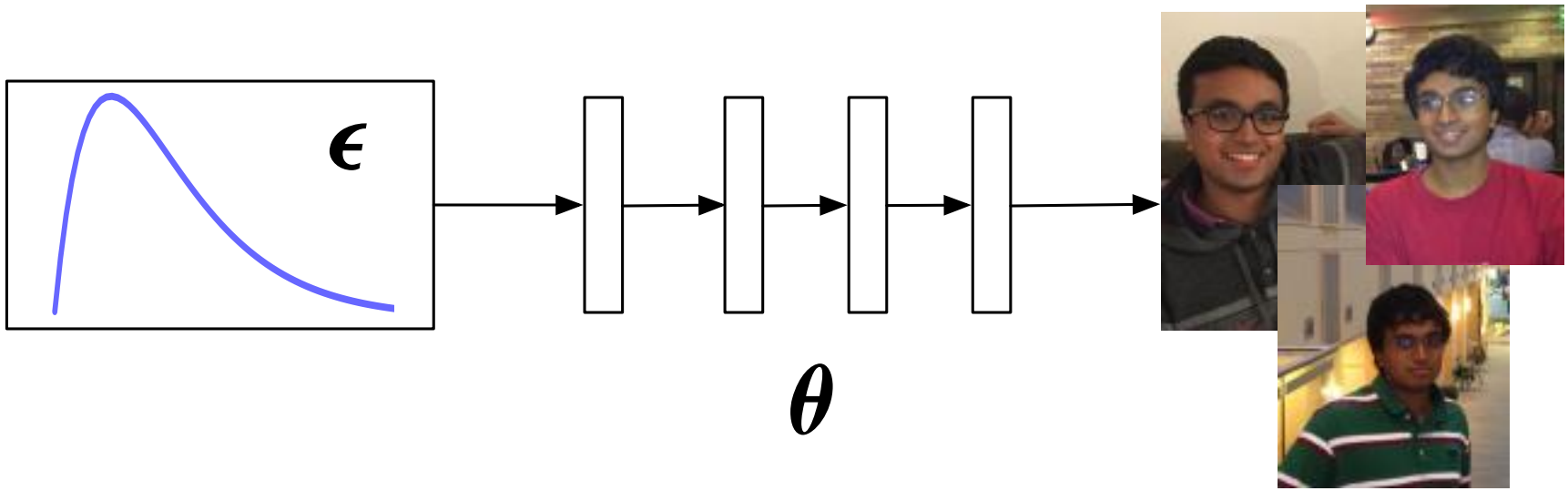
Direct Likelihood Models

Deep generative model with latent variables



- Challenge is in accurate inference of **z** given **x**
- Inference is needed because amount of noise is larger than data

Deep generative model with Noise



- Try to model density directly
- $\mathbf{x} \sim p_{\theta}(\mathbf{x}) \iff \mathbf{x} = f(\boldsymbol{\theta}, \boldsymbol{\epsilon})$

Maximum Likelihood

How do we learn a model $p_{\boldsymbol{\theta}}$ given samples $\mathbf{x}_1, ..., \mathbf{x}_n$?

1. Write down a model $p_{\boldsymbol{\theta}}(x)$
2. Write the observed likelihood of the data

$$\mathscr{L} = \log p_{\boldsymbol{\theta}}(\mathbf{x}_1, ..., \mathbf{x}_n) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i).$$

3. Take gradient with respect to $\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} \mathscr{L} = \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i).$$

4. Use (stochastic) gradients to maximize the likelihood of the data

Intuition: the model should place high probability on the data

Example: Coin Flips

Let $x_1, \ldots, x_n$ be binary observations

Use a Bernoulli model to fit $p = \sigma(\theta)$

$$\mathscr{L} = \sum_{i=1}^{n} \log p_\theta(x_i) = \sum_{i=1}^{n} x_i \theta - \log(1 + \exp(\theta))$$

Differentiate with respect to θ

$$\nabla_\theta \mathscr{L} = -n\sigma(\theta) + \sum_{i=1}^{n} x_i$$

Setting equal to zero

$$\sigma(\theta) = \frac{1}{n} \sum_{i=1}^{n} x_i$$

The maximum likelihood p equals the empirical frequency

Is Maximum Likelihood Good?

Define the empirical distribution

$$\hat{F}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \delta_{\mathbf{x}_i}(\mathbf{x})$$

The maximum likelihood objective $\mathscr{L}$ is

$$\begin{aligned}
\mathscr{L} &= \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \\
&= n\mathbb{E}_{\hat{F}}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i)] \\
&= n\mathbb{E}_{\hat{F}}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) - \log \hat{F}(\mathbf{x}_i)] + C \\
&= -nKL(\hat{F}(\mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{x})) + C
\end{aligned}$$

Maximum likelihood minimizes the KL-divergence between the empirical distribution on the data to the model

How to use maximum likelihood to fit observed data?

- We need a flexible distribution over the multivariate observations $\mathbf{x}$
- Hard for audio, text, images
- Can use known distributions like multivariate normal
- Known distributions places strong assumptions on the data

Idea 1: Decompose $p(\mathbf{x})$

Use the chain rule of probability. Let $\mathbf{x}$ be a 3-dimensional vector

$$p(\mathbf{x}) = p(\mathbf{x}_1)p(\mathbf{x}_2 \mid \mathbf{x}_1)p(\mathbf{x}_3 \mid \mathbf{x}_1, \mathbf{x}_2)$$

No assumptions were made

For a general k-dimensional vector

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \prod_{i=1}^{k} p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{x}_{<i})$$

Use univariate models to build a flexible multivariate model

Problem 1: Decompose p(x)

The challenge: dependencies are growing.

Consider $\mathbf{x}_3$ for $\mathbf{x}$ binary

$$p_{\boldsymbol{\theta}}(\mathbf{x}_3 \,|\, \mathbf{x}_2, \mathbf{x}_1) = \boldsymbol{\theta}[\mathbf{x}_2, \mathbf{x}_1]$$

As the dependency set grows, the conditional distribution parameter size grows exponentially.

Use deep learning to share parameters

Recurrent Neural Networks

Recurrent neural networks are sequence models

$$h_i = f_{\boldsymbol{\theta}}(h_{i-1}, \mathbf{x}_{i-1})$$

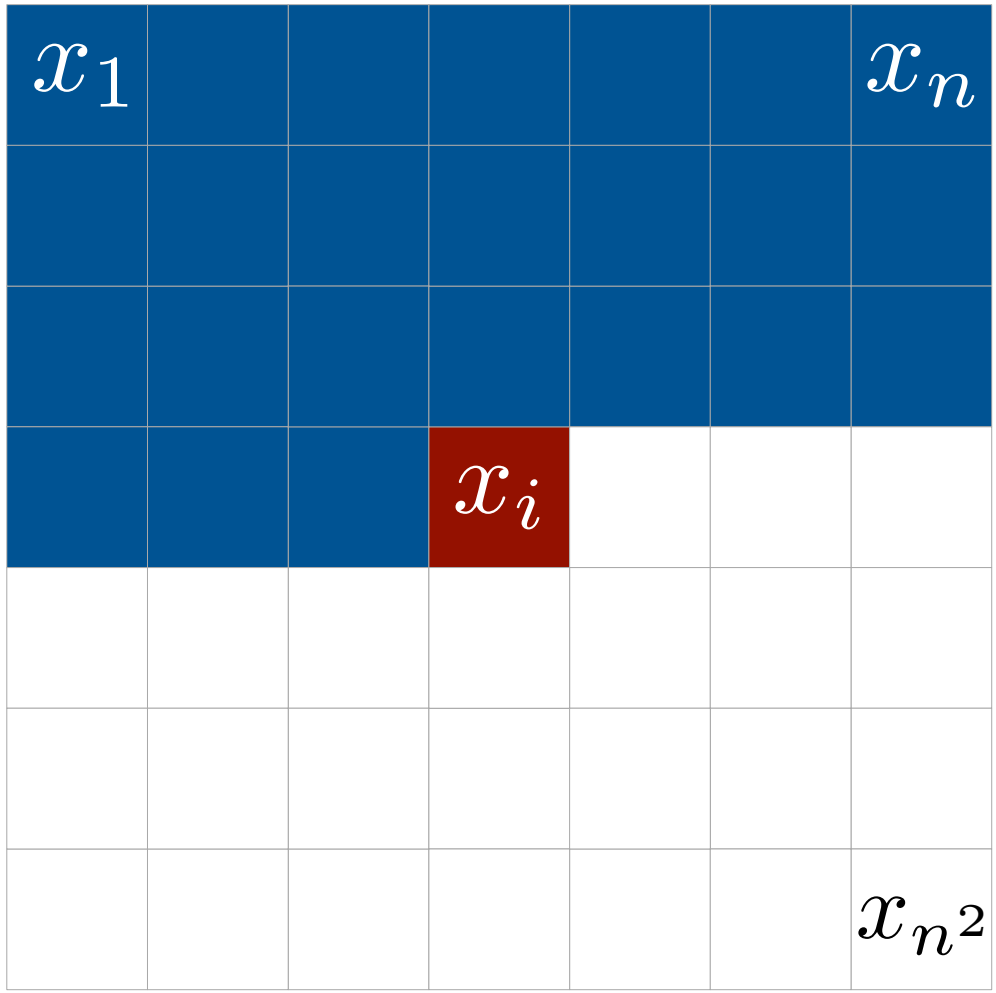
The hidden state h_i has information about all the previous $\mathbf{x}_{<i}$.

Use this to build a probability model over $\mathbf{x}$

$$\begin{aligned} h_i &= f_{\boldsymbol{\theta}}(h_{i-1}, \mathbf{x}_{i-1}) \\ p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{x}_{<i}) &= p(\mathbf{x}_i \mid h_i) \end{aligned}$$

Shares parameters among the univariate density problems

Still a Problem?



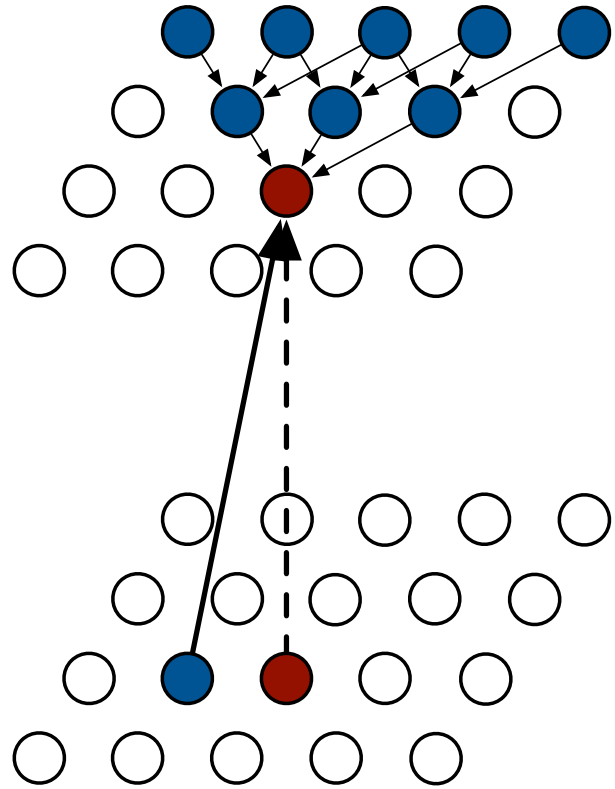
Long range dependencies are hard. Related pixels are far!

[van den Oord+ 2016]

Row LSTM

Operate on rows instead of pixels

$$
\begin{aligned}
[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] &= \sigma(\mathbf{K}^{ss} \circledast \mathbf{h}_{i-1} + \mathbf{K}^{is} \circledast \mathbf{x}_i) \\
\mathbf{c}_i &= \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i \\
\mathbf{h}_i &= \mathbf{o}_i \odot \tanh(\mathbf{c}_i)
\end{aligned}
$$

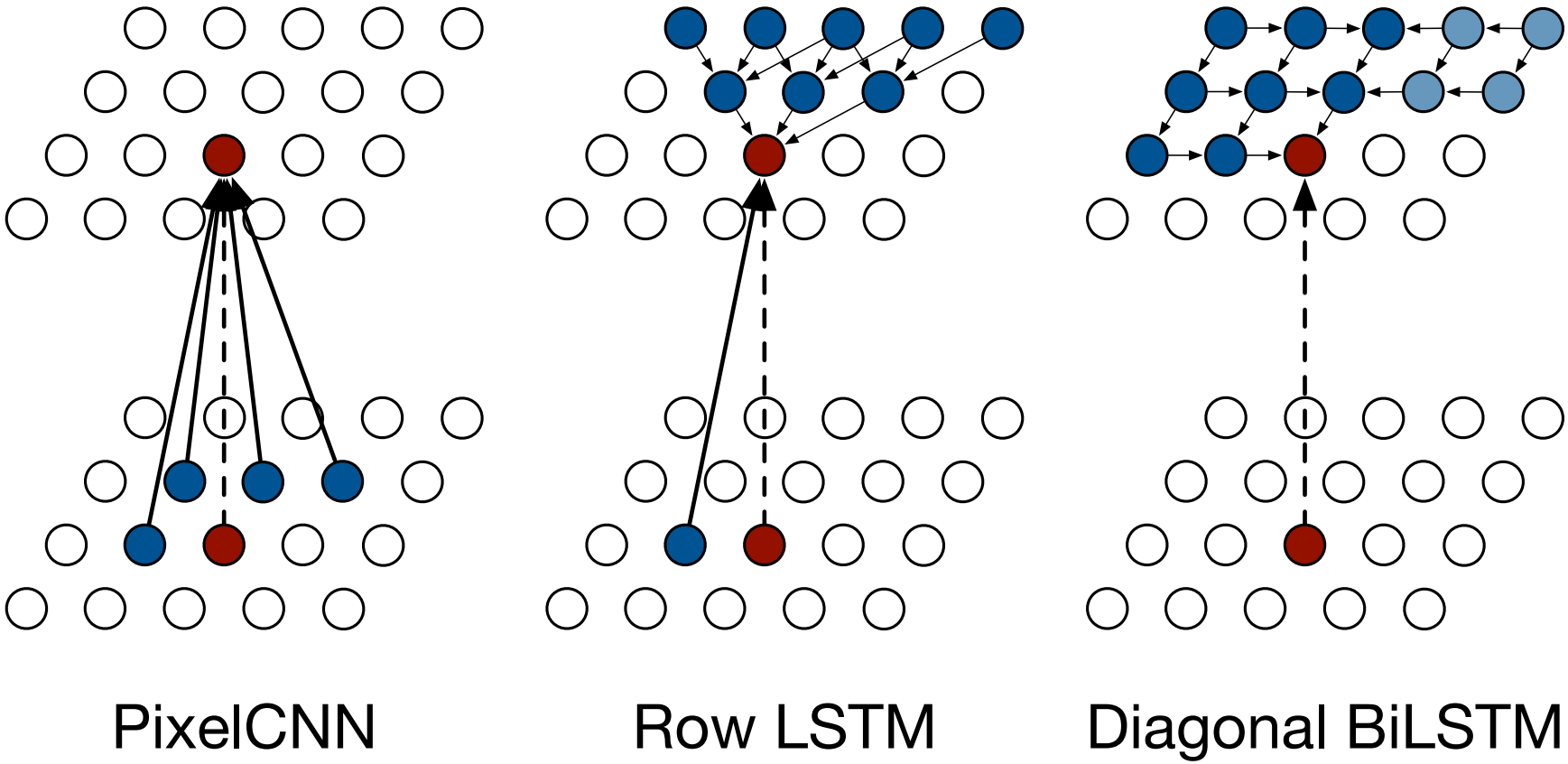


Row LSTM

Even more parameter sharing with convolution and stacking

[van den Oord+ 2016]

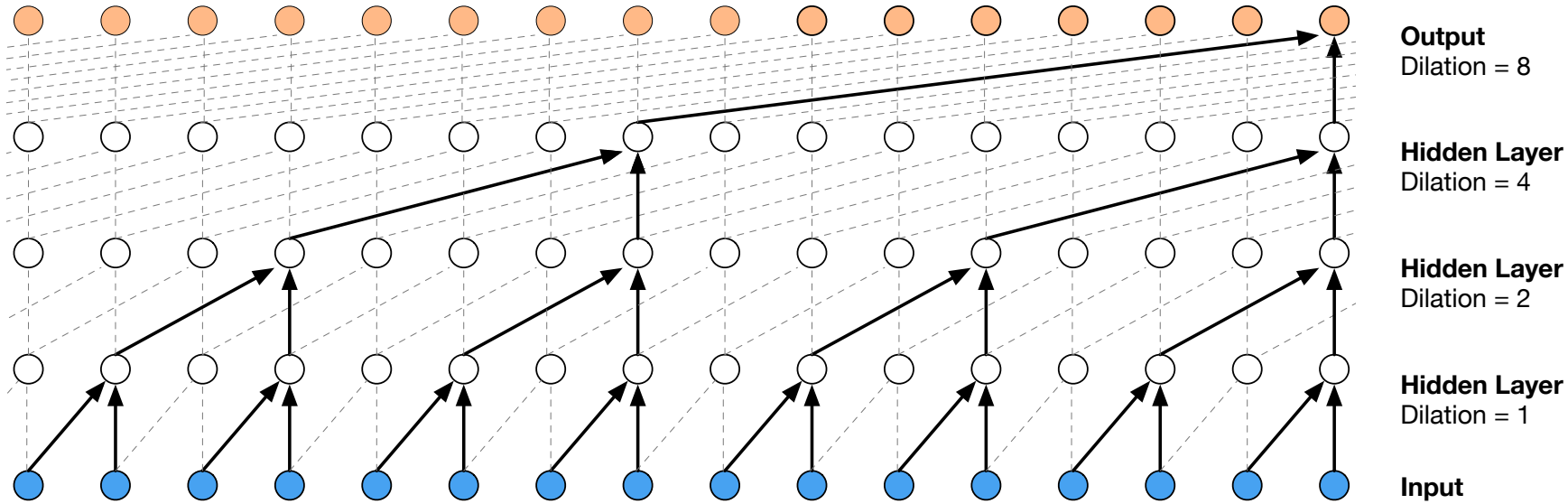
Different Architectures



Tradeoff receptive field size, computation, and flexibility

Different Architectures

Get dependencies by convolution with spacing



- Exponential spacing means exponentially long context
- Originally took 90 minutes to generate one second
- In 2017 reported a 1,000 faster approach. Now in Google Assistant

[van den Oord+ 2016]

Results

[van den Oord+ 2016]

Idea 2: Transform Noise Vectors

Directly get k dimensional densities by transforming noise

$$\boldsymbol{\epsilon} \sim s(\boldsymbol{\epsilon})$$
$$\mathbf{x} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$$

What's the density of $\mathbf{x}$? Assume f is invertible

$$p(\mathbf{x}) = s(f_{\boldsymbol{\theta}}^{-1}(\mathbf{x})) \left| \frac{df_{\boldsymbol{\theta}}^{-1}(\mathbf{x})}{d\mathbf{x}} \right|$$

Easy to write!

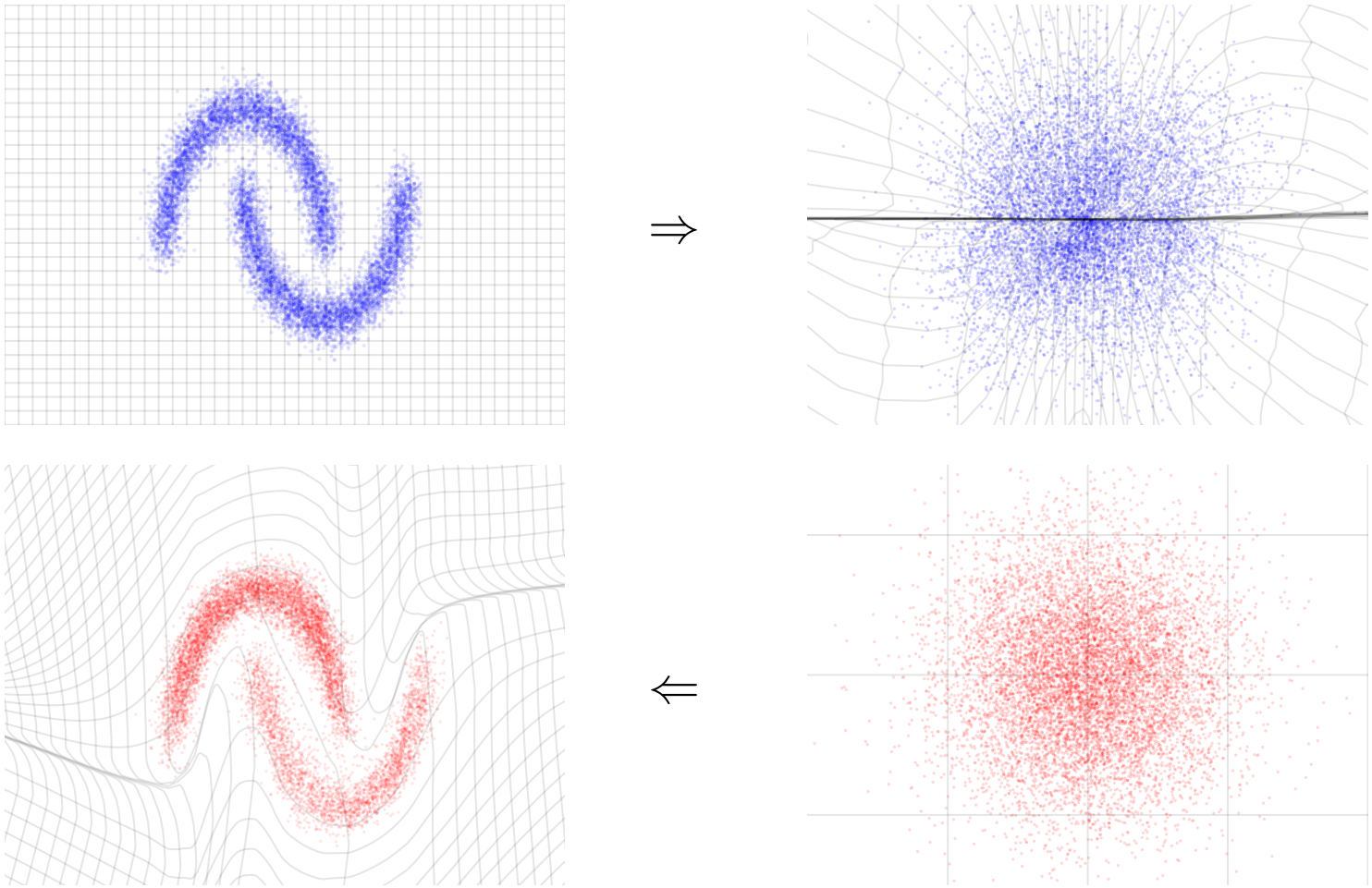
Problem 2: Jacobians are slow

Density of $\mathbf{x}$

$$p(\mathbf{x}) = s(f_{\boldsymbol{\theta}}^{-1}(\mathbf{x})) \left| \frac{df_{\boldsymbol{\theta}}^{-1}(\mathbf{x})}{d\mathbf{x}} \right|$$

- Requires computing the determinant of a $k \times k$ matrix
- Determinant computation in general is cubic
- Sampling and density requires both f and its inverse to be easy to compute

$$h(x) = x^x, \quad x > 1$$



[Dinh+ 2017]

Solution 2: Use functions O(n) Jacobians

Lower triangular matrix determinant can be computed in O(n)

$$|A| = \prod_{i=1}^{k} |a_{ii}|,$$

How do you build a function with lower triangular Jacobian? Limit the dependencies between dimensions

$$\begin{aligned}
\mathbf{x}_1 &= f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}_1) \\
\mathbf{x}_2 &= f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2) \\
\mathbf{x}_3 &= f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}_3)
\end{aligned}$$

We have

$$\frac{d\mathbf{x}_1}{d\boldsymbol{\epsilon}_{>1}} = 0$$

Example: Real NVP

Transformation:

$$\mathbf{x}_{1:d} = \boldsymbol{\epsilon}_{1:d}$$
$$\mathbf{x}_{d+1:k} = \boldsymbol{\epsilon}_{d+1:k} \cdot \exp(s(\boldsymbol{\epsilon}_{1:d})) + t(\boldsymbol{\epsilon}_{1:d})$$

Jacobian lower triangular because $\mathbf{x}_d$ depends only on $\boldsymbol{\epsilon}_{\leq d}$
Can compose functions because

$$|J[f \circ g]| = |Jf||Jg|$$

Tradeoff in choosing d:

- Large d means transformation functions s and t depend on large part of the vector
- Small d means more variables get transformed

[Dinh+ 2017]

Example: Real NVP

Transformation:

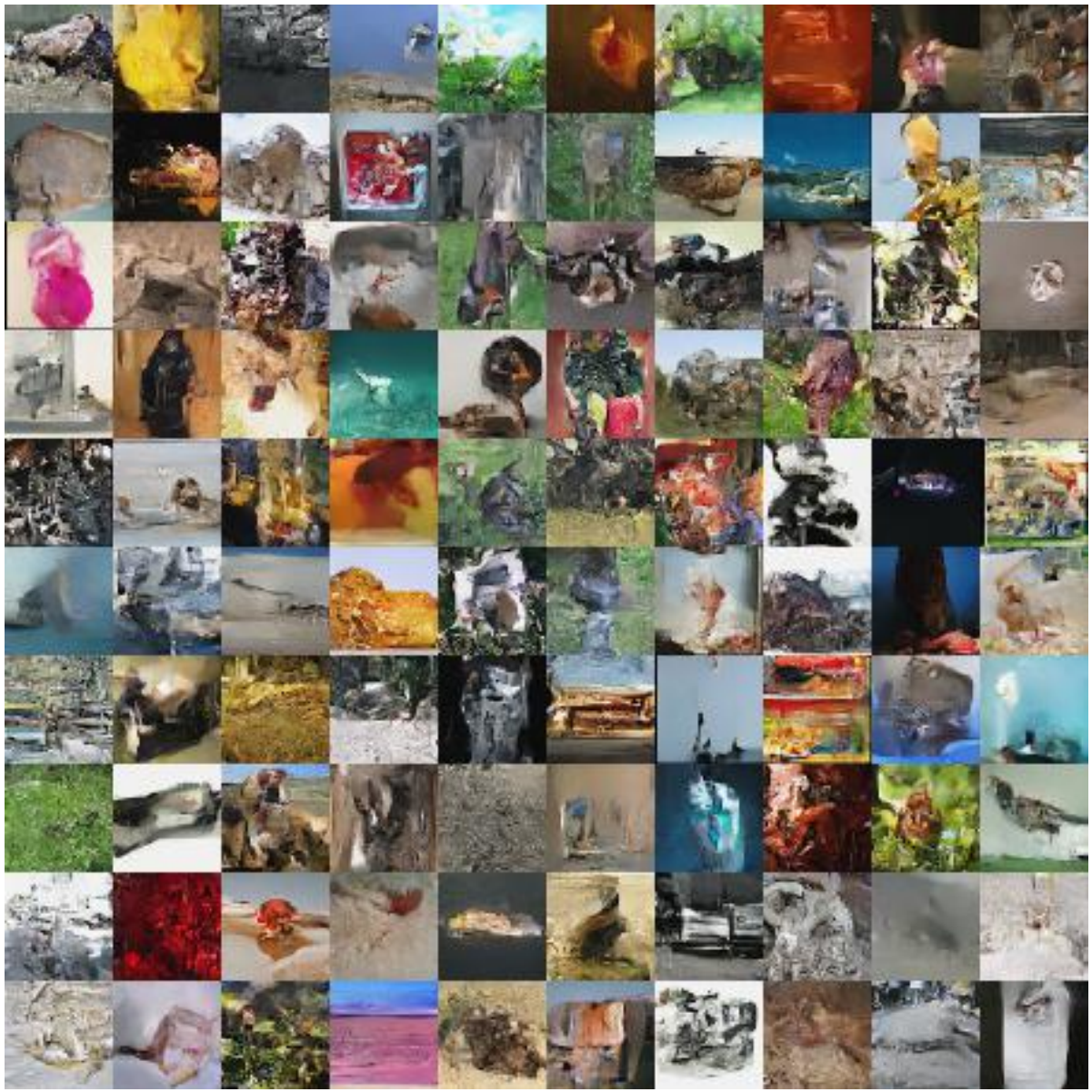
$$\mathbf{x}_{1:d} = \boldsymbol{\epsilon}_{1:d}$$
$$\mathbf{x}_{d+1:k} = \boldsymbol{\epsilon}_{d+1:k} \cdot \exp(s(\boldsymbol{\epsilon}_{1:d})) + t(\boldsymbol{\epsilon}_{1:d})$$

Jacobian lower triangular because $\mathbf{x}_d$ depends only on $\boldsymbol{\epsilon}_{\leq d}$

- Functions can use convolution or other prior knowledge
- The d can change so all dimensions can effect other dimensions with enough composition

[Dinh+ 2017]

Real NVP in Action



[Dinh+ 2017]

A Connection

Consider the autoregressive case where $\mathbf{x}$ is three dimensional

$$p(\mathbf{x}) = p_{\theta}(\mathbf{x}_1)p_{\theta}(\mathbf{x}_2 \mid \mathbf{x}_1)p_{\theta}(\mathbf{x}_3 \mid \mathbf{x}_1, \mathbf{x}_2)$$

Implies for some noise $\boldsymbol{\epsilon}$

$$\mathbf{x}_1 = f_{\theta}(\boldsymbol{\epsilon}_1), \quad \mathbf{x}_2 = f_{\theta}(\boldsymbol{\epsilon}_2, \mathbf{x}_1), \quad \mathbf{x}_3 = f_{\theta}(\boldsymbol{\epsilon}_3, \mathbf{x}_1, \mathbf{x}_2)$$

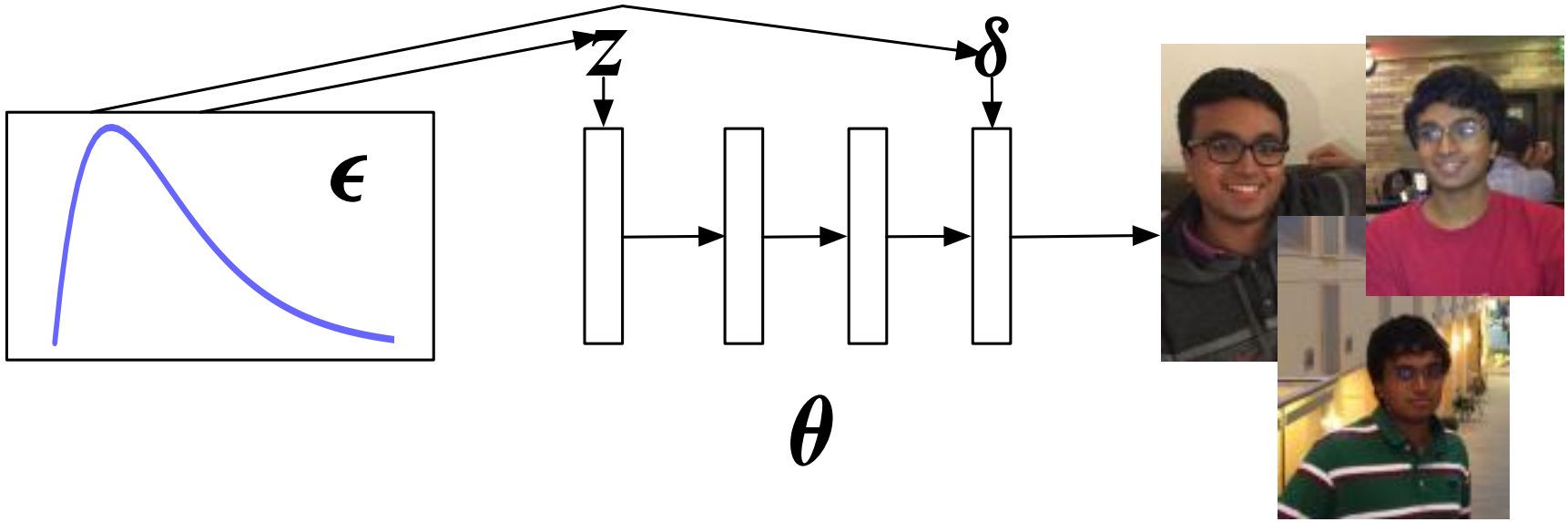
Substituting definitions of $\mathbf{x}_1$ and $\mathbf{x}_2$

$$\begin{aligned}
\mathbf{x}_1 &= f_{\theta}(\boldsymbol{\epsilon}_1) \\
\mathbf{x}_2 &= f_{\theta}(\boldsymbol{\epsilon}_2, f_{\theta}(\boldsymbol{\epsilon}_1)) \\
\mathbf{x}_3 &= f_{\theta}(\boldsymbol{\epsilon}_3, f_{\theta}(\boldsymbol{\epsilon}_1), f_{\theta}(\boldsymbol{\epsilon}_2, f_{\theta}(\boldsymbol{\epsilon}_1)))
\end{aligned}$$

This gives a vector transform with lower triangular Jacobian. The two approaches are almost the same!

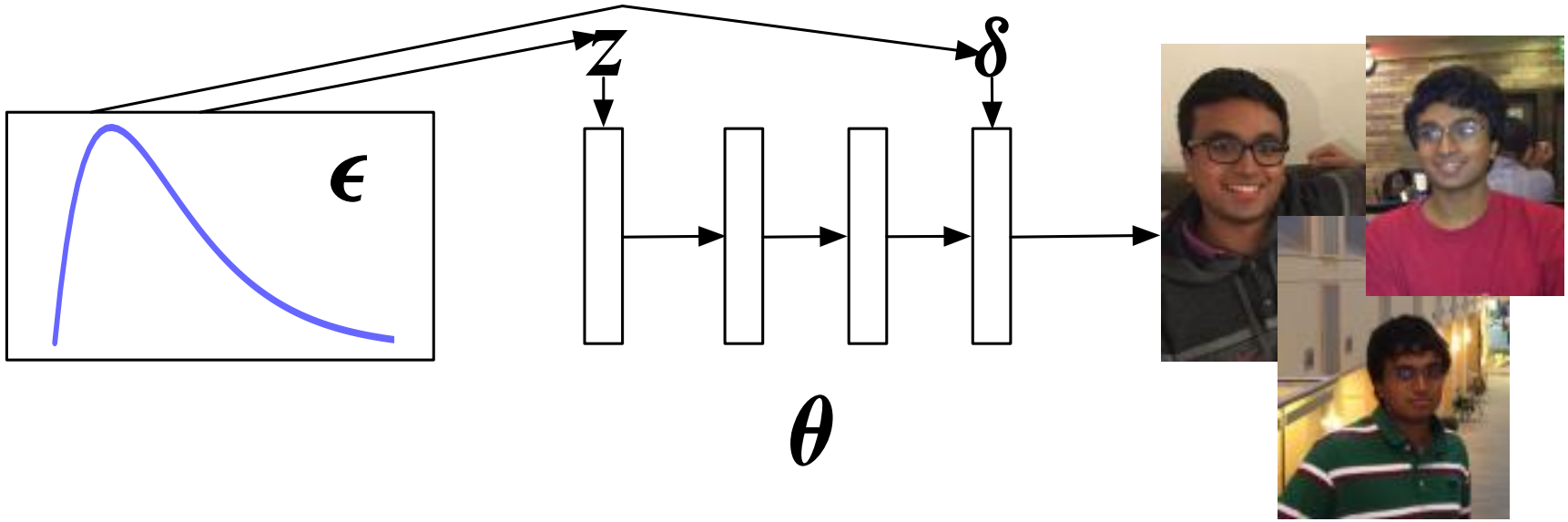
Generative Adversarial Networks

Deep generative model with latent variables



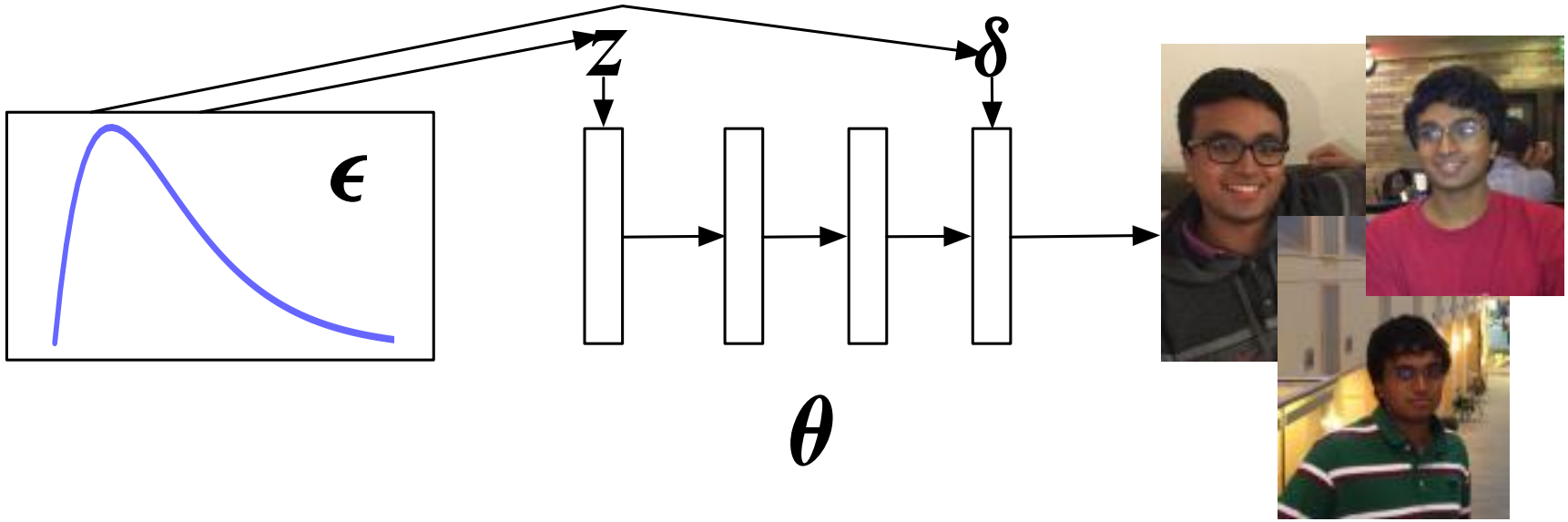
Build deep generative model using layers of hidden variables **z**

Deep generative model with latent variables



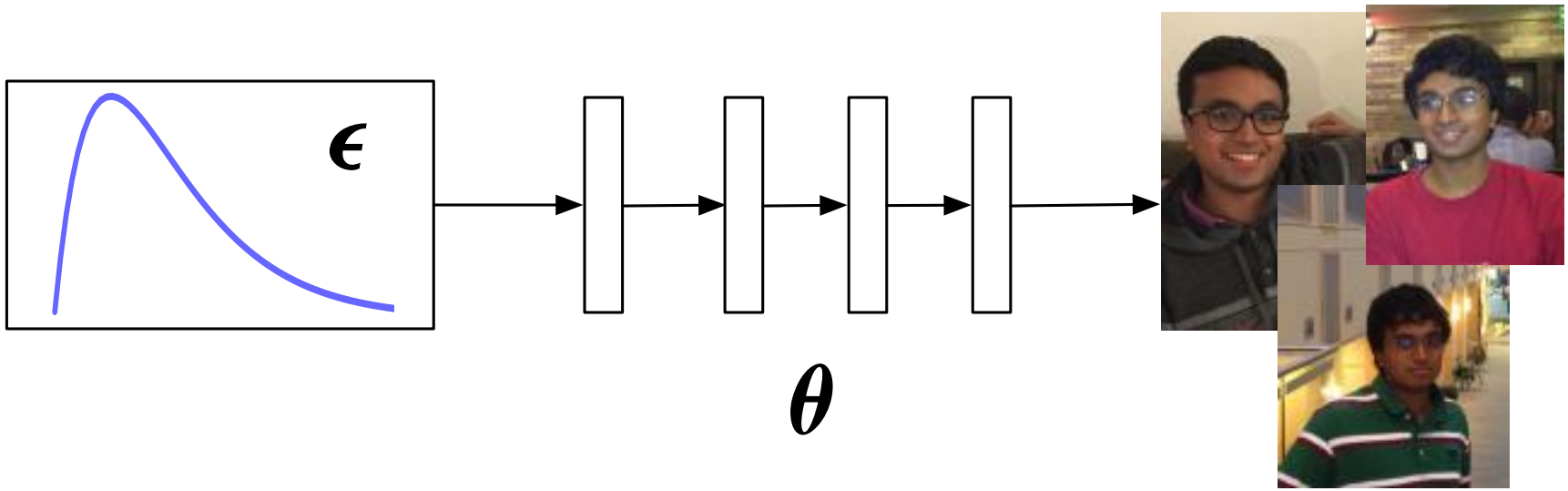
- Hidden variables have prior $p(\mathbf{z})$
- Data conditions on the hidden variables $p(\mathbf{x}|\mathbf{z})$

Deep generative model with latent variables



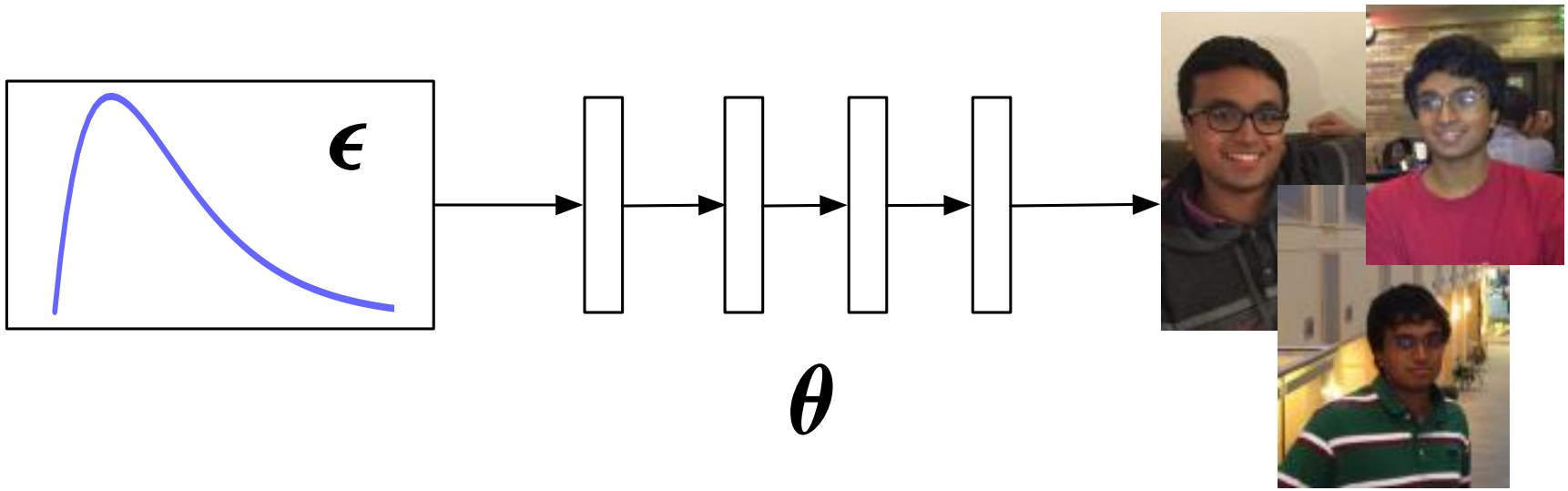
- Challenge is in accurate inference of **z** given **x**
- Inference is needed because amount of noise is larger than data

Deep generative model with Noise



- Try to model density directly
- $\mathbf{x} \sim p_{\theta}(\mathbf{x}) \iff \mathbf{x} = f(\boldsymbol{\theta}, \boldsymbol{\epsilon})$

Deep generative model with Noise



- $\mathbf{x} = f(\boldsymbol{\theta}, \boldsymbol{\epsilon})$
- Density might not exist

Adversarial Learning

How do we learn a model $f_{\boldsymbol{\theta}}$ given samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$?

1. Write down a model $f(\boldsymbol{\theta}, \boldsymbol{\epsilon})$
2. Write down a discriminator $D_{\boldsymbol{\phi}}$ between
   Real data $\mathbf{x}_i$
   Model samples $\mathbf{x}_i^*$
3. Classification objective for discriminator

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta}, \boldsymbol{\epsilon})}[\log(1 - D(\mathbf{x}))]$$

4. Use stochastic gradients to maximize with respect to $\boldsymbol{\phi}$
5. Use stochastic gradients to minimize with respect to $\boldsymbol{\theta}$

Make it hard to distinguish between model and real samples

Adversarial Learning

Objective:

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})}[\log(1-D(\mathbf{x}))]$$

Goal:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})}[\log(1-D(\mathbf{x}))]$$

Gradients:

$$\nabla_{\boldsymbol{\phi}} \mathscr{L} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\nabla_{\boldsymbol{\phi}} \log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} \log(1-D(\mathbf{x}))]$$
$$\nabla_{\boldsymbol{\theta}} \mathscr{L} = \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})}[\nabla_{\mathbf{x}} \log(1-D(\mathbf{x}))\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta},\boldsymbol{\epsilon})]$$

- Two player game. Do gradient methods make sense?
- Differentiable observations?

Optimal Discriminator

Loss for the discriminator

$$\max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})}[\log(1-D(\mathbf{x}))]$$

p_{model} defined implicitly at preimage

$$p_{\text{model}}(\mathbf{x}) = \int_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})} s(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$$

Differentiate $\mathscr{L}$ functionally

$$\nabla_D \mathscr{L} = \frac{p_{\text{data}}(\mathbf{x})}{D(\mathbf{x})} - \frac{p_{\text{model}}(\mathbf{x})}{1-D(\mathbf{x})}$$

Maximized at

$$D(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}$$

Objective At Optimal Discriminator

Recall

$$\mathcal{L}(M, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta}, \boldsymbol{\epsilon})}[\log(1 - D(\mathbf{x}))]$$

Substitute optimal discriminator

$$\mathcal{L}(M) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim p_{\text{model}}} \left[\log \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})} \right]$$

$$\mathcal{L}(M) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log \frac{\frac{1}{2} p_{\text{data}}(\mathbf{x})}{\frac{1}{2} p_{\text{data}}(\mathbf{x}) + \frac{1}{2} p_{\text{model}}(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim p_{\text{model}}} \left[\log \frac{\frac{1}{2} p_{\text{model}}(\mathbf{x})}{\frac{1}{2} p_{\text{data}}(\mathbf{x}) + \frac{1}{2} p_{\text{model}}(\mathbf{x})} \right]$$

Objective At Optimal Discriminator

From the previous slide

$$\mathscr{L}(M) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{\frac{1}{2}p_{\text{data}}(\mathbf{x})}{\frac{1}{2}p_{\text{data}}(\mathbf{x}) + \frac{1}{2}p_{\text{model}}(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_{\text{model}}}\left[\log \frac{\frac{1}{2}p_{\text{model}}(\mathbf{x})}{\frac{1}{2}p_{\text{data}}(\mathbf{x}) + \frac{1}{2}p_{\text{model}}(\mathbf{x})}\right]$$

Pull out the constant

$$\mathscr{L}(M) = -\log 4 + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{1}{2}p_{\text{data}}(\mathbf{x}) + \frac{1}{2}p_{\text{model}}(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_{\text{model}}}\left[\log \frac{p_{\text{model}}(\mathbf{x})}{\frac{1}{2}p_{\text{data}}(\mathbf{x}) + \frac{1}{2}p_{\text{model}}(\mathbf{x})}\right]$$

Objective At Optimal Discriminator

From the previous slide

$$\mathscr{L}(M) = -\log 4 + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{1}{2}p_{\text{data}}(\mathbf{x}) + \frac{1}{2}p_{\text{model}}(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_{\text{model}}}\left[\log \frac{p_{\text{model}}(\mathbf{x})}{\frac{1}{2}p_{\text{data}}(\mathbf{x}) + \frac{1}{2}p_{\text{model}}(\mathbf{x})}\right]$$

Convert to probability "distance"

$$\begin{aligned}\mathscr{L}(M) &= -\log 4 + \text{KL}\left(p_{\text{data}} || \frac{1}{2}p_{\text{data}} + \frac{1}{2}p_{\text{model}}\right) \\ &\quad + \text{KL}\left(p_{\text{model}} || \frac{1}{2}p_{\text{data}} + \frac{1}{2}p_{\text{model}}\right) \\ &= -\log 4 + 2\text{JSD}(p_{\text{model}} || p_{\text{data}})\end{aligned}$$

This is the Jenson Shannon Divergence

Alternative View: Likelihood Ratio Estimation

Was there something special about the original objective?

- Consider pseudolabels y
- Assign pseudolabel $y = 1$ to $\mathbf{x} \sim p_{\text{data}}$
- Assign pseudolabel $y = 0$ to $\mathbf{x} \sim p_{\text{model}}$

$$\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x})} = \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} = \frac{p(y=1|\mathbf{x})p(\mathbf{x})p(y=1)^{-1}}{p(y=0|\mathbf{x})p(\mathbf{x})p(y=0)^{-1}}$$
$$= \frac{p(y=1|\mathbf{x})p(y=1)^{-1}}{p(y=0|\mathbf{x})p(y=0)^{-1}}$$

Assume equal samples from data and model

$$\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{1-p(y=1|\mathbf{x})}$$

Alternative View: Likelihood Ratio Estimation

From the previous

$$\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x})} = \frac{p(y=1\,|\,\mathbf{x})}{p(y=0\,|\,\mathbf{x})} = \frac{p(y=1\,|\,\mathbf{x})}{1-p(y=1\,|\,\mathbf{x})}$$

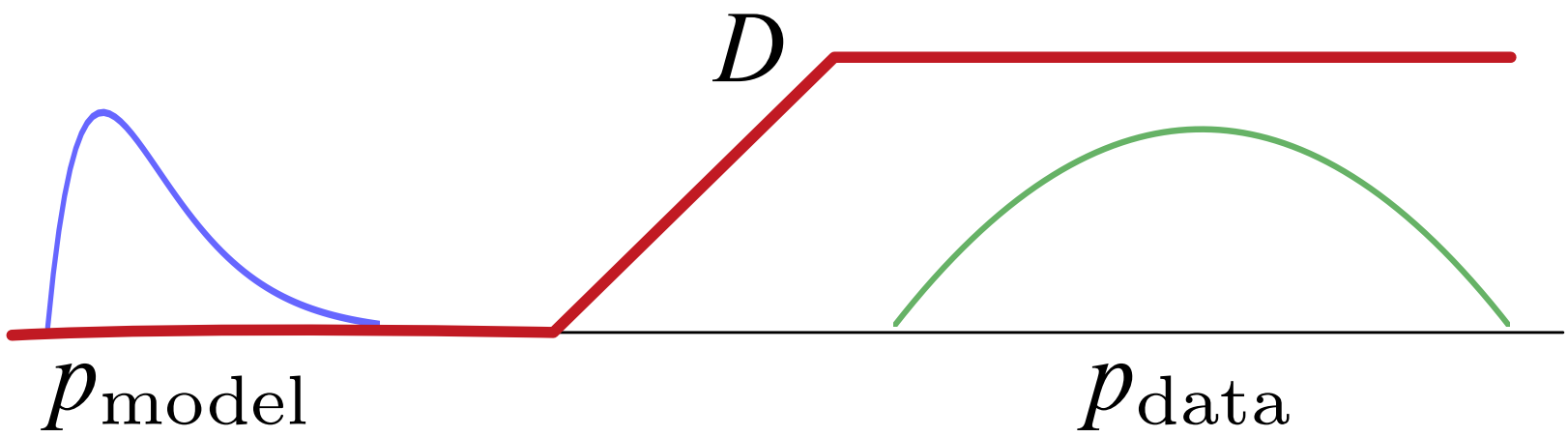
Solve for $p(y=1\,|\,\mathbf{x})$

$$p(y=1\,|\,\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}$$

- Conditional distribution has form of optimal discriminator
- Learn conditionals with classification
- Any classification method that is “proper” recovers this!

A Problem?

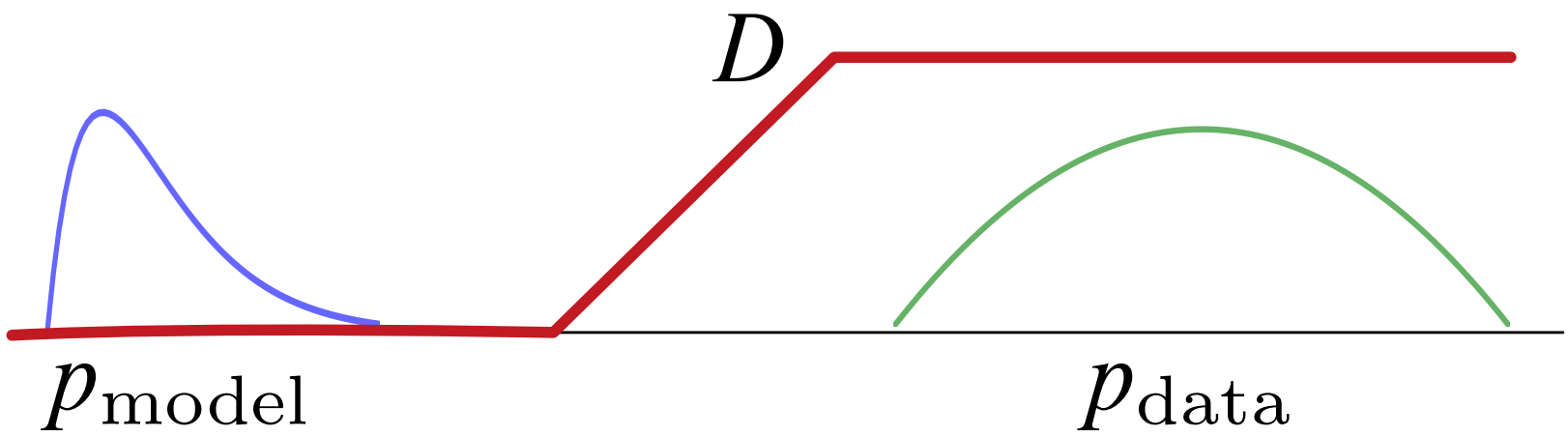
Consider two distribution whose supports don't overlap?



Does something bad happen?

A Problem?

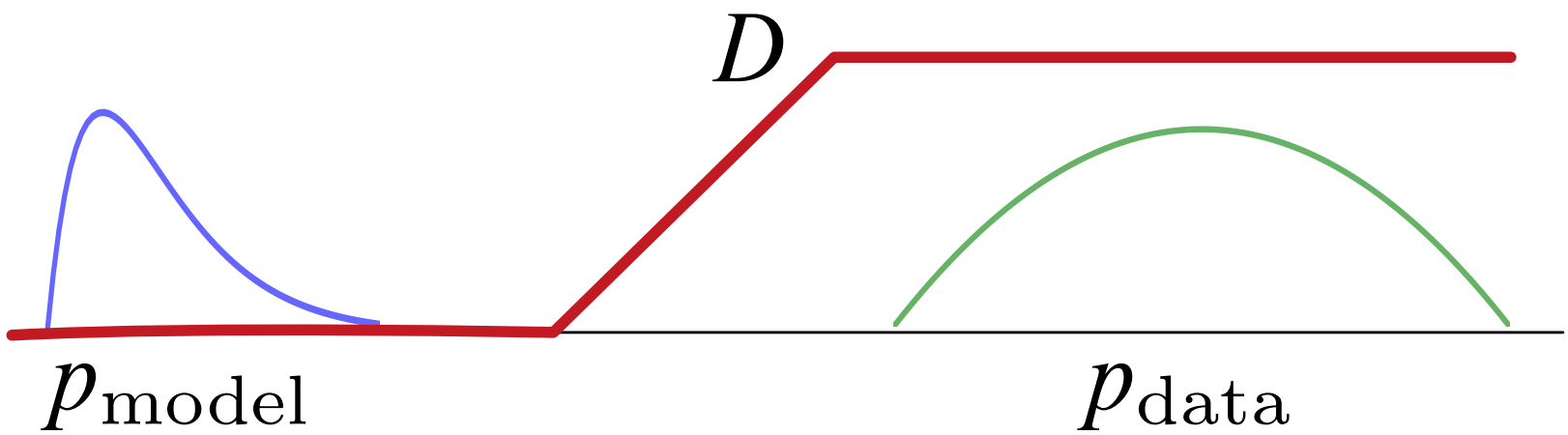
Consider two distribution whose supports don't overlap?



- Discriminator is optimal
- Gradient of discriminator is zero at non-zero probability

A Problem?

Consider two distribution whose supports don't overlap?



Recall

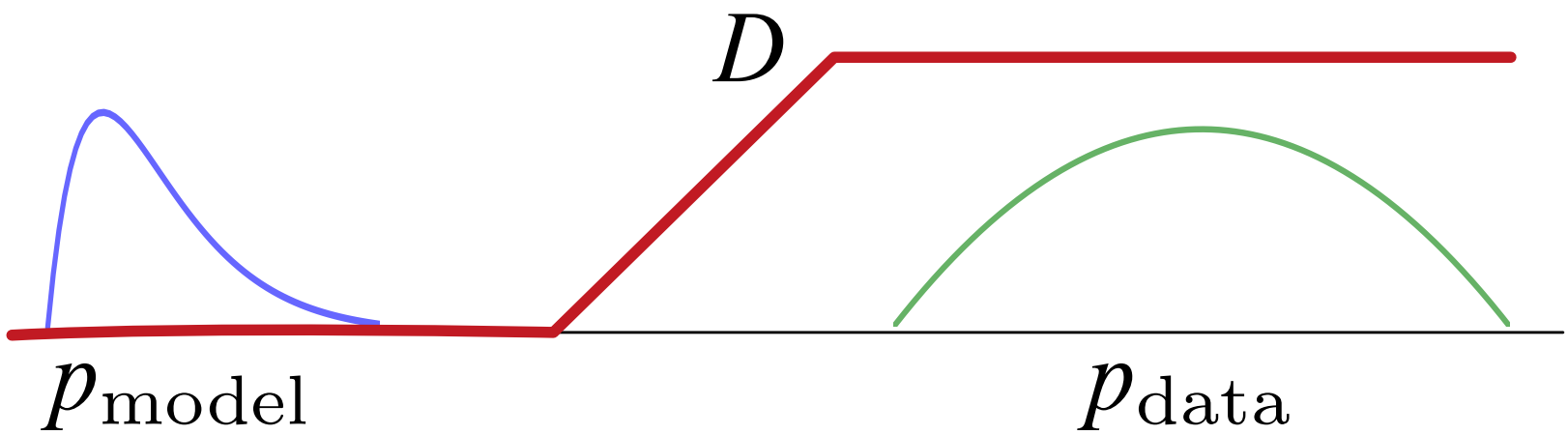
$$\mathcal{L}(M,D) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})}[\log(1-D(\mathbf{x}))]$$

then,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L} &= \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})}[\nabla_{\mathbf{x}}\log(1-D(\mathbf{x}))\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta},\boldsymbol{\epsilon})] \\ &= \mathbb{E}_{\mathbf{x}=f(\boldsymbol{\theta},\boldsymbol{\epsilon})}\left[\frac{\nabla_{\mathbf{x}}D(\mathbf{x})}{1-D(\mathbf{x})}\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta},\boldsymbol{\epsilon})\right] \\ &= 0\end{aligned}$$

A Problem?

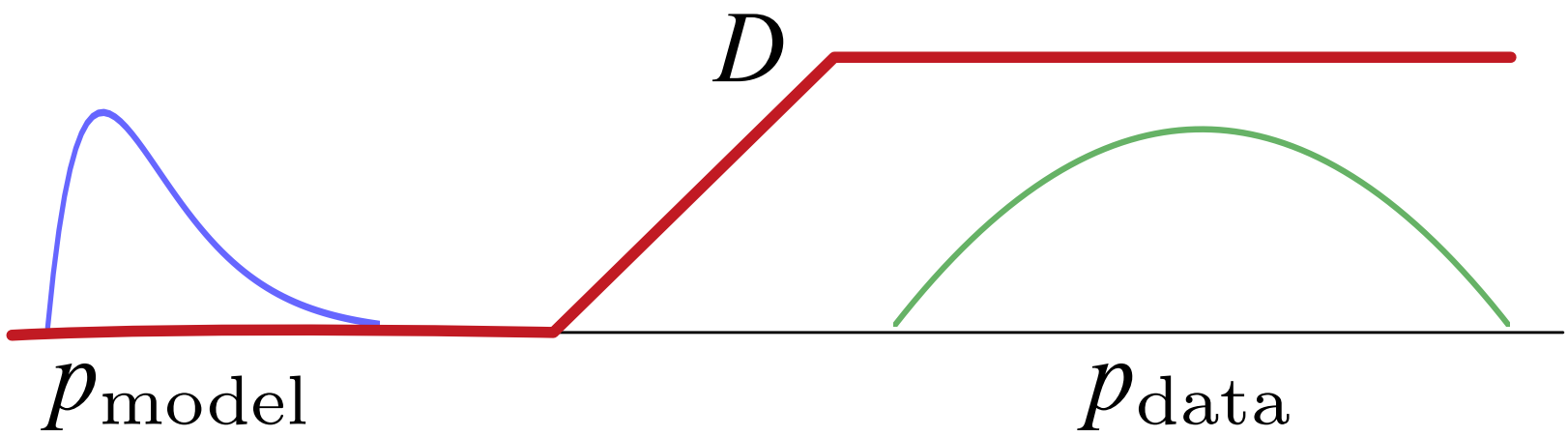
Consider two distribution whose supports don't overlap?



- Is this a problem with real data?
- Low dim manifolds in high dim spaces likely to not overlap
- Can cost even be discontinuous in this case?

A Problem?

Consider two distribution whose supports don't overlap?



- Why does it work?
- A suboptimal discriminator smooths distributions
- Is this a good idea?

Recall: Maximum Likelihood

Define the empirical distribution

$$\hat{F}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \delta_{\mathbf{x}_i}(\mathbf{x})$$

The the maximum likelihood objective $\mathscr{L}$ is

$$\begin{aligned}
\mathscr{L} &= \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \\
&= n\mathbb{E}_{\hat{F}}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i)] \\
&= n\mathbb{E}_{\hat{F}}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) - \log \hat{F}(\mathbf{x}_i)] + C \\
&= -n\mathrm{KL}(\hat{F}(\mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{x})) + C
\end{aligned}$$

Maximum likelihood minimizes the KL-divergence between the empirical distribution on the data to the model

Comparison with Maximum Likelihood

The objective for negative maximum likelihood

$$\mathscr{L}_{\text{neg-ml}} = n\text{KL}(\hat{F}(\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{x})) + C$$

The objective for a vanilla GAN

$$\mathscr{L}_{\text{gan}} = \text{JSD}(p_{\text{model}}||p_{\text{data}}) + C$$

Suppose $\text{Supp}(p_{\boldsymbol{\theta}}(\mathbf{x})) \subseteq \text{Supp}(\hat{F}(\mathbf{x}))$

- KL is infinite, JSD is not
- Difference lies between distances
- Trade off: sharpness for mode dropping
- Can go beyond KL with Wasserstein distances

Results

[Goodfellow+ 2014]

Results

[Kerras+ 2017]

The Evidence Lower Bound

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_q\left[\log p(\mathbf{x}, \mathbf{z})\right] - \mathbb{E}_q\left[\log q(\mathbf{z}; \boldsymbol{\lambda})\right]$$

- KL is intractable; VI optimizes the **evidence lower bound** (ELBO)
  - It is a lower bound on $\log p(\mathbf{x})$
  - Maximizing the ELBO is equivalent to minimizing the KL
- The ELBO trades off two terms
  - The first term prefers $q(\cdot)$ to place its mass on the MAP estimate
  - The second term encourages $q(\cdot)$ to be diffuse
- Approximation q is chosen to be in the mean field

The Evidence Lower Bound

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_q\left[\log p(\mathbf{x}\,|\,\mathbf{z})\right] - \mathrm{KL}(q(\mathbf{z};\boldsymbol{\lambda})||p(\mathbf{z}))$$

- KL is intractable; VI optimizes the **evidence lower bound** (ELBO)
  - It is a lower bound on $\log p(\mathbf{x})$
  - Maximizing the ELBO is equivalent to minimizing the KL
- The ELBO trades off two terms
  - The first term prefers $q(\cdot)$ to maximize the likelihood
  - The second term regularizes $q(\cdot)$ to the prior
- Approximation q is chosen to be in the mean field

Connecting to Variational Inference

ELBO with amortized inference

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_x[\mathbb{E}_q[\log p(\mathbf{x}\,|\,\mathbf{z})] - \mathrm{KL}(q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})||p(\mathbf{z}))]$$

Idea: Richer variational approximations

$$\begin{aligned}\mathbf{z} &\sim q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda}) \\ &\iff \mathbf{z} = f(\boldsymbol{\lambda},\boldsymbol{\epsilon},\mathbf{x})\end{aligned}$$

Challenge: $q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})$ no longer tractable

$$q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda}) = \int_{\mathbf{z}=f(\boldsymbol{\lambda},\boldsymbol{\epsilon},\mathbf{x})} s(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$$

Solution: KL divergence needs likelihood ratio. Use likelihood ratio estimation!

Connecting to Variational Inference

Recall

$$p(y=1\,|\,\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}$$

Move to $\mathbf{z}, \mathbf{x}$ space

$$p(y=1\,|\,\mathbf{x},\mathbf{z}) = \frac{q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})p(\mathbf{x})}{p(\mathbf{z})p(\mathbf{x}) + q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})p(\mathbf{x})} = \frac{1}{1+\frac{p(\mathbf{z})}{q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})}}$$

$$\rightarrow \log\left(\frac{p(y=1\,|\,\mathbf{x},\mathbf{z})}{1-p(y=1\,|\,\mathbf{x},\mathbf{z})}\right) = \log q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda}) - \log p(\mathbf{z})$$

This is exactly the intractable term in the ELBO

Can be estimated by classification estimate $r_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z})$

Connecting to Variational Inference

Approximate ELBO:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda}) &= \mathbb{E}_x[\mathbb{E}_q[\log p(\mathbf{x}\,|\,\mathbf{z})] - \mathrm{KL}(q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})||p(\mathbf{z}))] \\ &= \mathbb{E}_x[\mathbb{E}_q[\log p(\mathbf{x}\,|\,\mathbf{z})] - \mathbb{E}_q[r_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z})]]\end{aligned}$$

Algorithm:

- Sample from x, then sample $q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})$ and $p(\mathbf{z})$
- Use samples to fit (log) classifier $r_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z})$
- Maximize approximate ELBO with $r_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z})$ [Mescheder+ 2017]

Connecting to Variational Inference

Approximate ELBO:

$$\begin{aligned}\mathscr{L}(\boldsymbol{\lambda}) &= \mathbb{E}_x[\mathbb{E}_q[\log p(\mathbf{x}\,|\,\mathbf{z})] - \mathrm{KL}(q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})||p(\mathbf{z}))] \\ &= \mathbb{E}_x[\mathbb{E}_q[\log p(\mathbf{x}\,|\,\mathbf{z})] - \mathbb{E}_q[r_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z})]]\end{aligned}$$

Can get better models as well by allowing intractable $p(\mathbf{x}\,|\,\mathbf{z})$

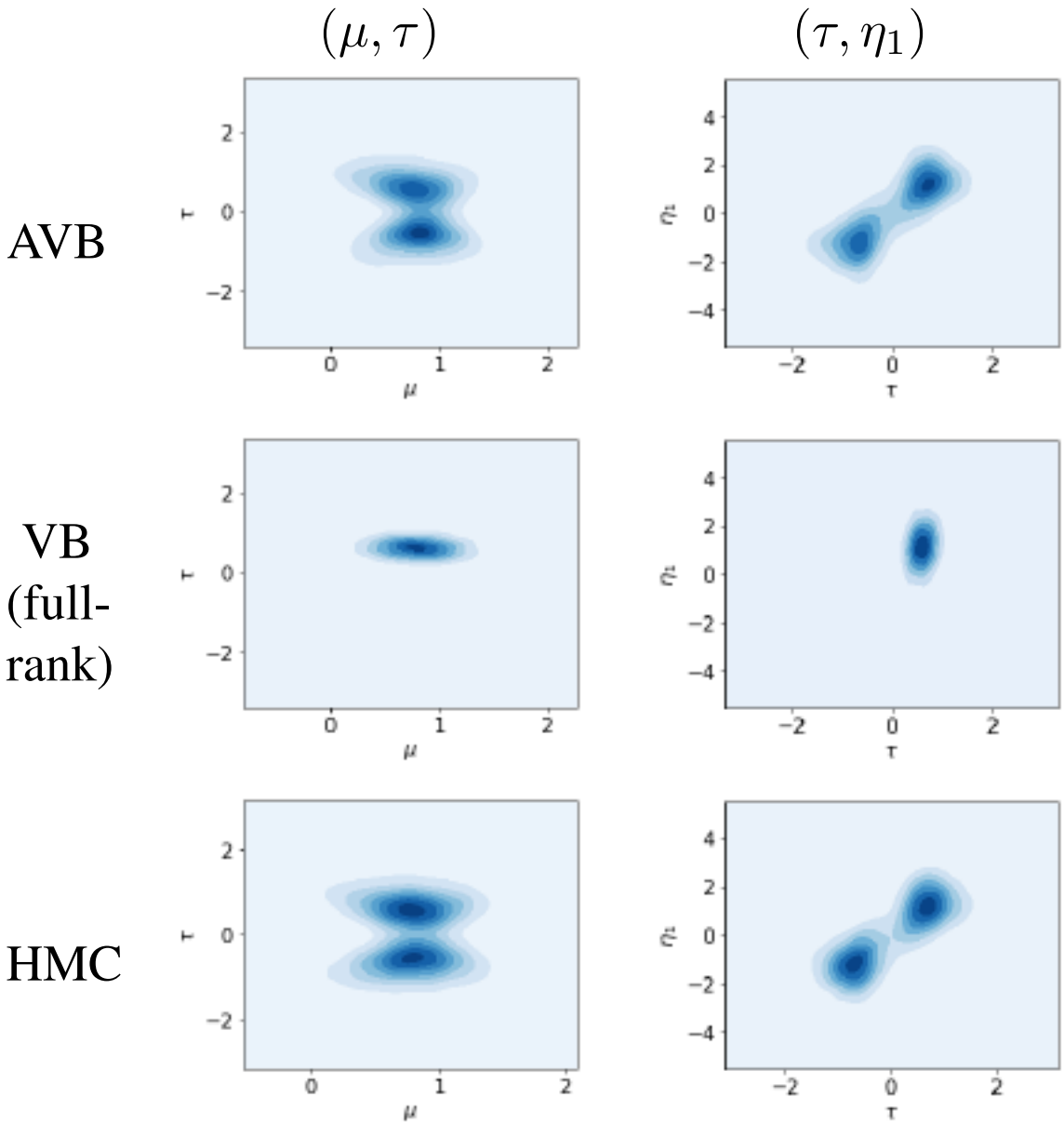
$$\begin{aligned}\mathscr{L}(\boldsymbol{\lambda}) &= \mathbb{E}_x[\mathbb{E}_q[\log p(\mathbf{x}\,|\,\mathbf{z})] - \mathrm{KL}(q(\mathbf{z}\,|\,\mathbf{x};\boldsymbol{\lambda})||p(\mathbf{z}))] \\ &= \mathbb{E}_x[\mathbb{E}_q[\log p(\mathbf{x}\,|\,\mathbf{z})p(\mathbf{z}) - \log q(\mathbf{z}\,|\,\mathbf{x})q(\mathbf{x})]] + C\end{aligned}$$

For any arbitrary $q(\mathbf{x})$ independent of $\boldsymbol{\lambda}$.

All parts of the model now no longer need tractable likelihoods

[Tran+ 2017]

Results



[Mescheder+ 2017]

Want to generate samples with some $p(\mathbf{x})$

Neural networks can transform unstructured noise to structured observations

Variational autoencoders do it with noise greater than the dimension of the data

Direct likelihood estimation does it with noise equal in dimension to the data

GANs do it with noise smaller than the dimension of the data

New fun generative models that use diffusions

How could we use deep generative models to create a new molecule to treat knee pain?