EE 628
Deep Learning
Spring 2020

Lecture 13

04/23/2020

Sergul Aydore

Applied Scientist

Amazon Web Services

Announcements

- Deadline for Projects **04/27/2020 Monday at 5pm ET**
  - project presentation on **04/30/2020** (40 %)
  - Details on grading: https://github.com/sergulaydore/EE-628-Spring-2020

Overview

- Last lecture we covered
  - Advanced Topics in Computer Vision

- Today, we will cover
  - Generative Models as Unsupervised Learning

- Source material:
  - Dive into Deep Learning (https://d2l.ai/)
  - https://github.com/sergulaydore/EE-628-Spring-2020
  - https://sites.google.com/view/berkeley-cs294-158-sp20/home

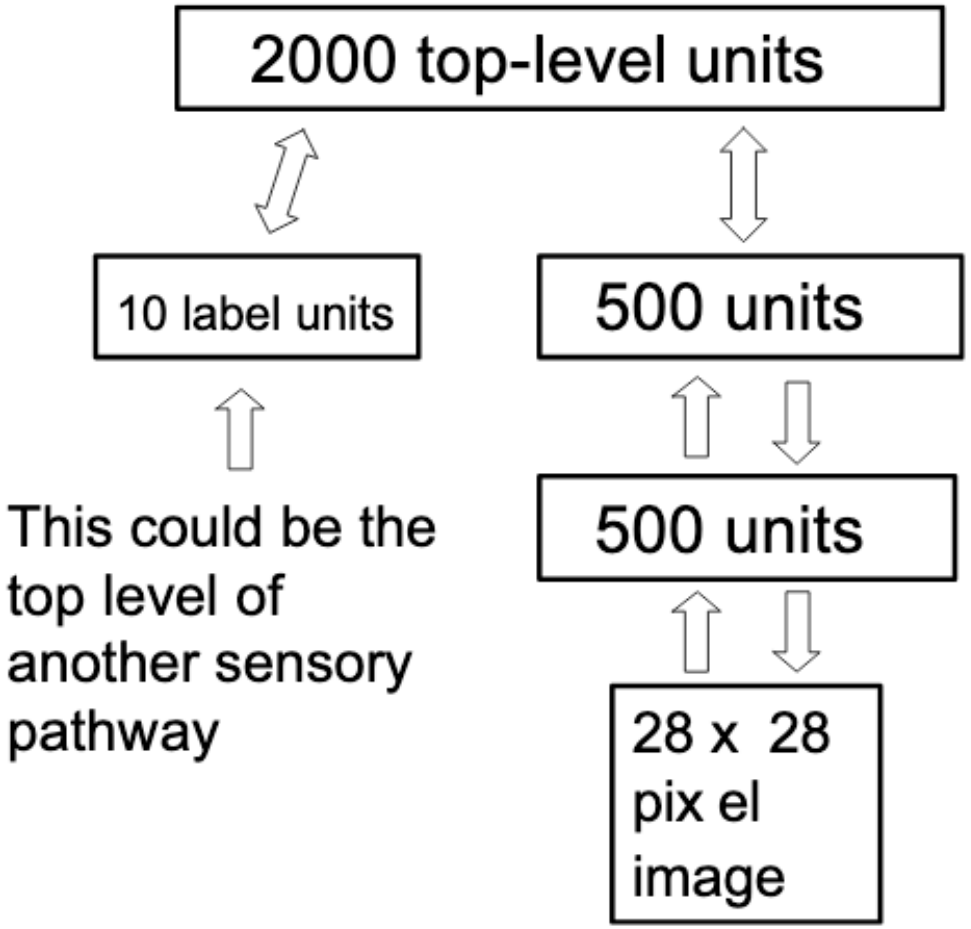
Unsupervised Learning

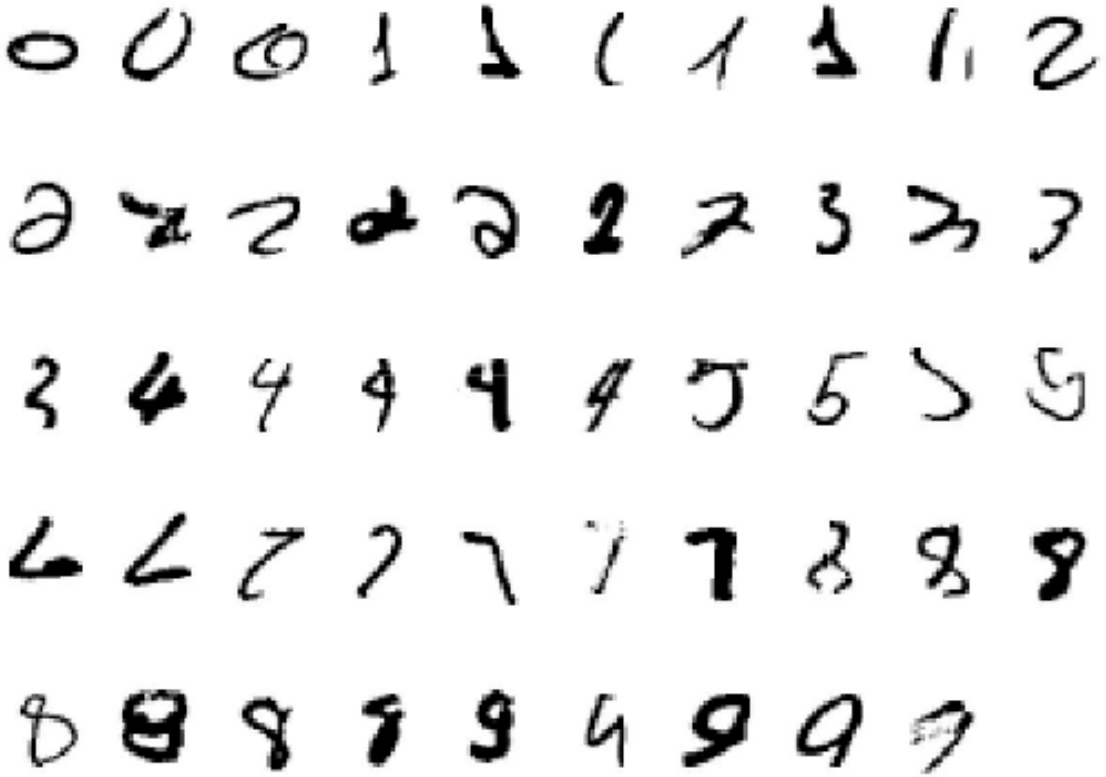
- Capturing rich patterns in raw data in a label-free way
  - Generative Models: recreate raw data distribution
  - Self-supervised Learning: “puzzle” tasks that require semantic understanding

Applications for Unsupervised Learning

- Clustering
- Visualization
- Dimensionality reduction
- Anomaly detection
- Generating synthetic data

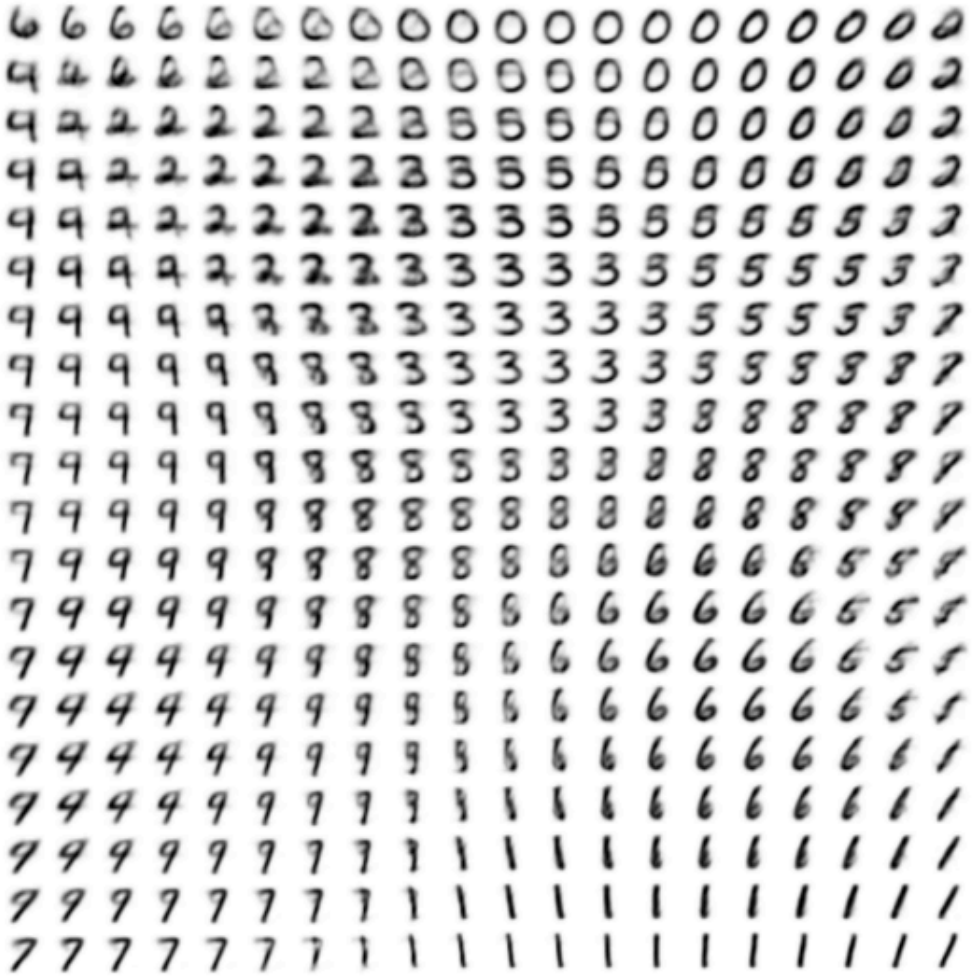
Generate Images



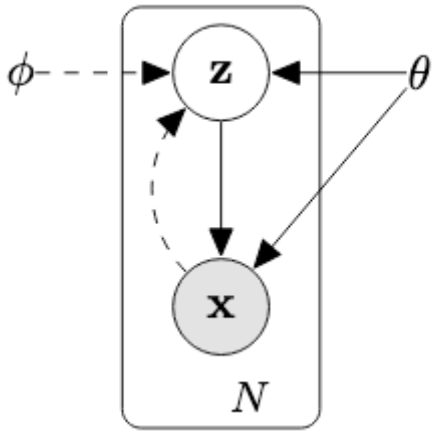


Deep Belief Nets, Hinton, Osindero, The, 2006

Generate Images



(b) Learned MNIST manifold

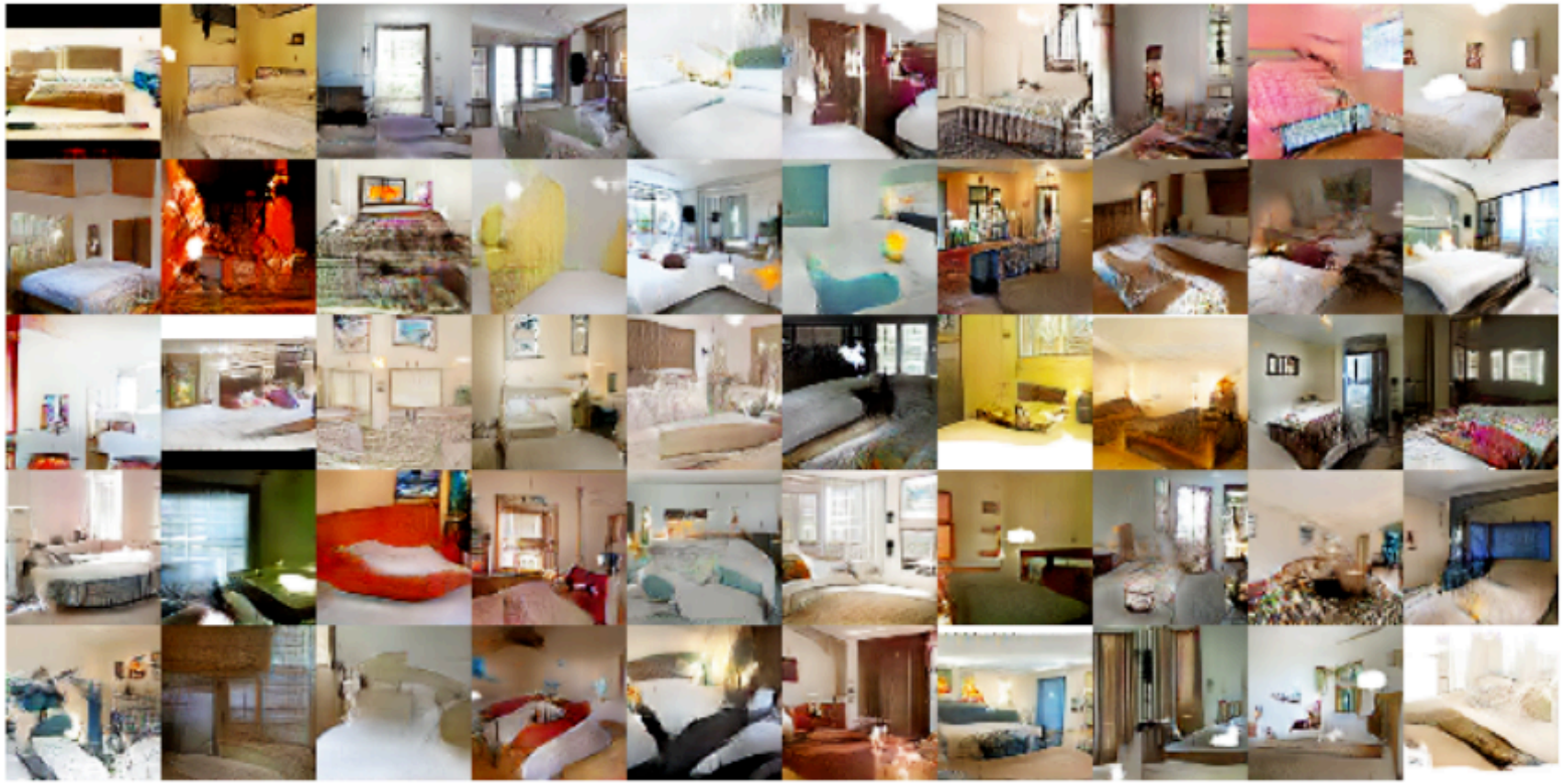


VAE, Kingma and Welling, 2013

Generate Images

GAN, Goodfellow et al. 2014

Generate Images



DCGAN, Radford, Metz, Chintala

Generate Audio

Figure 1: A second of generated speech.

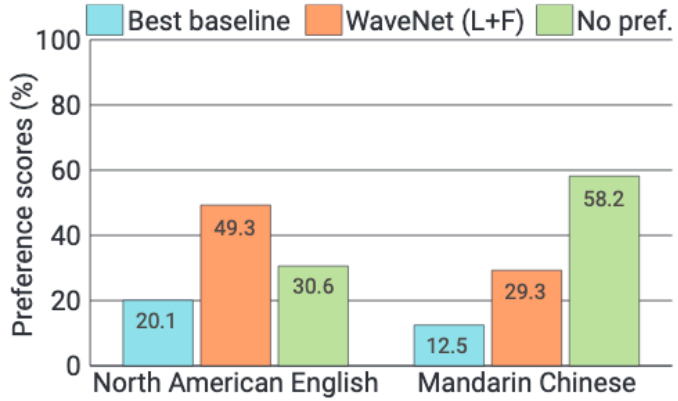


Figure 5: Subjective preference scores (%) of speech samples between (top) two baselines, (middle) two WaveNets, and (bottom) the best baseline and WaveNet. Note that LSTM and Concat correspond to LSTM-RNN-based statistical parametric and HMM-driven unit selection concatenative baseline synthesizers, and WaveNet (L) and WaveNet (L+F) correspond to the WaveNet conditioned on linguistic features only and that conditioned on both linguistic features and $\log F_0$ values.

WaveNet, Oord et al., 2018

Generate Video

Figure 1: Selected frames from videos generated by a DVD-GAN trained on Kinetics-600 at 256×256, 128 × 128, and 64 × 64 resolutions (top to bottom).

DVD-GAN Clark, Donaheu, Simonyan, 2019

Generate Text

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

Char-rnn, karpathy, 2015

Compression - Lossless

Model	Bits per byte
CIFAR-10	
PixelCNN (Oord et al., 2016)	3.03
PixelCNN++ (Salimans et al., 2017)	2.92
Image Transformer (Parmar et al., 2018)	2.90
PixelSNAIL (Chen et al., 2017)	2.85
Sparse Transformer 59M (strided)	**2.80**
Enwik8	
Deeper Self-Attention (Al-Rfou et al., 2018)	1.06
Transformer-XL 88M (Dai et al., 2018)	1.03
Transformer-XL 277M (Dai et al., 2018)	**0.99**
Sparse Transformer 95M (fixed)	**0.99**
ImageNet 64x64	
PixelCNN (Oord et al., 2016)	3.57
Parallel Multiscale (Reed et al., 2017)	3.7
Glow (Kingma & Dhariwal, 2018)	3.81
SPN 150M (Menick & Kalchbrenner, 2018)	3.52
Sparse Transformer 152M (strided)	**3.44**
Classical music, 5 seconds at 12 kHz	
Sparse Transformer 152M (strided)	**1.97**

Generative models provide better bit-rates than distribution-unaware methods like JPEG, etc.

Generating Long Sequences with Sparse Transformers, Child et. al, 2019

Likelihood-based models

- Say, we would like to synthesize data from its own distribution.
- We need to estimate p_{data} from samples $x^1, \ldots, x^n \sim p_{data}(x)$
- Once we learn this distribution, we can sample from it $x \sim p(x)$

Challenges

- We want to estimate distributions of complex, high dimensional data
  - A 128x128x3 image lies in about 50,000-dimensional space
- We want computational and statistical efficiency
  - Efficient training and model representation
  - Expressiveness and generalization
  - Sampling quality and speed
  - Compression rate and speed

Learning a parameterized function $p_\theta(x)$

- Function approximation: learn θ so that $p_\theta(x) \approx p_{data}(x)$
- Given data $x^1, \ldots, x^n$ sampled from a "true" distribution p_{data}
- Pose a search problem over parameters
  - $\operatorname{argmin}_\theta loss(\theta, x^1, \ldots, x^n)$
- Think of the loss as a distance between the distributions
- The training procedure can only see the empirical data distribution, not the true data distribution
  - We want the model to generalize

Maximum likelihood

- Given a dataset, find θ by solving

$$\arg\min_{\theta} \ \text{loss}(\theta, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = \frac{1}{n} \sum_{i=1}^{n} -\log p_\theta(\mathbf{x}^{(i)})$$

- Equivalent to minimizing KL divergence between the empirical distribution and the model

$$\hat{p}_{\text{data}}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[\mathbf{x} = \mathbf{x}^{(i)}]$$

$$\text{KL}(\hat{p}_{\text{data}} \,\|\, p_\theta) = \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\text{data}}}[-\log p_\theta(\mathbf{x})] - H(\hat{p}_{\text{data}})$$

Designing the model

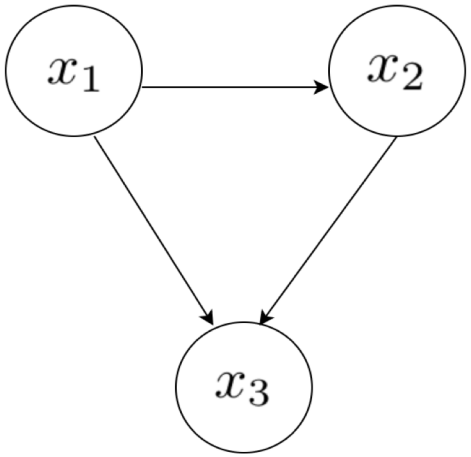
- We will choose p_θ to be deep neural networks
- Our design should satisfy

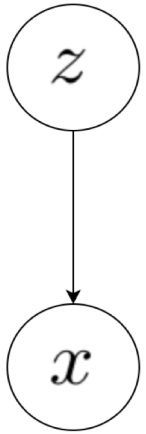
$$\text{for all } \theta, \quad \sum_{\mathbf{x}} p_\theta(\mathbf{x}) = 1 \quad \text{and} \quad p_\theta(\mathbf{x}) \geq 0 \quad \text{for all } \mathbf{x}$$

- $\log p_\theta(\mathbf{x})$ should be easy to evaluate and differentiate with respect to θ
- This can be a tricky set up!

Latent Variable Models (LVMs)

- Some random variables are hidden – we do not get to observe





Why Latent Variable Models

- Simpler, lower-dimensional representations of data are often possible.
- LVMs can automatically identify those hidden representations.

Why Latent Variable Models

- Other models such as AR are slow to sample because all pixels (observation dims) are assumed to be dependent on each other
- We can make part of observation space independent conditioned on some latent variables
  - LVMs can have faster sampling by explotigin statistical patterns.

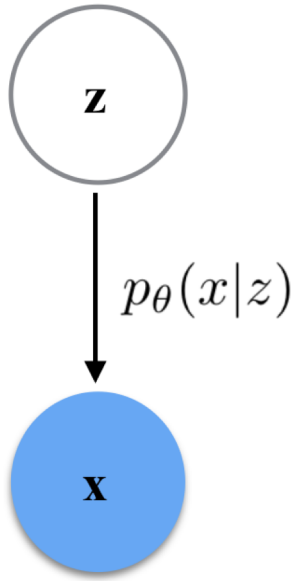
Latent Variable Models

- Sometimes, it’s possible to design a latent variable model with an understanding of the causal process that generates data
- In general, we don’t know what are the latent variables and how they interact with observations
  - Most popular models make little assumption about what are the latent variables
  - Best way to specify latent variables is still an active area of research

Example latent variable model

$$z = (z_1, z_2, \cdots, z_K) \sim p(z; \beta) = \prod_{k=1}^{K} \beta_k^{z_k} (1 - \beta_k)^{1 - z_k}$$

$$x = (x_1, x_2, \cdots, x_L) \sim p_\theta(x|z) \Leftrightarrow \text{Bernoulli}(x_i; \text{DNN}(z))$$



Latent Variable Model

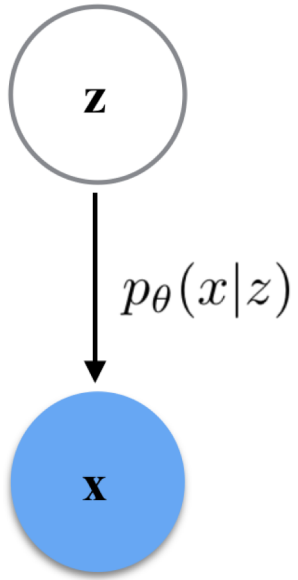
- Sample $z \sim p_Z(z)$

  $x \sim p_\theta(x \mid z)$

- Evaluate likelihood $p_\theta(x) = \sum_z p_Z(z) p_\theta(x \mid z)$

- Train $\max_\theta \sum_i \log p_\theta(x^{(i)}) = \sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} \mid z)$

- Representation $x \rightarrow z$



Training Latent Variable Models

- Objective:

$$\max_{\theta} \sum_{i} \log p_\theta(x^{(i)}) = \sum_{i} \log \sum_{z} p_Z(z) p_\theta(x^{(i)} \mid z)$$

- Scenario 1: z can only take on a small number of values -> exact objective tractable

- Scenario 2: z can take on an impractical number of values to enumerate -> approximate

Exact Likelihood Objective

- **Example**: mixture of 3 Gaussians, with uniform prior over components

$$p_\theta(x) = \sum_z p_Z(z) p_\theta(x \mid z)$$

$$p_Z(z = A) = p_Z(z = B) = p_Z(z = C) = \frac{1}{3}$$

$$p_\theta(x \mid z = k) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k)\right)$$

- Training objective:

$$\max_\theta \sum_i \log p_\theta(x^{(i)})$$

$$= \max_{\mu,\Sigma} \sum_i \log[\quad \frac{1}{3} \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_A|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_A)^\top \Sigma_A^{-1}(x^{(i)} - \mu_A)\right)$$

$$+ \frac{1}{3} \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_B|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_B)^\top \Sigma_B^{-1}(x^{(i)} - \mu_B)\right)$$

$$+ \frac{1}{3} \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_C|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_C)^\top \Sigma_C^{-1}(x^{(i)} - \mu_C)\right)]$$

Prior Sampling

- Main idea: if z can take on many values -> sample z

$$\sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} \mid z) \approx \sum_i \log \frac{1}{K} \sum_{k=1}^{K} p_\theta(x^{(i)} \mid z_k^{(i)}) \qquad z_k^{(i)} \sim p_Z(z)$$

- Run SGD on the approximate objective
- Challenge: When going to higher dimensional data, it becomes near impossible to be lucky enough that a sampled z is a good match for a data point $x^{(i)}$.

Importance Sampling Motivation

- Problem setting:
  - We want to compute $\mathbb{E}_{z\sim p_Z(z)}\left[f(z)\right]$
  - But
    1. Hard to sample from $p_Z(z)$
    2. Samples are not very informative

Importance Sampling - Algorithm

- **Formulation:**

$$
\begin{aligned}
\mathbb{E}_{z\sim p_Z(z)}\left[f(z)\right] &= \sum_z p_Z(z) f(z) \\
&= \sum_z \frac{q(z)}{q(z)} p_Z(z) f(z) \\
&= \mathbb{E}_{z\sim q(z)}\left[\frac{p_Z(z)}{q(z)} f(z)\right] \\
&\approx \frac{1}{K}\sum_{k=1}^{K} \frac{p_Z(z^{(k)})}{q(z^{(k)})} f(z^{(k)}) \quad \text{with} \quad z^{(k)} \sim q(z)
\end{aligned}
$$

➢ We can sample from q to compute expectation wrt p

Importance Sampling for Latent Variable Model

- **Training Objective**

$$\sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} \mid z) \approx \sum_i \log \frac{1}{K} \sum_{k=1}^{K} \frac{p_Z(z_k^{(i)})}{q(z_k^{(i)})} p_\theta(x^{(i)} \mid z_k^{(i)}) \quad \text{with} \quad z_k^{(i)} \sim q(z_k^{(i)}))$$

- **Good proposal distribution $q(z)$?**
  - We want samples compatible with $x^{(i)}$
  - How about

$$q(z) = p_\theta(z \mid x^{(i)}) = \frac{p_\theta(x^{(i)} \mid z) p_Z(z)}{p_\theta(x^{(i)})}$$

  - **Issue**: not clear how to sample from this distribution…

Importance Sampling Proposal Distribution

- General Principle of Variational Approach:
  - We can't use p we want
  - So, instead, we propose a parameterized distribution q we know we can work with easily
  - And try to find a parameter setting that makes it as good as possible

- E.g. find $q(z) = \mathcal{N}(z; \mu, \sigma^2)$ as close as possible to $p_\theta(z \mid x^{(i)}) = \frac{p_\theta(x^{(i)} \mid z) p_Z(z)}{p_\theta(x^{(i)})}$

Importance Sampling Proposal Distribution

- Optimize to find q

$$
\begin{aligned}
& \min_{q(z)} \mathrm{KL}(q(z) \| p_\theta(z \mid x^{(i)}) \\
= \quad & \min_{q(z)} \mathbb{E}_{z \sim q(z)} \log \left(\frac{q(z)}{p_\theta(z|x^{(i)})} \right) \\
= \quad & \min_{q(z)} \mathbb{E}_{z \sim q(z)} \log \left(\frac{q(z)}{p_\theta(x^{(i)}|z) p_Z(z) / p_\theta(x^{(i)})} \right) \\
= \quad & \min_{q(z)} \mathbb{E}_{z \sim q(z)} \left[\log q(z) - \log p_Z(z) - \log p_\theta(x^{(i)} \mid z) \right] + \log p_\theta(x^{(i)}) \\
= \quad & \min_{q(z)} \mathbb{E}_{z \sim q(z)} \left[\log q(z) - \log p_Z(z) - \log p_\theta(x^{(i)} \mid z) \right] + \text{constant independent of z}
\end{aligned}
$$

- Note: all needed quantities in the objective readily computable

Amortized Inference

- **General Idea of Amortization**: if same inference problem needs to be solved many times, can we parameterize a neural network to solve it?
- **Our case:** for all $x^{(i)}$ we want to solve:

$$\min_{q(z)} \mathrm{KL}(q(z) \| p_\theta(z \mid x^{(i)})$$

- **Amortized formulation:**

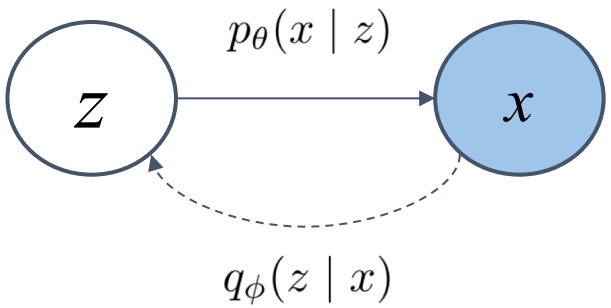
$$\min_{\phi} \sum_{i} \mathrm{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)}))$$

- Pro: faster, Con: not as precise

Amortized Inference

- **Amortized formulation:**

$$\min_{\phi} \sum_{i} \mathrm{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)}))$$



- **E.g.** $q_\phi(z \mid x) = \mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x))$

Equivalently: $z = \mu_\phi(x) + \varepsilon\sigma_\phi(x)$ with $\varepsilon \sim \mathcal{N}(0, I)$

Variational Lower Bound (VLB)

- We now have an objective amenable to stochastic optimization

$$D_{\mathrm{KL}}\left[q_x(z) \parallel p(z|x)\right] = \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z)\right] + \log p(x)$$

- Turns out

$$\begin{aligned}\log p(x) &= -\mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z)\right] + D_{\mathrm{KL}}\left[q_x(z) \parallel p(z|x)\right] \\ &= \underbrace{\mathbb{E}_{z\sim q_x(z)}\left[\log p(z) + \log p(x|z) - \log q_x(z)\right]}_{\text{Variational Lower Bound}} + \underbrace{D_{\mathrm{KL}}\left[q_x(z) \parallel p(z|x)\right]}_{\geq 0}\end{aligned}$$

VLB Maximization

- Given a data distribution $x \sim p_{\text{data}}$, we can know train the generative model by maximizing the VLB under data distribution

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\mathbb{E}_{\mathbf{z}\sim \mathbf{q}_\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{z}) + \log p_\theta(\mathbf{x} \mid \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})\right]\right] \leq \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log p(\mathbf{x})\right]$$

VLB Maximization

- Given a data distribution $x \sim p_{\text{data}}$, we can know train the generative model by maximizing the VLB under data distribution

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\mathbb{E}_{\mathbf{z}\sim \mathbf{q}_\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{z}) + \log p_\theta(\mathbf{x} \mid \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})\right]\right] \leq \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log p(\mathbf{x})\right]$$

HOW CAN WE PERFORM STOCHASTIC GRADIENT?

Pathwise Derivative (PD)

- When z is continuous, we can cast z as a function of a simple noise such as standard Gaussian.

$$z = g(\epsilon, \phi), \epsilon \sim \mathcal{N}(0, I)$$

$$\mathbb{E}_{z \sim q_\phi(\cdot|x)} [f(z)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} [f(g(\epsilon, \phi))]$$

- When f is differentiable

$$\nabla_\phi \mathbb{E}_{z \sim q_\phi(\cdot|x)} [f(z)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} [\nabla_\phi f(g(\epsilon, \phi))]$$

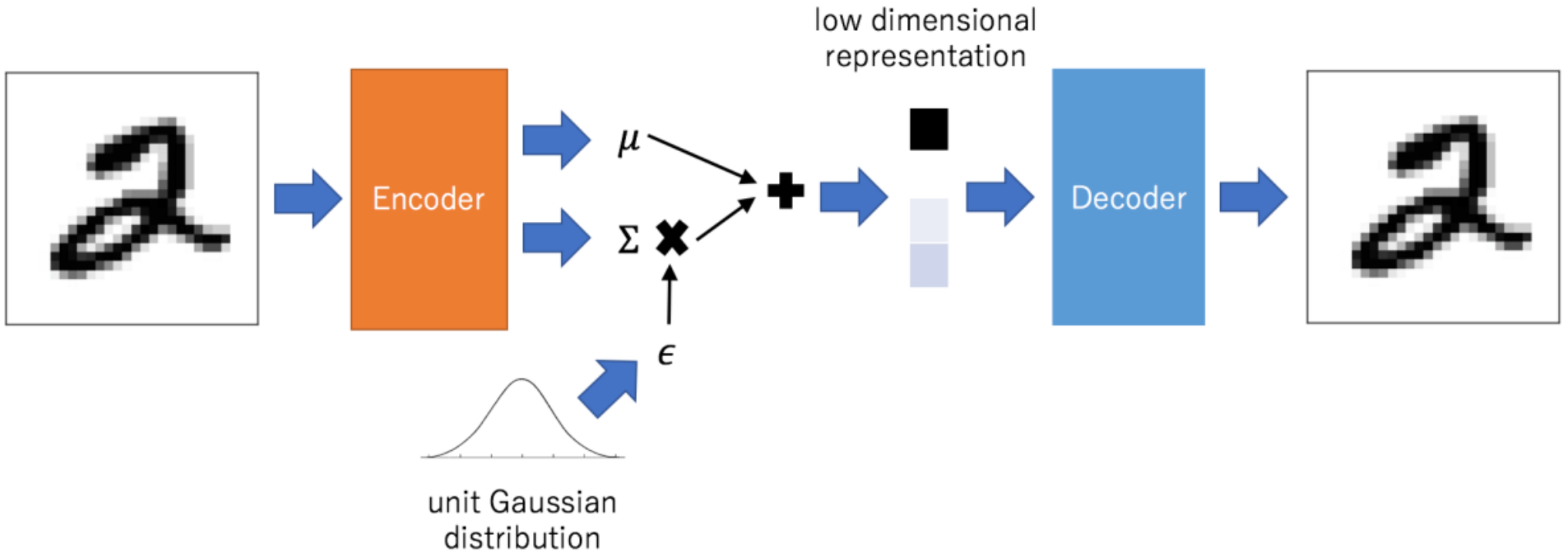
Variational AutoEncoder (VAE)

- $q_\phi(\mathbf{z}\,|\mathbf{x})$ is modeled as a Gaussian with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$
  - A DNN encoder of $\mathbf{x}$
- The DNN decoder $p_\theta(\mathbf{x}|\mathbf{z})$is differentiable
- Let $\mathbf{z} = \Sigma^{1/2}(\mathbf{x};\phi)\epsilon + \mu(\mathbf{x};\phi)$
- Then,

$$\begin{aligned}\mathrm{VLB} &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z}) - \log q_\phi(\mathbf{z}\mid\mathbf{x}) + \log p(\mathbf{z})\right] \\ &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z})\right] - KL(q_\phi(\mathbf{z}\mid\mathbf{x})||p(\mathbf{z}))\end{aligned}$$

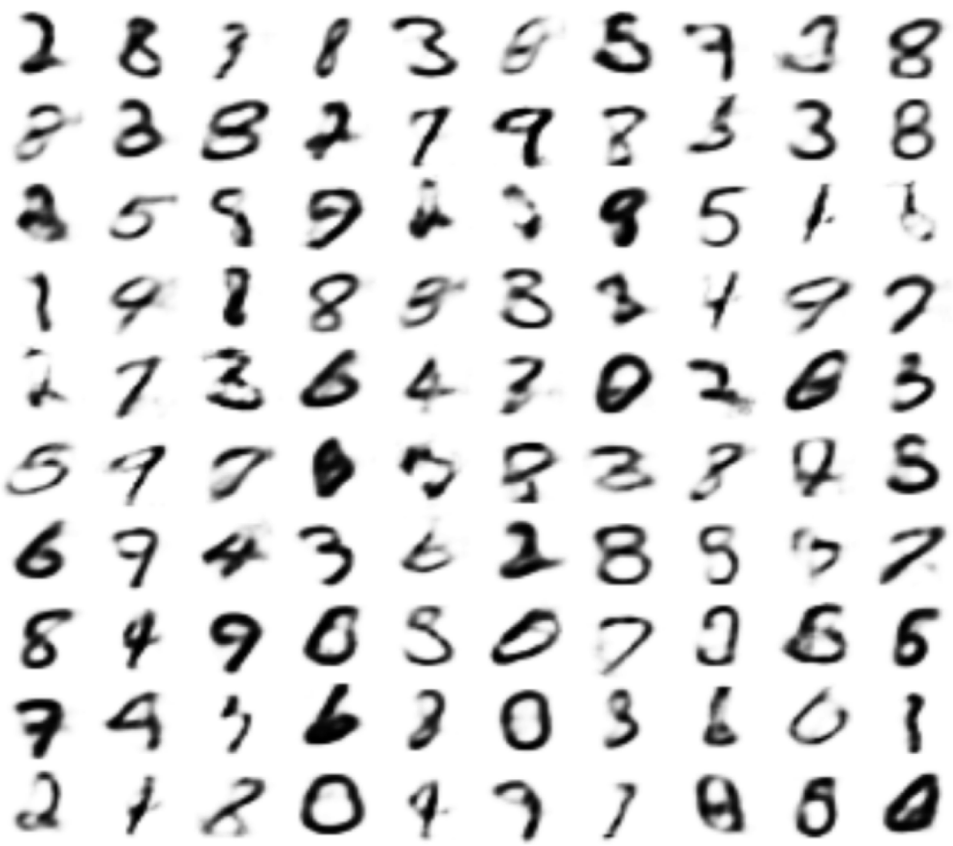
- Gradients wrt to both θ and ϕ can now be effectively computed with SGD.

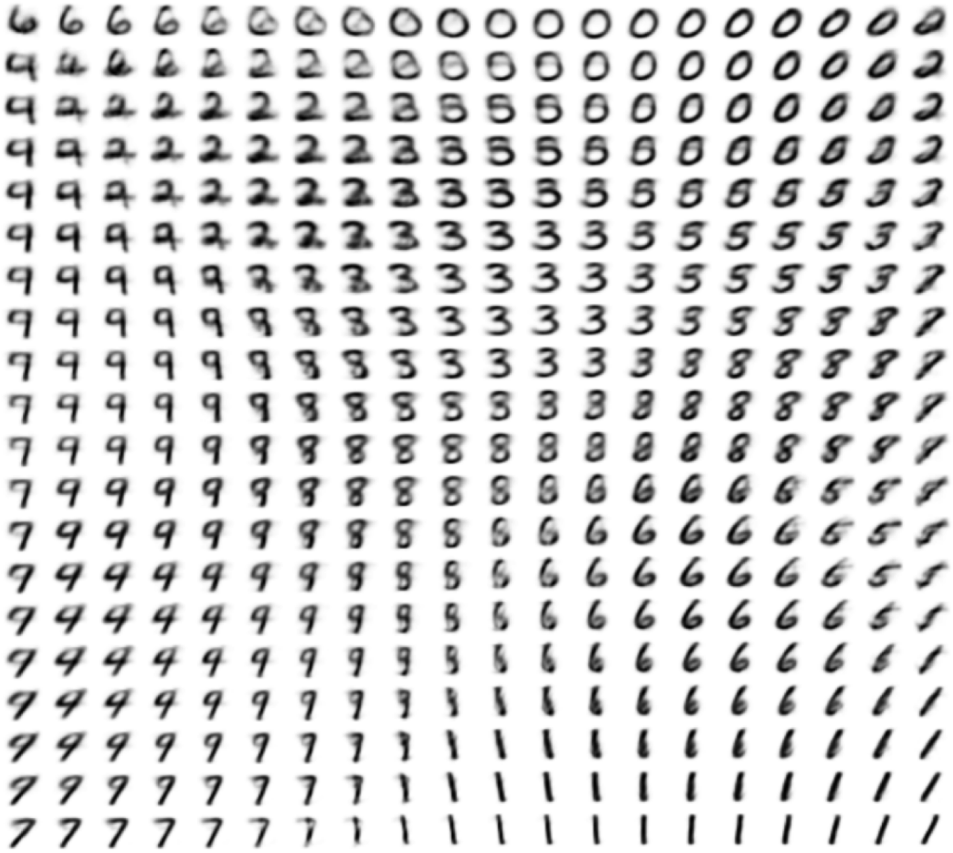
Variational AutoEncoder (VAE)



https://mlexplained.com/2017/12/28/an-intuitive-explanation-of-variational-autoencoders-vaes-part-1/

Variational AutoEncoder (VAE)





Let's recall Generative Adversarial Networks

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

- Two player minimax game between generator (G) and a discriminator (D)
- D tries to maximize the log-likelihood for the binary classification problem
  - Data: real (1)
  - Generated: fake (0)
- (G) tries to minimize the log-likelihood of its samples being classified as "fake" by the discriminator (D)

Let’s recall Generative Adversarial Networks

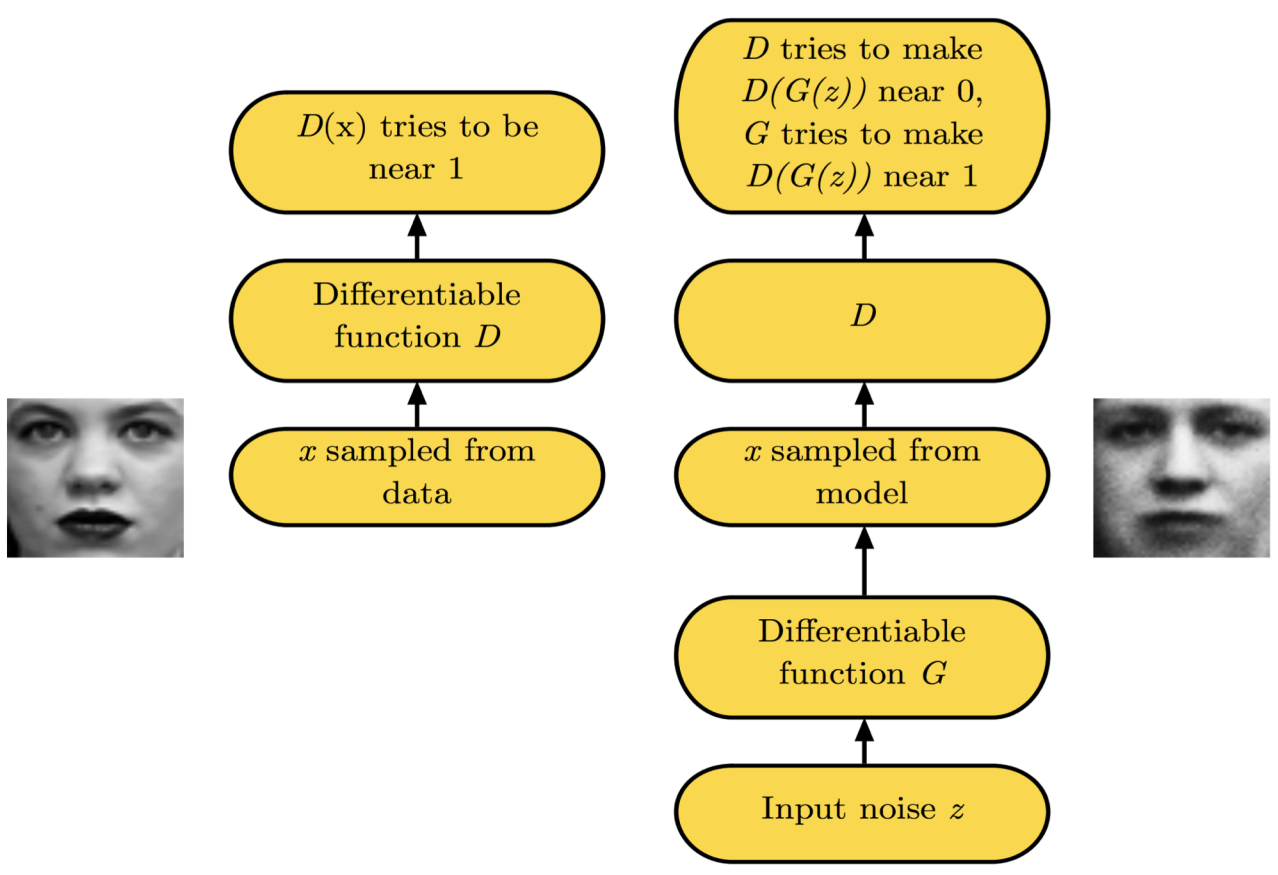


Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

Key pieces of GANs

- Fast sampling
- No inference
- Notion of optimizing directly for what you care about

GAN: Bayes-Optimal Discriminator

- What is the optimal discriminator given generated and true distributions?

$$
\begin{aligned}
V(G,D) &= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p(z)}\left[\log(1-D(G(z)))\right] \\
&= \int_x p_{\text{data}}(x)\log D(x)dx + \int_z p(z)\log(1-D(G(z)))dz \\
&= \int_x p_{\text{data}}(x)\log D(x)dx + \int_x p_g(x)\log(1-D(x))dx \\
&= \int_x \left[p_{\text{data}}(x)\log D(x) + p_g(x)\log(1-D(x))\right]dx
\end{aligned}
$$

$$
\nabla_y \left[a\log y + b\log(1-y)\right] = 0 \implies y^* = \frac{a}{a+b} \quad \forall \quad [a,b] \in \mathbb{R}^2 \backslash [0,0]
$$

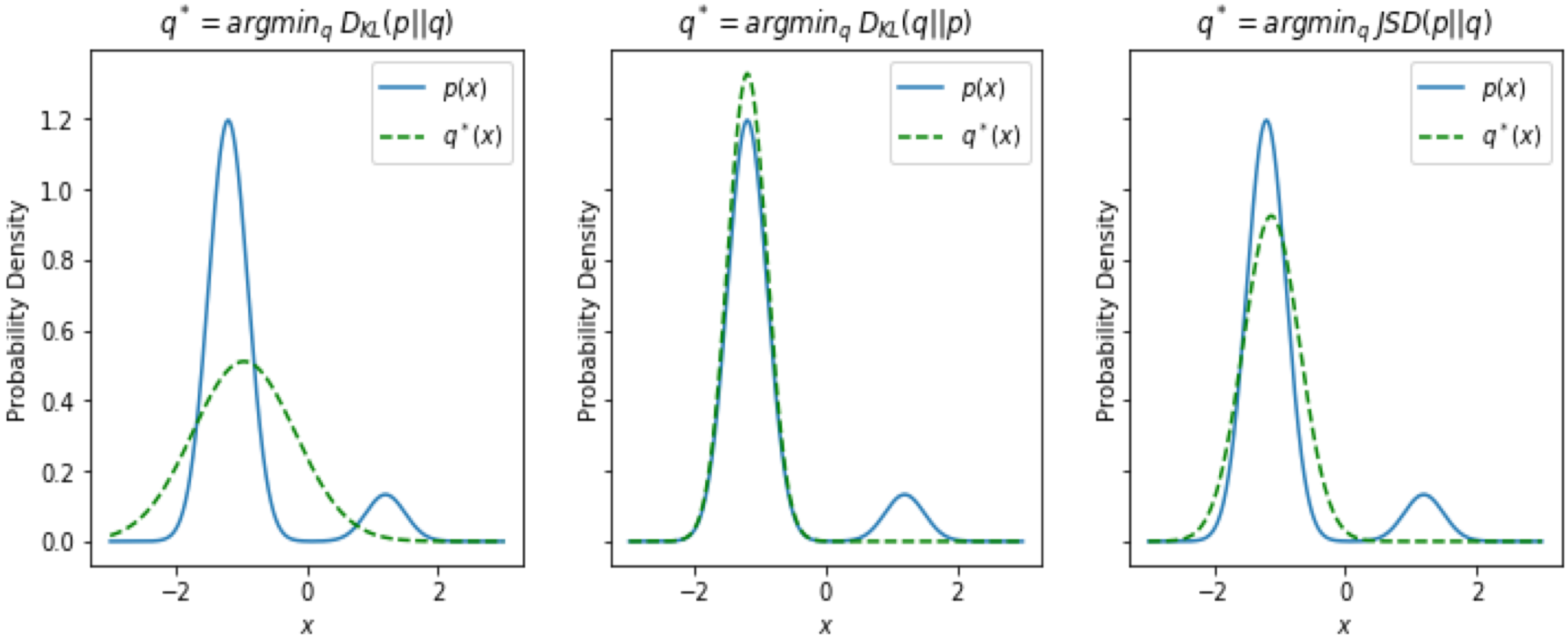
$$
\implies D^*(x) = \frac{p_{\text{data}}(x)}{(p_{\text{data}}(x) + p_g(x))}
$$

GAN: Generator Objective under Bayes-Optimal Discriminator D^*

$$
\begin{aligned}
V(G, D^*) &= \mathbb{E}_{x \sim p_{\text{data}}} \left[\log D^*(x) \right] + \mathbb{E}_{x \sim p_g} \left[\log(1 - D^*(x)) \right] \\
&= \mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right] \\
&= -\log(4) + \underbrace{KL\left(p_{\text{data}} \| \left(\frac{p_{\text{data}} + p_g}{2} \right) \right) + KL\left(p_{\text{g}} \| \left(\frac{p_{\text{data}} + p_g}{2} \right) \right)}_{\text{(Jensen-Shannon Divergence (JSD) of } p_{\text{data}} \text{ and } p_g) \geq 0}
\end{aligned}
$$

$$
V(G^*, D^*) = -\log(4) \text{ when } p_g = p_{\text{data}}
$$

KL and JSD



Deep Learning Textbook (Goodfellow 2016)- Chapter 3

Wasserstein Distance

- Another measure inspired from Optimal Transport is the Earth Mover (EM) distance

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[||x - y|| \right]$$

- **Goal**: Design a GAN objective function such that the generator minimizes the EM distance between data and generated distributions

Wasserstein GAN

Wasserstein GAN

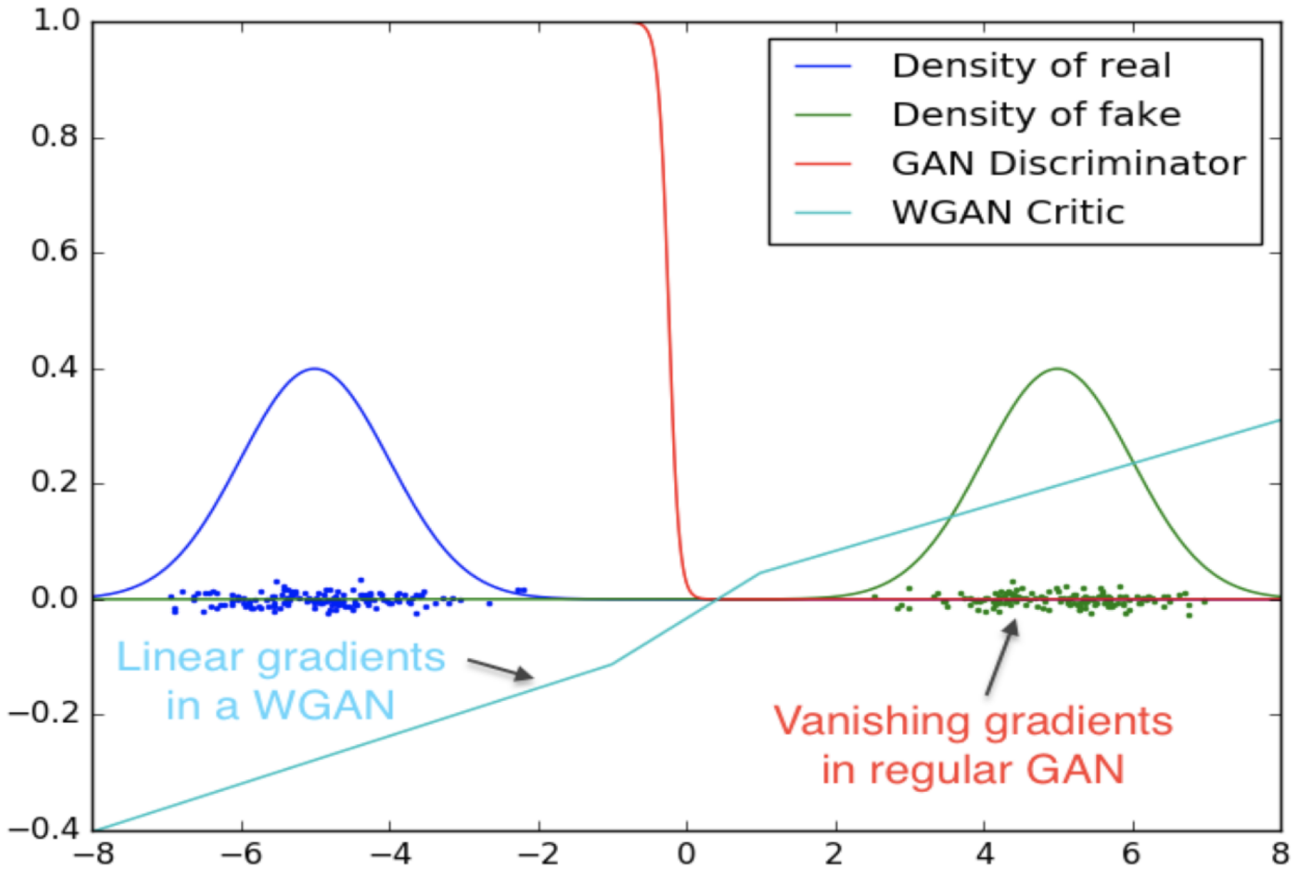
Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

Kantorovich Rubinstein Duality: $W(P_r, P_g) = \sup_{\|f_L\| \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)]$

- Supremum over linear function space

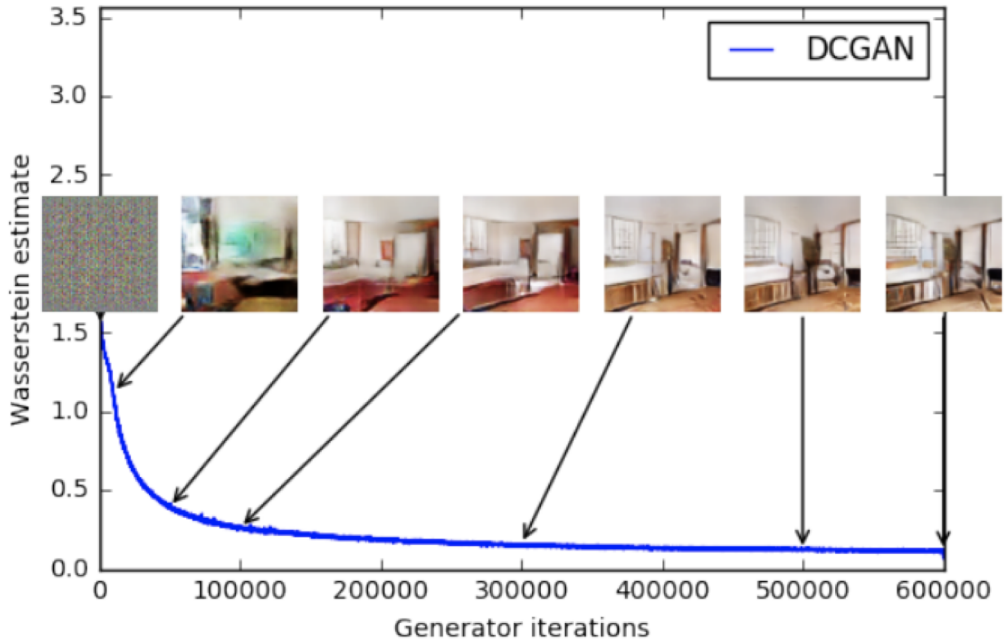
Linear gradients



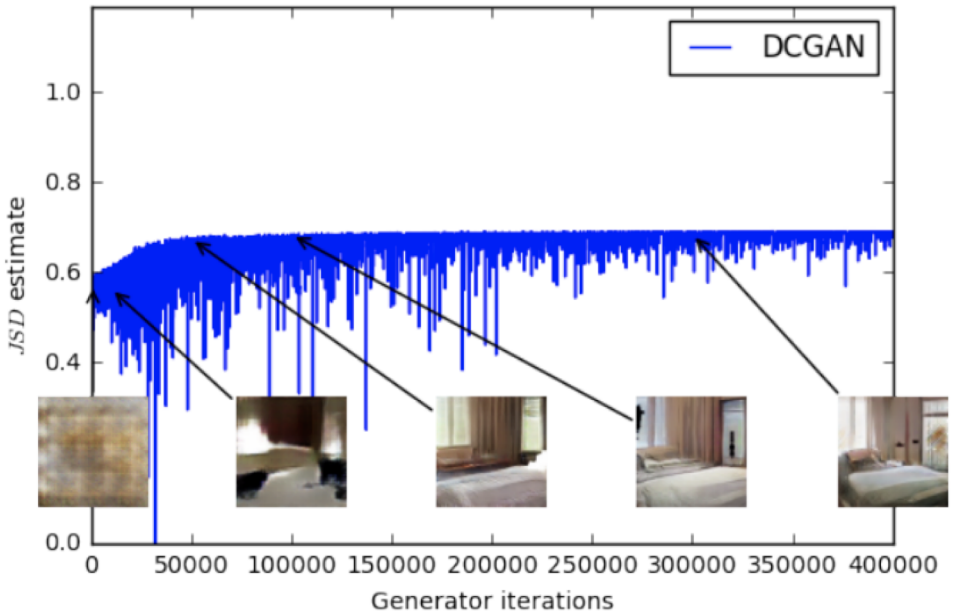
(Arjovsky et al 2017)

Wasserstein distance correlated with sample quality

Wasserstein Estimate

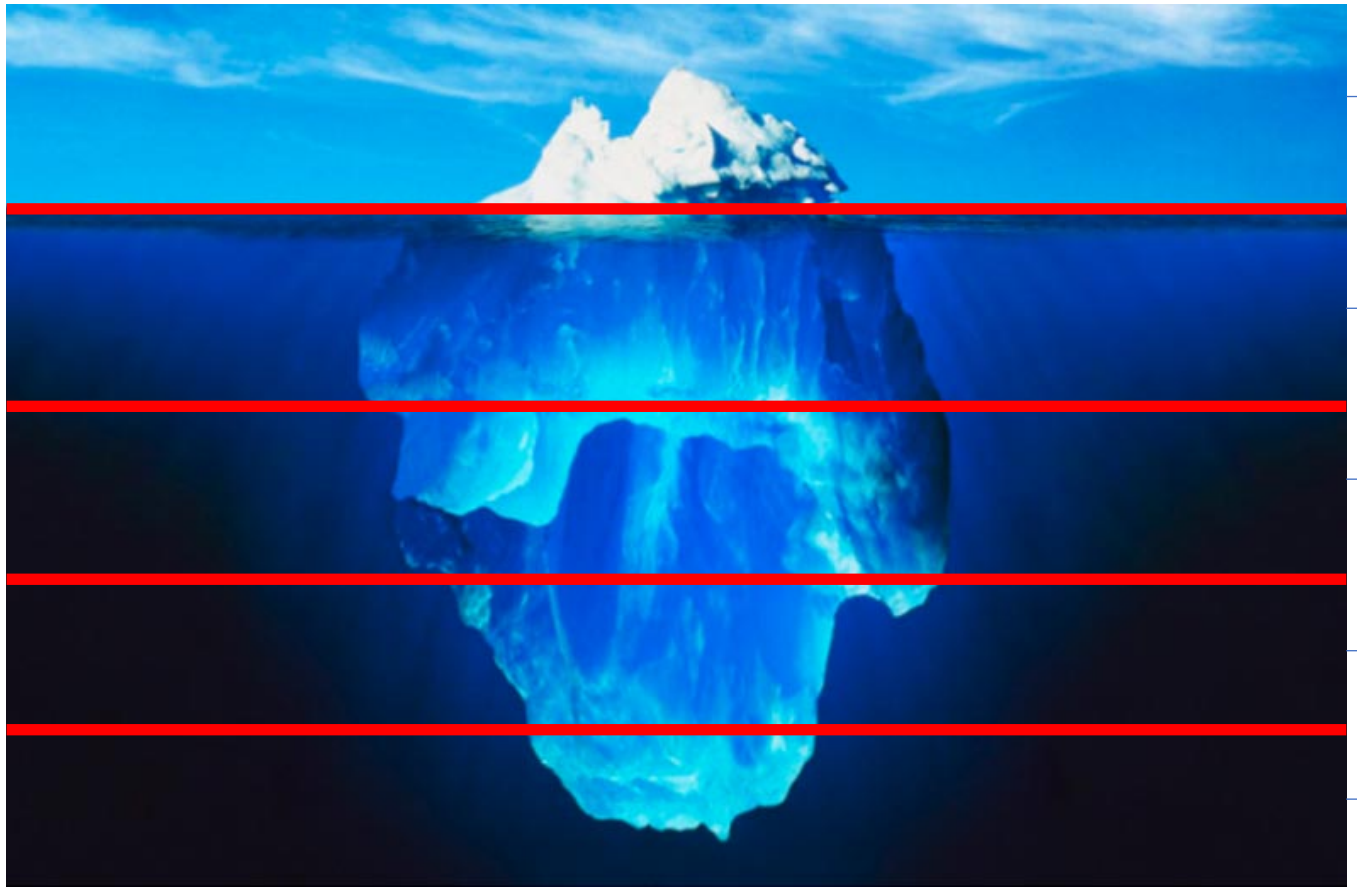


JSD Estimate



(Arjovsky et al 2017)

Wrapping Up



What we covered in this course

Semi-supervised Learning

Reinforcement Learning

Adversarial Robustness

And more …

Thank You!