Deep Generative Models

Adji Bousso Dieng

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Deep Learning Indaba
Nairobi, Kenya
August, 2019

@adjiboussodieng

Setup

- Observations $\mathbf{x}_1, \ldots, \mathbf{x}_N \overset{iid}{\sim} p_d(\mathbf{x})$
- Model $\mathbf{x} \sim p_\theta(\mathbf{x})$
- Goal: learn θ to make $p_\theta(\mathbf{x})$ as "close" to $p_d(\mathbf{x})$ as possible.

$$\theta^* = \arg\min_\theta \mathrm{D}\left(p_d(\mathbf{x}) \| p_\theta(\mathbf{x})\right)$$

$$\mathrm{D}\left(p\|q\right) \geq 0 \text{ and } \mathrm{D}\left(p\|q\right) = 0 \iff p = q \text{ a.e.}$$

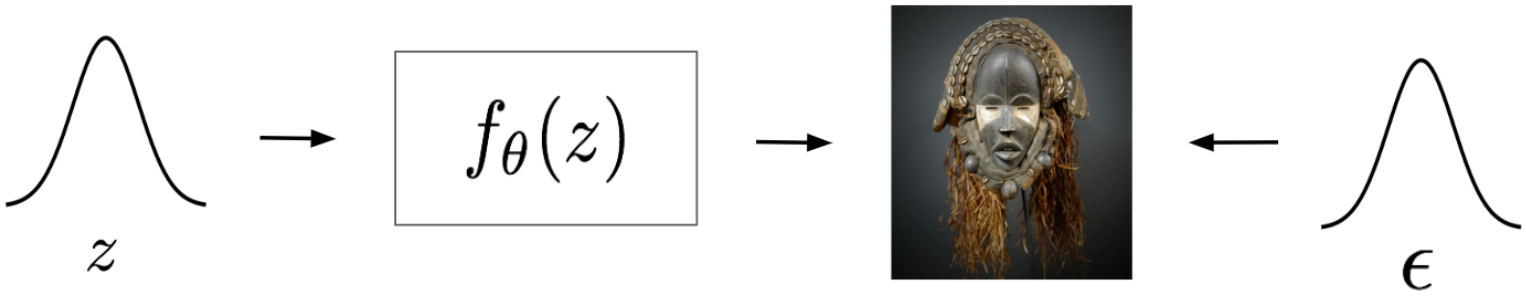
- Many approaches to this...
- Focus of this talk:
  - Variational autoencoders
  - Generative Adversarial Networks

Roadmap

1. **Variational Autoencoders**
2. Generative Adversarial Networks
3. Concluding Remarks

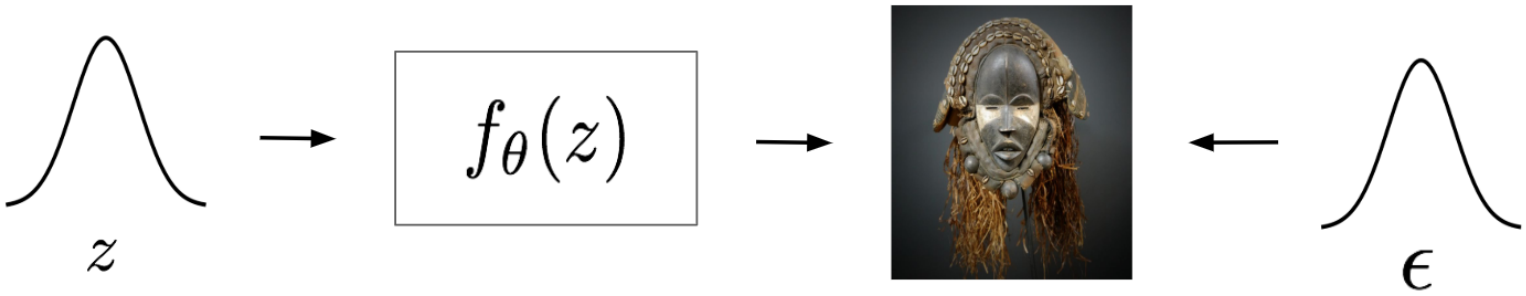
Variational Autoencoders

Prescribed Models



Generative distribution $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$

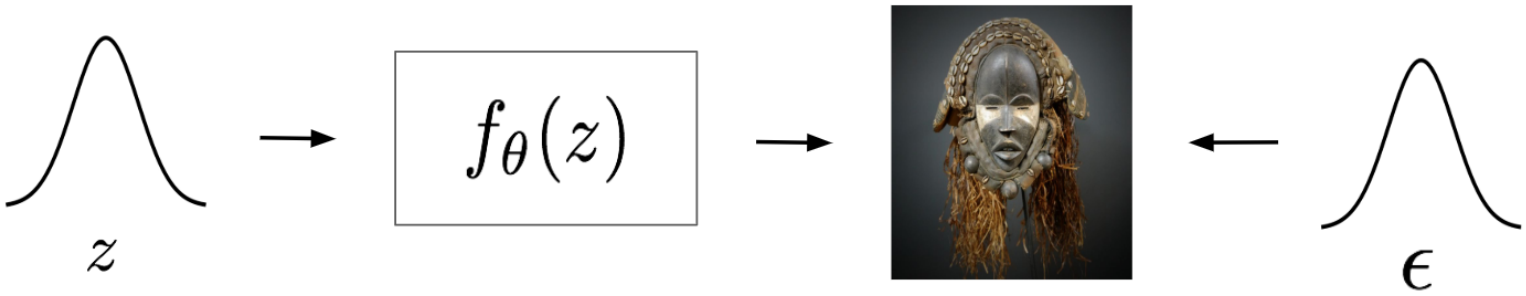
Prescribed Models



Generative distribution $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$

Distribution	Name	Form
$p(\mathbf{z})$	Prior	$\mathcal{N}(\mathbf{0}, \mathbf{I})$ or $\mathcal{U}(\mathbf{0}, \mathbf{I})$
$p_\theta(\mathbf{x}\|\mathbf{z})$	"Likelihood"	Gaussian or Bernoulli
$p_\theta(\mathbf{x}, \mathbf{z})$	Joint	$p_\theta(\mathbf{x}\|\mathbf{z}) \cdot p(\mathbf{z})$
$p_\theta(\mathbf{z}\|\mathbf{x})$	Posterior	$\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{x})}$

A Typical VAE



Generative distribution $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$

Distribution	Form	Sampling
$p(\mathbf{z})$	$\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\mathbf{0}, \mathbf{I})$
$p_\theta(\mathbf{x}\|\mathbf{z})$	$\mathcal{N}(f_\theta(\mathbf{z}), \sigma^2 \mathbf{I})$	$\mathbf{x} = f_\theta(\mathbf{z}) + \sigma \odot \epsilon$

Maximum Likelihood

A simple model $\mathbf{x}_1, \ldots, \mathbf{x}_N \overset{iid}{\sim} \mathcal{N}(\theta, 1)$

Find θ that maximizes the likelihood

$\theta^* = \arg\max_\theta p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \arg\max_\theta \prod_{i=1}^N p_\theta(\mathbf{x}_i)$

What is the optimal MLE solution θ^*?

$$\mathcal{L}_{\text{MLE}} = \log \prod_{i=1}^N p_\theta(\mathbf{x}_i) = \sum_{i=1}^N \log p_\theta(\mathbf{x}_i) = -\frac{1}{2}\sum_{i=1}^N (\mathbf{x}_i - \theta)^2$$

$$\nabla_\theta \mathcal{L}_{\text{MLE}} = -\frac{1}{2}\sum_{i=1}^N \nabla_\theta (\mathbf{x}_i - \theta)^2 = \sum_{i=1}^N (\mathbf{x}_i - \theta) = \sum_{i=1}^N \mathbf{x}_i - N \cdot \theta$$

The optimal solution θ^* is such that $\nabla_\theta \mathcal{L}_{\text{MLE}} = 0 \rightarrow \theta^* = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i$

Maximum Likelihood

Reconsider the VAE model

$$\mathbf{x}_1, \ldots, \mathbf{x}_N \overset{iid}{\sim} p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

Find θ that maximizes the likelihood

$$\theta^* = \arg\max_\theta p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \arg\max_\theta \prod_{i=1}^N p_\theta(\mathbf{x}_i)$$

What is the optimal MLE solution θ^*?

$$\mathcal{L}_{\text{MLE}} = \log \prod_{i=1}^N p_\theta(\mathbf{x}_i) = \sum_{i=1}^N \log p_\theta(\mathbf{x}_i) = \sum_{i=1}^N \log \int p_\theta(\mathbf{x}_i \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

Oooops...

A Tractable Proxy for Learning

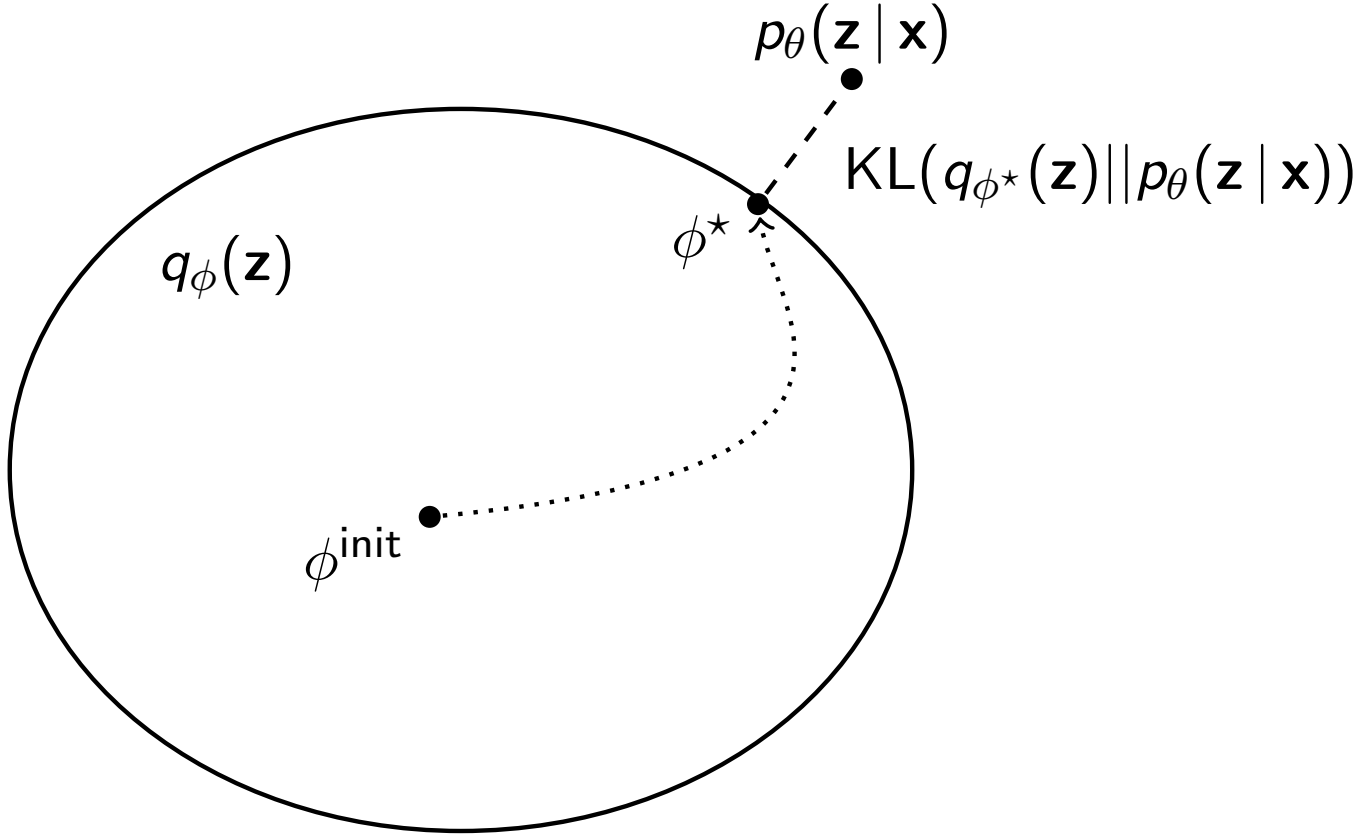
Bound the log marginal likelihood of each datapoint

$$\log p_\theta(\mathbf{x}_i) = \log \int p_\theta(\mathbf{x}_i, \mathbf{z}) d\mathbf{z} = \log \int \frac{p_\theta(\mathbf{x}_i, \mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}) d\mathbf{z}$$
$$= \log \mathbb{E}_{q(\mathbf{z})} \left[\frac{p_\theta(\mathbf{x}_i, \mathbf{z})}{q(\mathbf{z})} \right] \geq \mathbb{E}_{q(\mathbf{z})} \log \left[\frac{p_\theta(\mathbf{x}_i, \mathbf{z})}{q(\mathbf{z})} \right]$$

Note bound is tight when $q(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x}_i)$ a.e.

If we know $q(\mathbf{z})$ we can learn θ by maximizing the bound

Variational Inference



$$\log p_\theta(\mathbf{x}) = \mathrm{ELBO} + \mathrm{KL}(q_\phi(\mathbf{z}) || p_\theta(\mathbf{z} \mid \mathbf{x}))$$

$$\mathrm{ELBO} = \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}) \right]$$

An Impractical Learning Algorithm

Algorithm 1: Variational Expectation Maximization

Result: Optimized model parameters θ^*

initialize model parameters θ to θ_0;

for *iteration* $t = 1, 2, \ldots$ **do**

Find

$$\phi^* = \arg\max_{\phi} \mathit{ELBO} = \arg\max_{\phi} \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log p_{\theta_{t-1}}(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})\right];$$

Sample a minibatch $\mathcal{B}$ containing $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(b)}, \ldots \mathbf{x}^{(|\mathcal{B}|)}$;

Compute ELBO $= \sum_{b \in \mathcal{B}} \mathbb{E}_{q_{\phi_b^*}(\mathbf{z})} \left[\log p_\theta(\mathbf{x}^{(b)}, \mathbf{z}) - \log q_{\phi_b^*}(\mathbf{z})\right]$;

Update model parameters using SGD $\theta_t := \theta_{t-1} + \rho \cdot \nabla_\theta \mathit{ELBO}$

end

Amortized Inference

- → Explicitly condition on $\mathbf{x}$ and define $q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{N} q_\phi(\mathbf{z}_i|\mathbf{x}_i)$
- → ϕ denotes the parameters of a shared neural network
- → Each factor is

$$q_\phi(\mathbf{z}_i|\mathbf{x}_i) = \mathcal{N}(\mu_\phi(\mathbf{x}_i), \mathbf{\Sigma}_\phi(\mathbf{x}_i))$$

- → Sample using reparameterization to avoid high variance

$$\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|\mathbf{x}_i) \iff \mathbf{z}_i = \mu_\phi(\mathbf{x}_i) + \mathbf{\Sigma}_\phi(\mathbf{x}_i)^{\frac{1}{2}} \odot \epsilon$$

Instead of optimizing a set of parameters for each data point
... optimize parameters of a shared neural network!

Approximate Variational Expectation Maximization

Instead of optimizing ϕ to convergence at each iteration

... take one gradient step of the ELBO on a minibatch

Algorithm 2: Learning with Variational Autoencoders

Result: Optimized model parameters θ^*

initialize model and variational parameters θ, ϕ to θ_0, ϕ_0;

for *iteration* $t = 1, 2, \ldots$ **do**

  Sample a minibatch $\mathcal{B}$ containing $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(b)}, \ldots \mathbf{x}^{(|\mathcal{B}|)}$;

  Compute ELBO $= \sum_{b \in \mathcal{B}} \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log p_\theta(\mathbf{x}^{(b)}, \mathbf{z}) - \log q_\phi(\mathbf{z}) \right]$;

  Update variational parameters $\phi_t := \phi_{t-1} + \rho \cdot \nabla_\phi ELBO$;

  Update model parameters $\theta_t := \theta_{t-1} + \rho \cdot \nabla_\theta ELBO$

end

What Does Maximizing The ELBO Do?

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]$$

$\rightarrow$ Maximizing ELBO w.r.t θ approximates maximum likelihood

$$\nabla_\theta \text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\theta \log p_\theta(\mathbf{x}|\mathbf{z})\right]$$

$\rightarrow$ Maximizing ELBO w.r.t ϕ lets $q_\phi(\mathbf{z}|\mathbf{x})$ be "close" to true posterior

$$\nabla_\phi \text{ELBO}(\theta, \phi) = -\nabla_\phi \,\text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$$

This is because $\log p_\theta(\mathbf{x}) = \text{ELBO}(\theta, \phi) + \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}\,|\,\mathbf{x}))$

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}, \mathbf{z})\right] + \mathcal{H}(q_\phi)$$

Shades Of ELBO

ELBO maximization as entropy regularization

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z})\right] + \mathcal{H}(q_\phi)$$

ELBO maximization as KL regularization

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

ELBO maximization as joint distribution matching

$$\text{ELBO}(\theta, \phi) = -KL(q_\phi(\mathbf{z}, \mathbf{x})||p_\theta(\mathbf{x}, \mathbf{z}))$$
$$q_\phi(\mathbf{z}, \mathbf{x}) = q_\phi(\mathbf{z}|\mathbf{x}) p_d(\mathbf{x})$$

Posterior Collapse

Model is learned to make the posterior look like the prior :-(

- → Manifestation:
  $KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$ goes to zero quickly after start of training
- → Consequences:
  (1) bad generalization performance (log-likelihood)
  (2) defeats the purpose of posterior inference
- → Many remedies exist

Posterior Collapse Remedies

- → Richer variational approximations:
  - + hierarchical variational models [Ranganath et al. 2015]
  - + normalizing flows [Rezende & Mohamed, 2016]
  - + implicit variational distributions [Hoffman 2017]
- → Richer priors [Tomczak & Welling 2017]
- → Change likelihood [Dieng et al. 2019]
- → Change the objective:
  - + KL annealing [Bowman et al., 2015],
  - + penalized ELBO [Higgins et al., 2017]
  - + regularize with mutual information [Zhao et al., 2017]
  - + adversarial regularizer [Makhzan et al., 2015]
- → Change the optimization procedure:
  - + semi-amortized VAEs [Kim et al., 2018]
  - + Lagging inference networks [He et al., 2019]

Applications of VAEs: Chemistry

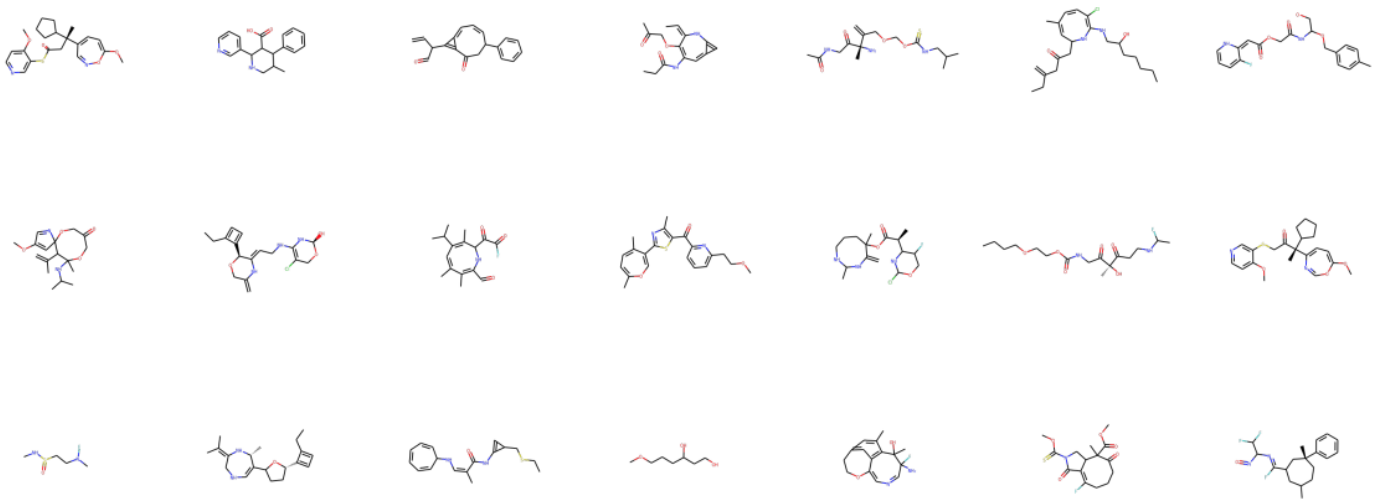


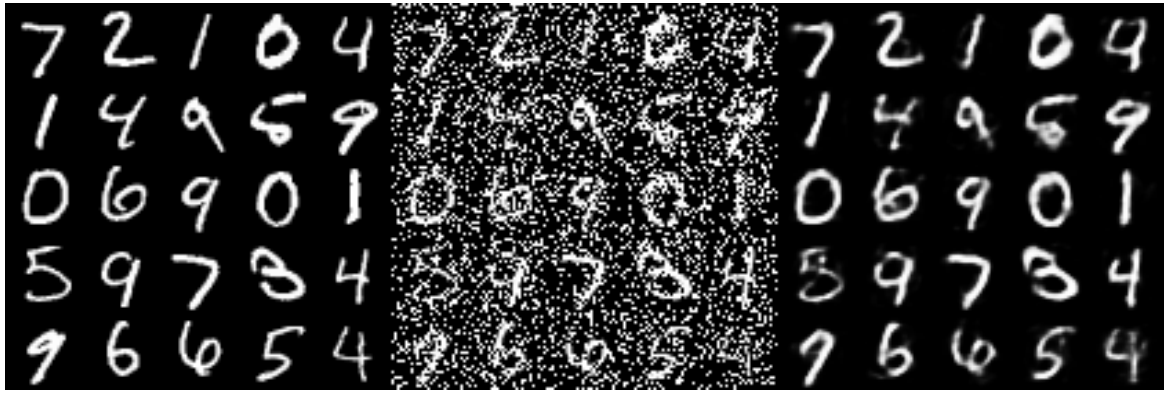
Figure 8: Molecules decoded from randomly-sampled points in the latent space of the ZINC VAE.

[Gomez-Bombarelli et al., 2017]

→ molecule generation

→ automatic drug design

Applications of VAEs: Vision

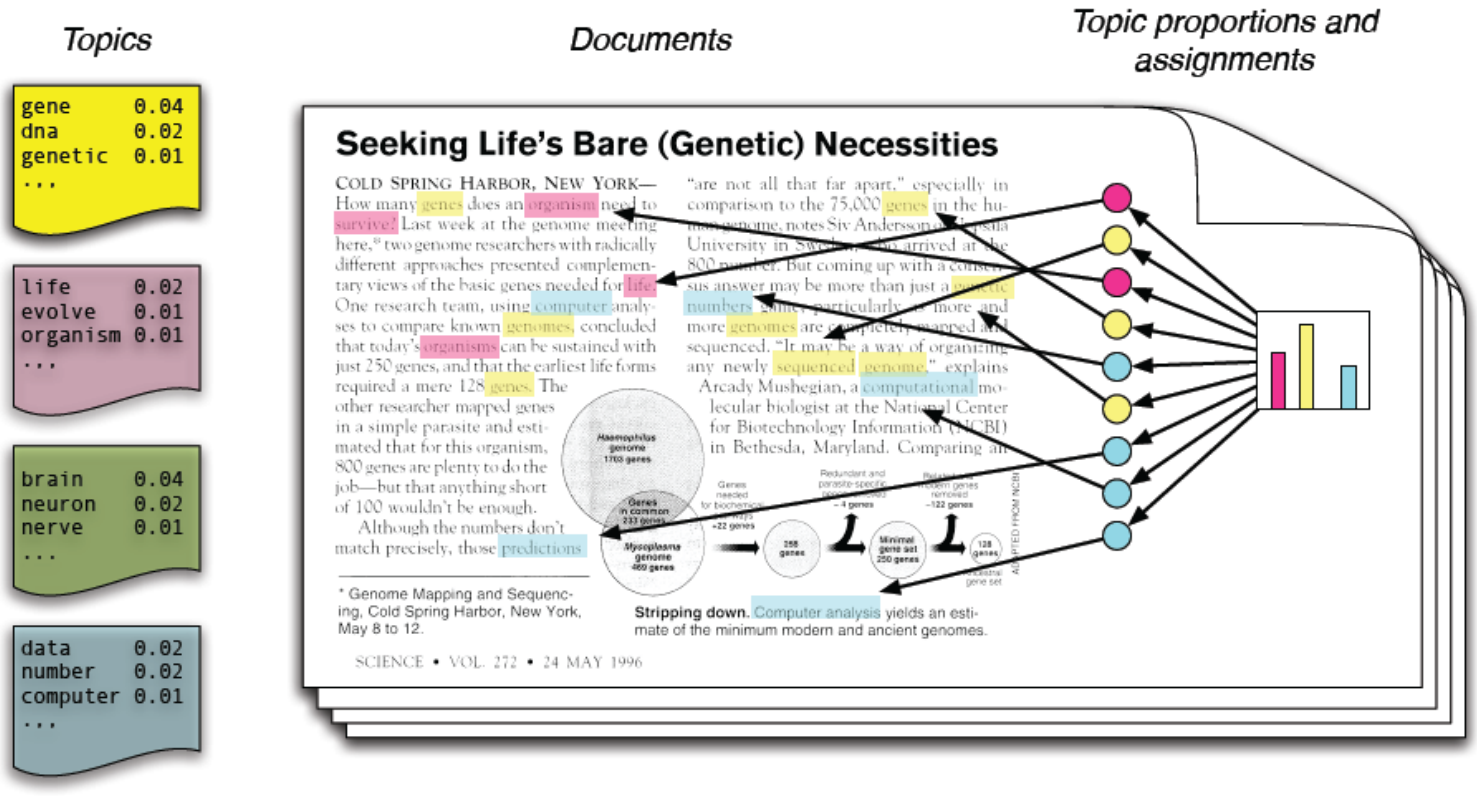


→ denoising

→ different noise perturbations lead to different applications

→ e.g. image completion

Applications of VAEs: Topic modeling



→ posterior on topic proportions parameterized using an encoder

→ linear decoder

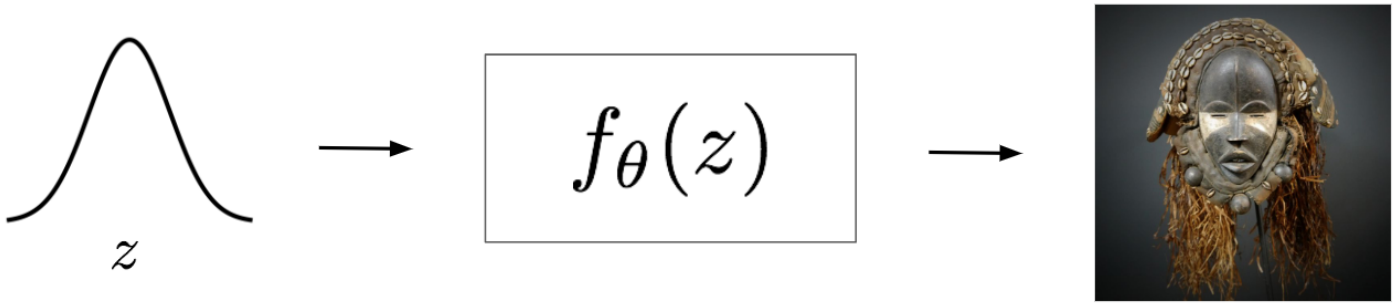
→ fast inference and evaluation

Roadmap

1. Variational Autoencoders
2. **Generative Adversarial Networks**
3. Concluding Remarks

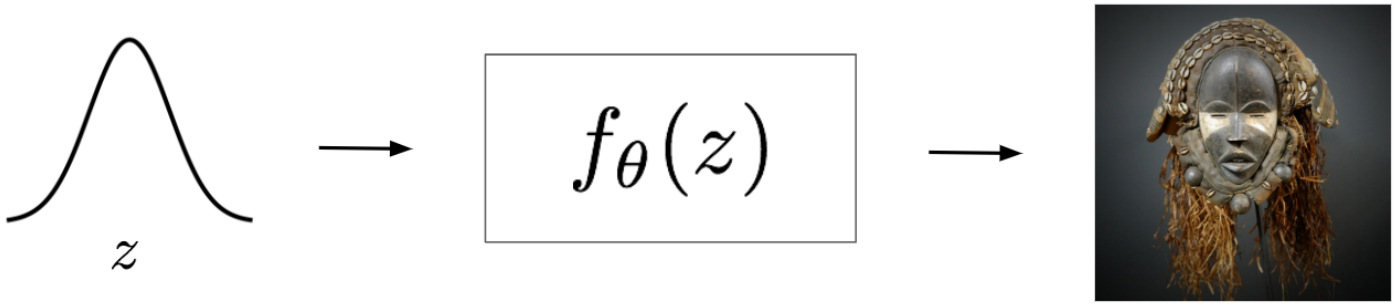
Generative Adversarial Networks

Implicit Models



Generative distribution $p_\theta(\mathbf{x})$ defined only via sampling
In particular, the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ is undefined.

Adversarial Learning



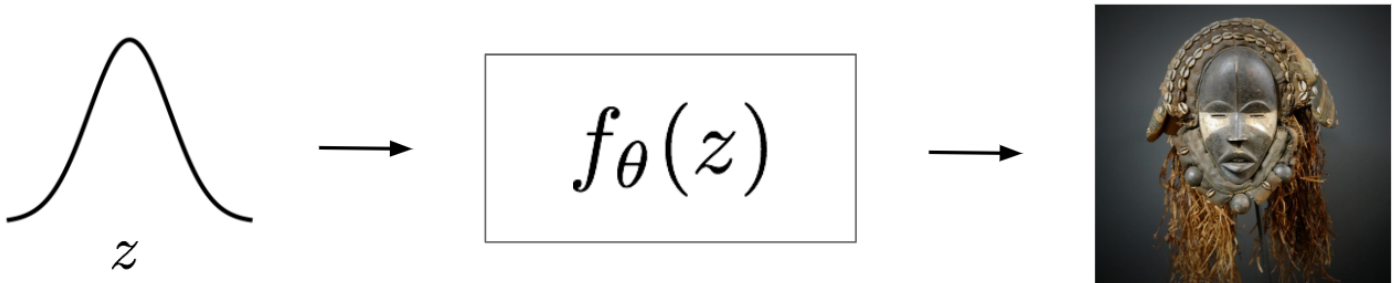
→ Only requires samples from $p_\theta(\mathbf{x})$

→ Introduce a classifier D_ϕ (discriminator)

→ Run a minimax procedure

$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})} \log D_\phi(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \log(1 - D_\phi(f_\theta(\mathbf{z})))$$

Adversarial Learning



$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})} \log D_\phi(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \log(1 - D_\phi(f_\theta(\mathbf{z})))$$

The optimal discriminator for a fixed θ is

$$D_{\phi^*}(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x}) + p_\theta(\mathbf{x})}$$

An Impractical Learning Algorithm

Algorithm 3:

Result: Optimized model parameters θ^*

initialize model parameters θ to θ_0;

for *iteration* $t = 1, 2, \ldots$ **do**

  Find $\phi^* = \arg\max_{\phi} \mathcal{L}(\theta_{t-1}, \phi)$;

  Compute $L(\theta) = \mathbb{E}_{p(\mathbf{z})} \log(1 - D_{\phi^*}(f_\theta(\mathbf{z})))$;

  Update model parameters using SGD $\theta_t := \theta_{t-1} - \rho \cdot \nabla_\theta L(\theta)$

end

- $\rightarrow$ Too costly to run discriminator to convergence
- $\rightarrow$ May lead to overfitting

A Practical Learning Algorithm

Algorithm 4:

Result: Optimized model parameters θ^*

initialize model and discriminator parameters θ, ϕ to θ_0, ϕ_0;

for *iteration* $t = 1, 2, \ldots$ **do**

Sample a minibatch $\mathcal{B}$ containing $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(b)}, \ldots \mathbf{x}^{(|\mathcal{B}|)}$;

Compute $\mathcal{L}(\theta, \phi) = \sum_{b \in \mathcal{B}} \log D_\phi(\mathbf{x}^{(b)}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D_{\phi^*}(f_\theta(\mathbf{z})))$;

Update discriminator parameters using SGD

$\phi_t := \phi_{t-1} + \rho_D \cdot \nabla_\phi \mathcal{L}(\theta_{t-1}, \phi)$;

Update model parameters using SGD $\theta_t := \theta_{t-1} - \rho_G \cdot \nabla_\theta \mathcal{L}(\theta, \phi_t)$

end

$\rightarrow$ May take K steps when updating discriminator...

Divergence Perspective

$$\min_{\theta} \max_{\phi} \mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})} \log D_\phi(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \log(1 - D_\phi(f_\theta(\mathbf{z})))$$

$\rightarrow$ When the discriminator is the optimal one D_{ϕ^*}, then

$$\begin{aligned} \mathcal{L}(\theta, \phi^*) &= \mathrm{KL}\left(p_d(\mathbf{x}) \middle\| \frac{p_d(\mathbf{x}) + p_\theta(\mathbf{x})}{2}\right) + \mathrm{KL}\left(p_\theta(\mathbf{x}) \middle\| \frac{p_d(\mathbf{x}) + p_\theta(\mathbf{x})}{2}\right) \\ &= \mathrm{JS}(p_d(\mathbf{x}) \| p_\theta(\mathbf{x})) \quad \text{Jensen-Shannon divergence} \end{aligned}$$

$\rightarrow$ MLE corresponds to $\mathrm{KL}(p_d(\mathbf{x}) \| p_\theta(\mathbf{x}))$ when $p_\theta(\mathbf{x})$ is well-defined

Mode Collapse

- → GANs learn distributions $p_\theta(\mathbf{x})$ of low support
- → Manifestation: low sample diversity
- → Solution: maximize entropy of the generator

$$\mathcal{H}(p_\theta) = -\mathbb{E}_{p_\theta(\mathbf{x})} \log p_\theta(\mathbf{x})$$

- → Ooops... density $p_\theta(\mathbf{x})$ is unavailable so the entropy is undefined
- → Multiple proposed "fixes" to this problem including:
  - + VeeGAN [Srivastava et al., 2017]
  - + PacGAN [Lin et al., 2017]

Instability and Instance Noise

- → Instability is inherent to minimax procedures
- → $p_d(\mathbf{x})$ and $p_\theta(\mathbf{x})$ might share zero support at the beginning
- → Instance noise as a fix
  - + Add noise to samples from generator:

$$\text{fake sample} = f_\theta(\mathbf{z}) + \sigma \odot \epsilon$$

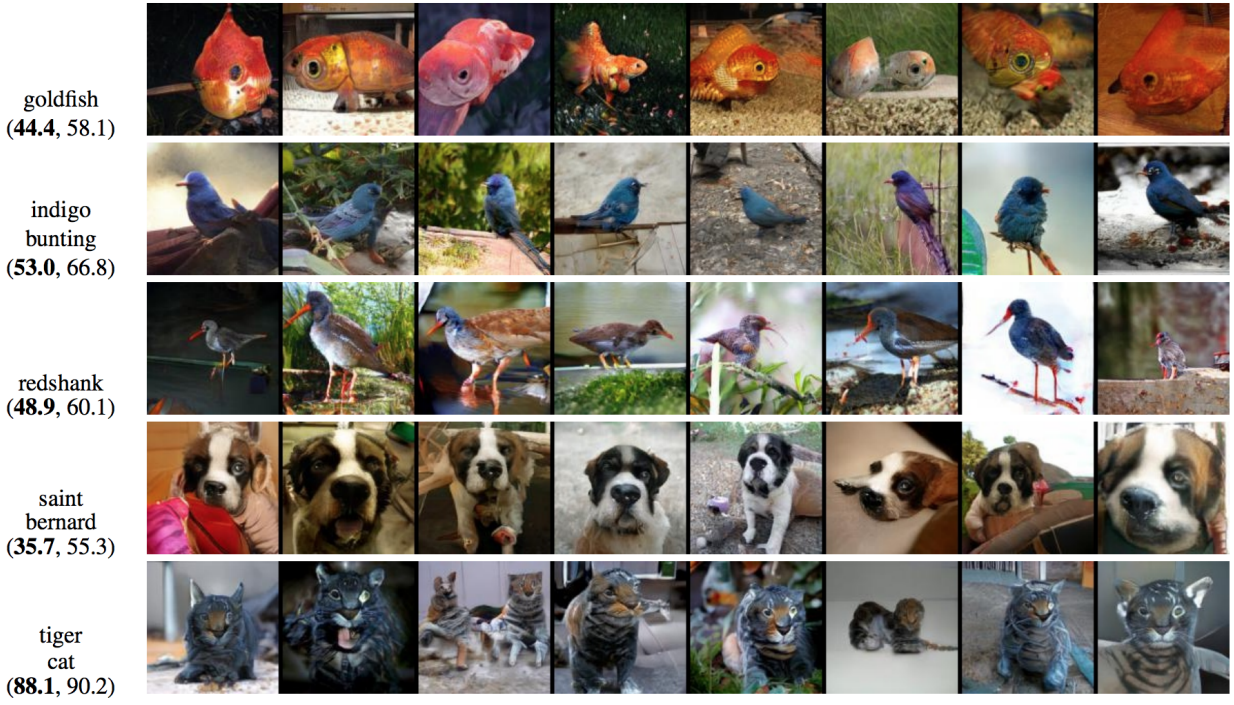
  - + Add noise to real:

$$\text{noised real sample} = \mathbf{x} + \sigma \odot \epsilon$$

  - + Proceed as usual by optimizing the GAN objective
  - + Anneal the noise variance σ to 0

[Sonderby et al., 2017, Arjovsky & Bottou, 2017, Huszar Blog Post].

Applications of GANs: Vision



[Zhang et al., 2017].

→ Image generation, image translation, art

Roadmap

1. Variational Autoencoders
2. Generative Adversarial Networks
3. **Concluding Remarks**

Concluding Remarks

Exciting research area: VAEs

- $\rightarrow$ Improve sample quality for VAEs
- $\rightarrow$ More expressive latent spaces: implicit variational models
- $\rightarrow$ How to perform maximum likelihood effectively?

Exciting research area: GANs

- → Adapt GANs to discrete outputs (e.g. NLP)
  - + GANs are somewhat less successful in NLP
  - + Solution: gumbel approximation, policy gradients
- → Fix mode collapse for GANs
- → How to learn better latent representations for GANs?
- → How to measure log-likelihood for GANs?

Evaluation

- → generalization: held-out log-likelihood

$$\log p_\theta(\mathbf{x}^*) = \log \mathbb{E}_{r(\mathbf{z})} \left[\frac{p_\theta(\mathbf{x}^*|\mathbf{z})p(\mathbf{z})}{r(\mathbf{z})} \right]$$

- → sample quality: Frechet Inception Distances and others
- → sample diversity: ?