CENG 796
Deep Generative Models

Discrete Latent Variable Models & Hybrid Models

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

Discrete Latent Variable Models

Why should we care about discreteness?

- Discreteness is all around us!
- Decision Making: Should I attend CS 236 lecture or not?
- Structure learning

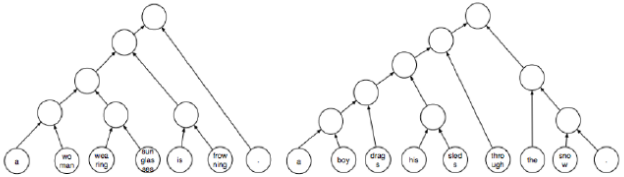


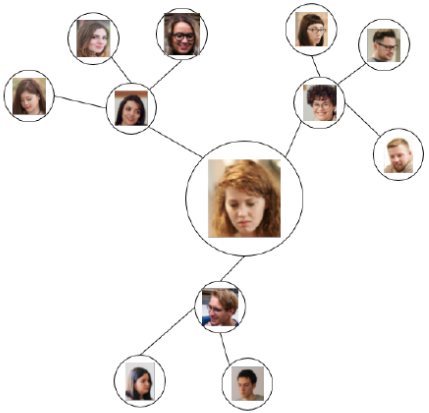
Figure 2: Examples of tree structures learned by our model which show that the model discovers simple concepts such as noun phrases and verb phrases.

Figure 3: Examples of unconventional tree structures.

Source: Yogatama et al., 2017

Why should we care about discreteness?

- Many data modalities are inherently discrete
  - Graphs



  - Text, DNA Sequences, Program Source Code, Molecules, and lots more

Slide by Stefano Ermon, Aditya Grover

Stochastic Optimization

- Consider the following optimization problem

$$\max_{\phi} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

- Recap example: Think of $q(\cdot)$ as the inference distribution for a VAE

$$\max_{\theta,\phi} E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right].$$

- Gradients w.r.t. θ can be derived via linearity of expectation

$$\begin{aligned}\nabla_\theta E_{q(\mathbf{z};\phi)}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q(\mathbf{z};\phi)] &= E_{q(\mathbf{z};\phi)}[\nabla_\theta \log p(\mathbf{z},\mathbf{x};\theta)] \\ &\approx \frac{1}{k}\sum_k \nabla_\theta \log p(\mathbf{z}^k,\mathbf{x};\theta)\end{aligned}$$

- If $\mathbf{z}$ is continuous, $q(\cdot)$ is reparameterizable, and $f(\cdot)$ is differentiable in ϕ, then we can use reparameterization to compute gradients w.r.t. ϕ
- What if any of the above assumptions fail?

Slide by Stefano Ermon, Aditya Grover

Stochastic Optimization with REINFORCE

- Consider the following optimization problem

$$\max_{\phi} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

- For many class of problem scenarios, reparameterization trick is inapplicable

REINFORCE for reinforcement learning

- Example: Pulling arms of slot machines Which arm to pull?
- Set A of possible actions. E.g., pull arm 1, arm 2, . . . , etc.
- Each action $\mathbf{z} \in A$ has a reward $f(\mathbf{z})$
- Randomized policy for choosing actions $q_\phi(\mathbf{z})$ parameterized by ϕ . For example, ϕ could be the parameters of a multinomial distribution
- **Goal:** Learn the parameters ϕ that maximize our earnings (in expectation)

$$\max_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

Slide by Stefano Ermon, Aditya Grover

Policy Gradients

- Want to compute a gradient with respect to ϕ of the expected reward

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

$$\begin{aligned}\frac{\partial}{\partial \phi_i} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] &= \sum_{\mathbf{z}} \frac{\partial q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \frac{1}{q_\phi(\mathbf{z})} \frac{\partial q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \frac{\partial \log q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) = E_{q_\phi(\mathbf{z})}\left[\frac{\partial \log q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z})\right]\end{aligned}$$

Slide by Stefano Ermon, Aditya Grover

REINFORCE Gradient Estimation

- Want to compute a gradient with respect to ϕ of

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

- The REINFORCE rule is

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = E_{q_\phi(\mathbf{z})}\left[f(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})\right]$$

- We can now construct a Monte Carlo estimate
- Sample $\mathbf{z}^1, \cdots, \mathbf{z}^K$ from $q_\phi(\mathbf{z})$ and estimate

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \approx \frac{1}{K} \sum_k f(\mathbf{z}^k) \nabla_\phi \log q_\phi(\mathbf{z}^k)$$

- Assumption: The distribution $q(\cdot)$ is easy to sample from and evaluate probabilities
- Works for both discrete and continuous distributions

Slide by Stefano Ermon, Aditya Grover

Variational Learning of Latent Variable Models

- To learn the variational approximation we need to compute the gradient with respect to ϕ of

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z},\mathbf{x};\theta) + H(q_\phi(\mathbf{z}|\mathbf{x})) \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))]\end{aligned}$$

- The function inside the brackets also depends on ϕ (and $\theta, \mathbf{x}$). Want to compute a gradient with respect to ϕ of

$$E_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\phi,\theta,\mathbf{z},\mathbf{x})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) f(\phi,\theta,\mathbf{z},\mathbf{x})$$

- The REINFORCE rule is

$$\nabla_\phi E_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\phi,\theta,\mathbf{z},\mathbf{x})] = E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[f(\phi,\theta,\mathbf{z},\mathbf{x})\nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}) + \nabla_\phi f(\phi,\theta,\mathbf{z},\mathbf{x})\right]$$

- We can now construct a Monte Carlo estimate of $\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)$

REINFORCE Gradient Estimates have High Variance

- Want to compute a gradient with respect to ϕ of

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

- The REINFORCE rule is

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = E_{q_\phi(\mathbf{z})}\left[f(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})\right]$$

- Monte Carlo estimate: Sample $\mathbf{z}^1, \cdots, \mathbf{z}^K$ from $q_\phi(\mathbf{z})$

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \approx \frac{1}{K} \sum_k f(\mathbf{z}^k) \nabla_\phi \log q_\phi(\mathbf{z}^k) := f_{\mathrm{MC}}(\mathbf{z}^1, \cdots, \mathbf{z}^K)$$

- Monte Carlo estimates of gradients are unbiased

$$E_{\mathbf{z}^1, \cdots, \mathbf{z}^K \sim q_\phi(\mathbf{z})}\left[f_{\mathrm{MC}}(\mathbf{z}^1, \cdots, \mathbf{z}^K)\right] = \nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

- Almost never used in practice because of high variance
- Variance can be reduced via carefully designed control variates

Monte Carlo Gradient Estimation in Machine Learning

Slide by Stefano Ermon, Aditya Grover

REINFORCE Gradient Estimates have High Variance

Our estimation problem.

$$\nabla_\theta E_q[x^2]$$

q(x) distribution. Note that here q is parameterized by *theta*, not *phi*.

$$q_\theta(x) = N(\theta, 1).$$

REINFORCE estimator.
(grad1)

$$E_q[x^2 \nabla_\theta \log q_\theta(x)] = E_q[x^2(x-\theta)]$$

Re-parameterization trick (VAE)
(grad2)

$$x = \theta + \epsilon, \quad \epsilon \sim N(0,1)$$
$$\nabla_\theta E_q[x^2] = \nabla_\theta E_p[(\theta+\epsilon)^2] = E_p[2(\theta+\epsilon)]$$

```
import numpy as np
N = 1000
theta = 2.0
eps = np.random.randn(N)
x = theta + eps

grad1 = lambda x: np.sum(np.square(x)*(x-theta)) / x.size
grad2 = lambda eps: np.sum(2*(theta + eps)) / x.size
```

Content adapted from: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/

REINFORCE Gradient Estimates have High Variance

```
Ns = [10, 100, 1000, 10000, 100000]
reps = 100

means1 = np.zeros(len(Ns))
vars1 = np.zeros(len(Ns))
means2 = np.zeros(len(Ns))
vars2 = np.zeros(len(Ns))

est1 = np.zeros(reps)
est2 = np.zeros(reps)
for i, N in enumerate(Ns):
    for r in range(reps):
        x = np.random.randn(N) + theta
        est1[r] = grad1(x)
        eps = np.random.randn(N)
        est2[r] = grad2(eps)
    means1[i] = np.mean(est1)
    means2[i] = np.mean(est2)
    vars1[i] = np.var(est1)
    vars2[i] = np.var(est2)
```

Content adapted from: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/

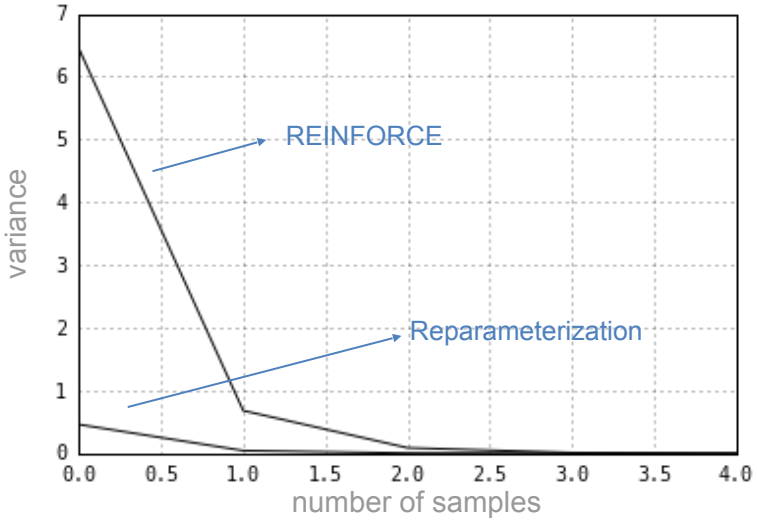
REINFORCE Gradient Estimates have High Variance

```
print means1
print means2
print
print vars1
print vars2
```

```
[ 3.8409546   3.97298803  4.03007634  3.98531095  3.99579423]
[ 3.97775271  4.00232825  3.99894536  4.00353734  3.99995899]

[  6.45307927e+00   6.80227241e-01   8.69226368e-02   1.00489791e-02
   8.62396526e-04]
[  4.59767676e-01   4.26567475e-02   3.33699503e-03   5.17148975e-04
   4.65338152e-05]
```

```
plt.plot(vars1)
plt.plot(vars2)
```



Content adapted from: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/

Towards reparameterized, continuous relaxations

- Consider the following optimization problem

$$\max_{\phi} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

- What if $\mathbf{z}$ is a discrete random variable?
  - Categories
  - Permutations
- Reparameterization trick is not directly applicable
- REINFORCE is a general-purpose solution, but needs careful design of control variates
- **Today:** Relax $\mathbf{z}$ to a continuous random variable with a reparameterizable distribution

Slide by Stefano Ermon, Aditya Grover

Gumbel Distribution

- Setting: We are given i.i.d. samples $y_1, y_2, ..., y_n$ from some underlying distribution How can we model the distribution of

$$g = \max\{y_1, y_2, ..., y_n\}$$

- E.g., predicting maximum water level in a river for a particular river based on historical data to detect flooding
- The **Gumbel distribution** is very useful for modeling extreme, rare events, e.g., natural disasters, finance
- CDF for a Gumbel random variable g is parameterized by a location parameter μ and a scale parameter β

$$F(g; \mu, \beta) = \exp\left(-\exp\left(-\frac{g-\mu}{\beta}\right)\right)$$

- Note: If g is a Gumbel(μ, β) r.v., $-\log g$ is an Exponential(μ, β) r.v. Often, Gumbel r.v. are referred to as doubly exponential r.v.

Categorical Distributions

- Let $\mathbf{z}$ denote a k-dimensional categorical random variable with distribution q parameterized by class probabilities $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_k\}$. We will represent $\mathbf{z}$ as a one-hot vector
- **Gumbel-Max reparameterization trick** for sampling from categorical random variables

$$\mathbf{z} = \text{one\_hot}\left(\arg\max_i (g_i + \log \pi_i)\right)$$

where $g_1, g_2, \ldots, g_k$ are i.i.d. samples drawn from Gumbel$(0, 1)$

- Reparametrizable since randomness is transferred to a fixed Gumbel$(0, 1)$ distribution!
- Problem: $\arg\max$ is non-differentiable w.r.t. $\boldsymbol{\pi}$

Slide by Stefano Ermon, Aditya Grover

Relaxing Categorical Distributions to Gumbel-Softmax

- Gumbel-Max Sampler (non-differentiable w.r.t. $\boldsymbol{\pi}$):

$$\mathbf{z} = \text{one\_hot}\left(\arg\max_i (g_i + \log \boldsymbol{\pi})\right)$$

- **Key idea:** Replace arg max with soft max to get a Gumbel-Softmax random variable $\hat{\mathbf{z}}$
- Ouput of softmax is differentiable w.r.t. $\boldsymbol{\pi}$
- Gumbel-Softmax Sampler (differentiable w.r.t. $\boldsymbol{\pi}$):

$$\hat{\mathbf{z}} = \text{soft}\max_i \left(\frac{g_i + \log \boldsymbol{\pi}}{\tau}\right)$$

where $\tau > 0$ is a tunable parameter referred to as the temperature

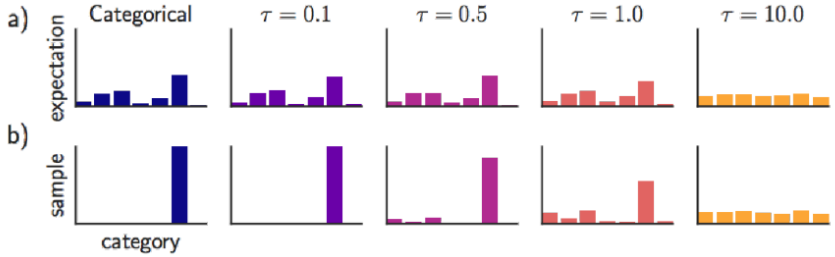
Slide by Stefano Ermon, Aditya Grover

Bias-variance tradeoff via temperature control

- Gumbel-Softmax distribution is parameterized by both class probabilities π and the temperature $\tau > 0$

$$\hat{\mathbf{z}} = \text{soft}\max_i \left(\frac{g_i + \log \boldsymbol{\pi}}{\tau} \right)$$

- Temperature τ controls the degree of the relaxation via a bias-variance tradeoff



Source: Jang et al., 2017

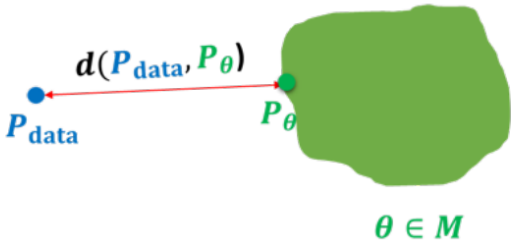
- As $\tau \to 0$, samples from Gumbel-Softmax$(\boldsymbol{\pi}, \tau)$ are similar to samples from Categorical$(\boldsymbol{\pi})$
  Pro: low bias in approximation **Con:** High variance in gradients
- As $\tau \to \infty$, samples from Gumbel-Softmax$(\boldsymbol{\pi}, \tau)$ are similar to samples from Categorical $\left(\left[\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k}\right]\right)$ (i.e., uniform over k categories)

Slide by Stefano Ermon, Aditya Grover

Hybrid Models

Summary

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$

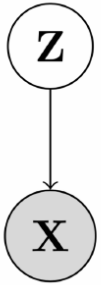


Story so far

- Representation: Latent variable vs. fully observed
- Objective function and optimization algorithm: Many divergences and distances optimized via likelihood-free (two sample test) or likelihood based methods
- Each have Pros and Cons

Plan for today: Combining models

Variational Autoencoder



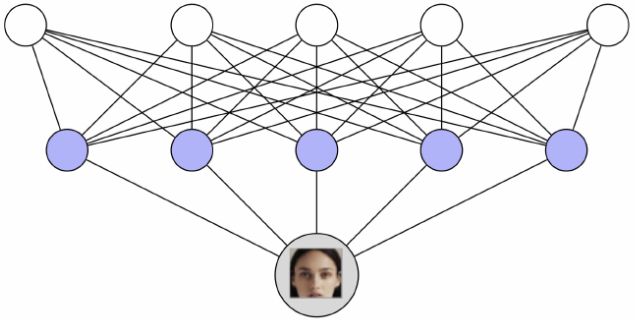


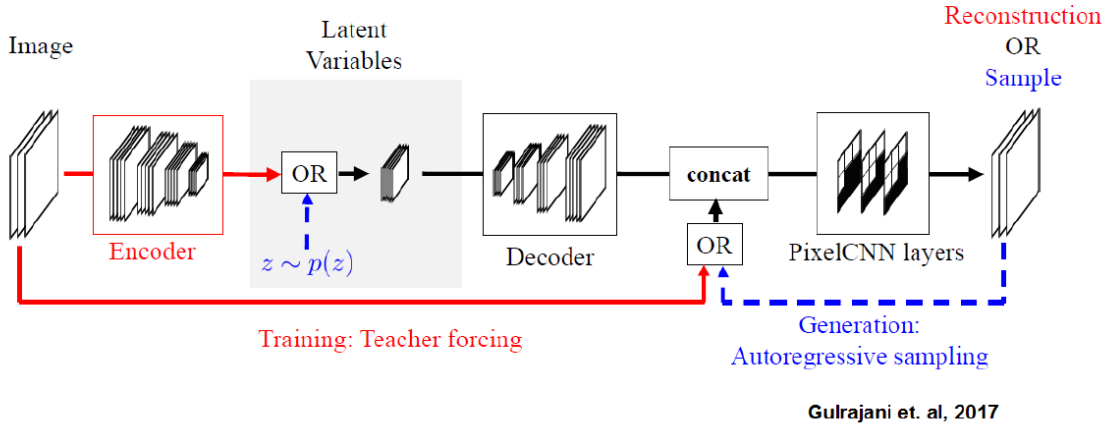
Image x

A mixture of an infinite number of Gaussians:

1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
3. $p(\mathbf{x} \mid \mathbf{z})$ and $p(\mathbf{z})$ usually simple, e.g., Gaussians or conditionally independent Bernoulli vars (i.e., pixel values chosen independently given $\mathbf{z}$)
4. **Idea**: increase complexity using an autoregressive model

Slide by Stefano Ermon, Aditya Grover

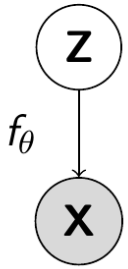
PixelVAE (Gulrajani et al.,2017)



- $\mathbf{z}$ is a feature map with the same resolution as the image $\mathbf{x}$
- Autoregressive structure: $p(\mathbf{x} \mid \mathbf{z}) = \prod_i p(x_i \mid x_1, \cdots, x_{i-1}, \mathbf{z})$
  - $p(\mathbf{x} \mid \mathbf{z})$ is a PixelCNN
  - Prior $p(\mathbf{z})$ can also be autoregressive
  - Can be hierarchical: $p(\mathbf{x} \mid \mathbf{z}_1)p(\mathbf{z}_1 \mid \mathbf{z}_2)$
- State-of-the art log-likelihood on some datasets; learns features (unlike PixelCNN); computationally cheaper than PixelCNN (shallower)

Slide by Stefano Ermon, Aditya Grover

Autoregressive flow

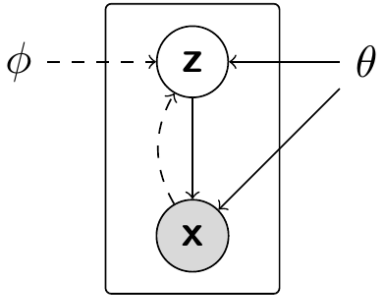


- Flow model, the marginal likelihood $p(\mathbf{x})$ is given by

$$p_X(\mathbf{x};\theta) = p_Z\left(\mathbf{f}_\theta^{-1}(\mathbf{x})\right)\left|\det\left(\frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right)\right|$$

  where $p_Z(\mathbf{z})$ is typically simple (e.g., a Gaussian). More complex prior?
- Prior $p_Z(\mathbf{z})$ can be autoregressive $p_Z(\mathbf{z}) = \prod_i p(z_i \mid z_1, \cdots, z_{i-1})$.
- Autoregressive models are flows. Just another MAF layer.
- See also neural autoregressive flows (Huang et al., ICML-18)

VAE + Flow Model

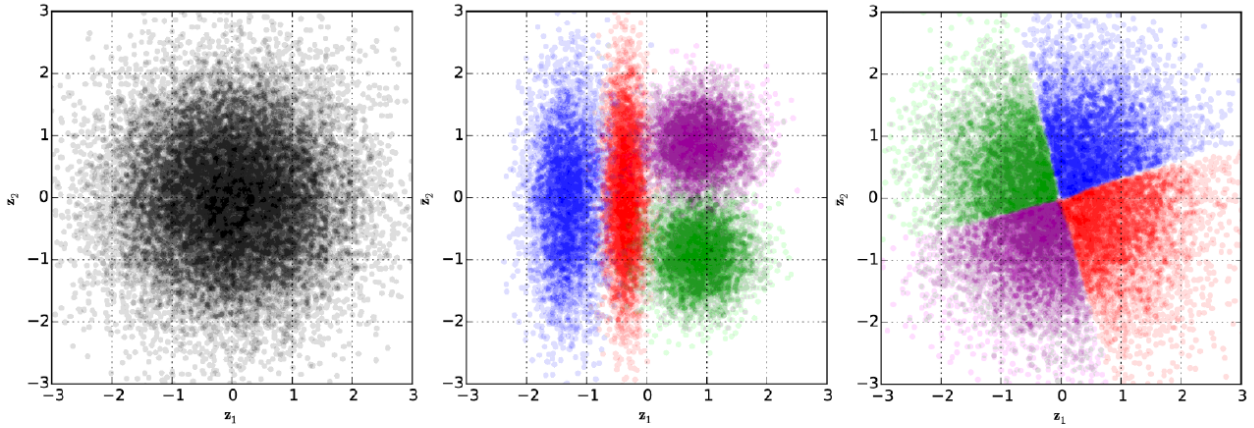


$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z}|\mathbf{x};\phi)) = \underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}}$$

$$\log p(\mathbf{x};\theta) = \mathcal{L}(\mathbf{x};\theta,\phi) + \underbrace{D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))}_{\text{Gap between true log-likelihood and ELBO}}$$

- $q(\mathbf{z}|\mathbf{x};\phi)$ is often too simple (Gaussian) compared to the true posterior $p(\mathbf{z}|\mathbf{x};\theta)$, hence ELBO bound is loose
- **Idea:** Make posterior more flexible: $\mathbf{z}' \sim q(\mathbf{z}'|\mathbf{x};\phi)$, $\mathbf{z} = f_{\phi'}(\mathbf{z}')$ for an invertible $f_{\phi'}$ (Rezende and Mohamed, 2015; Kingma et al., 2016)
- Still easy to sample from, and can evaluate density.

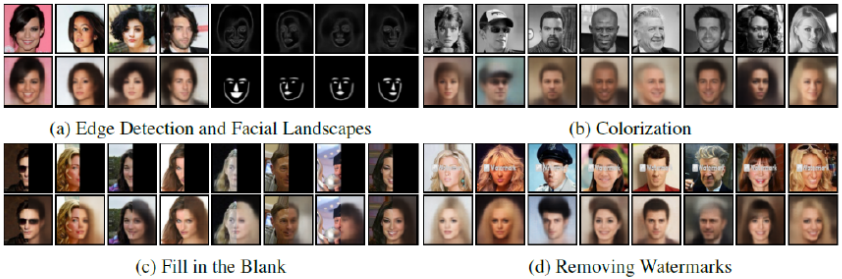
VAE + Flow Model



(a) Prior distribution (b) Posteriors in standard VAE (c) Posteriors in VAE with IAF

Posterior approximation is more flexible, hence we can get tighter ELBO (closer to true log-likelihood).

Multimodal variants

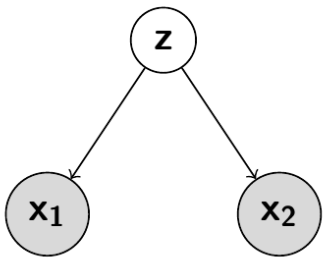


(a) Edge Detection and Facial Landscapes (b) Colorization

(c) Fill in the Blank (d) Removing Watermarks

Wu and Goodman, 2018

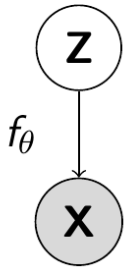
- **Goal:** Learn a joint distribution over the two domains $p(x_1, x_2)$, e.g., color and gray-scale images Can use a VAE style model:



- Learn $p_\theta(x_1, x_2)$, use inference nets $q_\phi(z \mid x_1)$, $q_\phi(z \mid x_2)$, $q_\phi(z \mid x_1, x_2)$.

Slide by Stefano Ermon, Aditya Grover

Combining losses



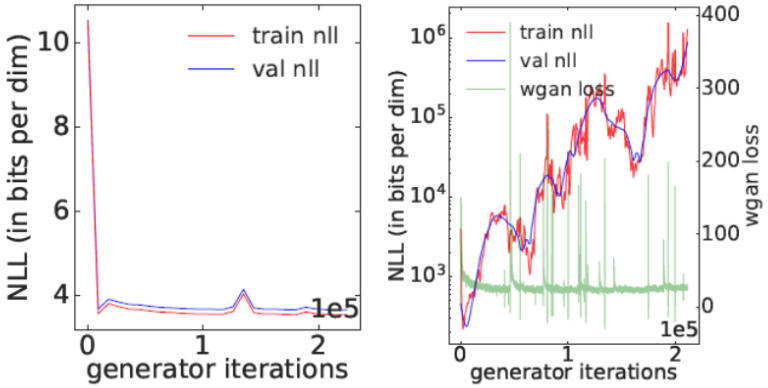
- Flow model, the marginal likelihood $p(\mathbf{x})$ is given by

$$p_X(\mathbf{x};\theta) = p_Z\left(\mathbf{f}_\theta^{-1}(\mathbf{x})\right)\left|\det\left(\frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right)\right|$$

- Can also be thought of as the generator of a GAN
- Should we train by $\min_\theta D_{KL}(p_{data}, p_\theta)$ or $\min_\theta JSD(p_{data}, p_\theta)$?

Slide by Stefano Ermon, Aditya Grover

FlowGAN



(a) MLE (b) ADV

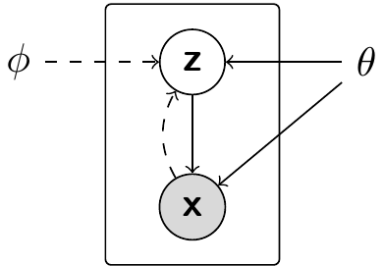
Although $D_{KL}(p_{data}, p_\theta) = 0$ if and only if $JSD(p_{data}, p_\theta) = 0$, optimizing one does not necessarily optimize the other. If $\mathbf{z}, \mathbf{x}$ have same dimensions, can optimize $\min_\theta KL(p_{data}, p_\theta) + \lambda JSD(p_{data}, p_\theta)$

Objective	Inception Score	Test NLL (in bits/dim)
MLE	2.92	**3.54**
ADV	**5.76**	8.53
Hybrid ($\lambda = 1$)	3.90	4.21

Interpolates between a GAN and a flow model

Slide by Stefano Ermon, Aditya Grover

Adversarial Autoencoder (VAE + GAN)



$$\log p(\mathbf{x};\theta) = \underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}} + D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))$$

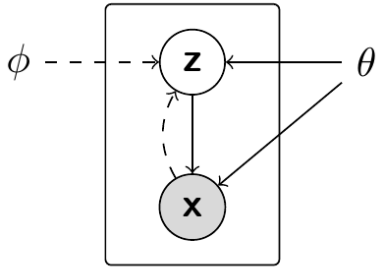
$$\underbrace{E_{\mathbf{x}\sim p_{data}}[\mathcal{L}(\mathbf{x};\theta,\phi)]}_{\approx\text{training obj.}} = E_{\mathbf{x}\sim p_{data}}\left[\log p(\mathbf{x};\theta) - D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right]$$

$$\overset{\text{up to const.}}{\equiv} \underbrace{-D_{KL}(p_{data}(\mathbf{x})\|p(\mathbf{x};\theta))}_{\text{equiv. to MLE}} - E_{\mathbf{x}\sim p_{data}}\left[D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right]$$

- Note: regularized maximum likelihood estimation (Shu et al, *Amortized inference regularization*)
- Can add in a GAN objective $-JSD(p_{data}, p(\mathbf{x};\theta))$ to get sharper samples, i.e., discriminator attempting to distinguish VAE samples from real ones.

Slide by Stefano Ermon, Aditya Grover

An alternative interpretation



$$\underbrace{E_{\mathbf{x}\sim p_{data}}[\mathcal{L}(\mathbf{x};\theta,\phi)]}_{\approx\text{training obj.}} = E_{\mathbf{x}\sim p_{data}}\left[\log p(\mathbf{x};\theta) - D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right]$$

$$\overset{\text{up to const.}}{\equiv} -D_{KL}(p_{data}(\mathbf{x})\|p(\mathbf{x};\theta)) - E_{\mathbf{x}\sim p_{data}}\left[D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right]$$

$$= -\sum_{\mathbf{x}} p_{data}(\mathbf{x})\left(\log\frac{p_{data}(\mathbf{x})}{p(\mathbf{x};\theta)} + \sum_{\mathbf{z}} q(\mathbf{z}\mid\mathbf{x};\phi)\log\frac{q(\mathbf{z}\mid\mathbf{x};\phi)}{p(\mathbf{z}|\mathbf{x};\theta)}\right)$$

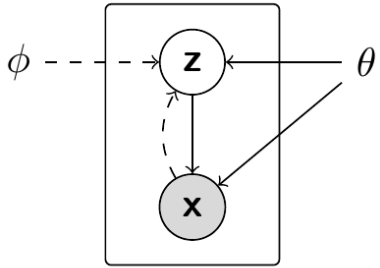
$$= -\sum_{\mathbf{x}} p_{data}(\mathbf{x})\left(\sum_{\mathbf{z}} q(\mathbf{z}\mid\mathbf{x};\phi)\log\frac{q(\mathbf{z}\mid\mathbf{x};\phi)p_{data}(\mathbf{x})}{p(\mathbf{z}|\mathbf{x};\theta)p(\mathbf{x};\theta)}\right)$$

$$= -\sum_{\mathbf{x},\mathbf{z}} p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)\log\frac{p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)}{p(\mathbf{x};\theta)p(\mathbf{z}|\mathbf{x};\theta)}$$

$$= -D_{KL}(\underbrace{p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)}_{q(\mathbf{z},\mathbf{x};\phi)}\|\underbrace{p(\mathbf{x};\theta)p(\mathbf{z}|\mathbf{x};\theta)}_{p(\mathbf{z},\mathbf{x};\theta)})$$

Slide by Stefano Ermon, Aditya Grover

An alternative interpretation



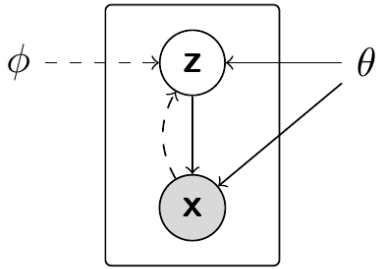
$$E_{\mathbf{x}\sim p_{data}}[\underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}}] \equiv -D_{KL}(\underbrace{p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)}_{q(\mathbf{z},\mathbf{x};\phi)} \| \underbrace{p(\mathbf{x};\theta)p(\mathbf{z}|\mathbf{x};\theta)}_{p(\mathbf{z},\mathbf{x};\theta)})$$

- Optimizing ELBO is the same as matching the inference distribution $q(\mathbf{z},\mathbf{x};\phi)$ to the generative distribution $p(\mathbf{z},\mathbf{x};\theta) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z};\theta)$
- **Intuition**: $p(\mathbf{x};\theta)p(\mathbf{z}|\mathbf{x};\theta) = p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)$ if
  1. $p_{data}(\mathbf{x}) = p(\mathbf{x};\theta)$
  2. $q(\mathbf{z}\mid\mathbf{x};\phi) = p(\mathbf{z}|\mathbf{x};\theta)$ for all $\mathbf{x}$
  3. Hence we get the VAE objective:
     $-D_{KL}(p_{data}(\mathbf{x})\|p(\mathbf{x};\theta)) - E_{\mathbf{x}\sim p_{data}}\left[D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right]$
- Many other variants are possible! VAE + GAN:

$$-JSD(p_{data}(\mathbf{x})\|p(\mathbf{x};\theta)) - D_{KL}(p_{data}(\mathbf{x})\|p(\mathbf{x};\theta)) - E_{\mathbf{x}\sim p_{data}}\left[D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right]$$

Slide by Stefano Ermon, Aditya Grover

Adversarial Autoencoder (VAE + GAN)



$$E_{\mathbf{x}\sim p_{data}}[\underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}}] \equiv -D_{KL}(\underbrace{p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)}_{q(\mathbf{z},\mathbf{x};\phi)} \| \underbrace{p(\mathbf{x};\theta)p(\mathbf{z}|\mathbf{x};\theta)}_{p(\mathbf{z},\mathbf{x};\theta)})$$

- Optimizing ELBO is the same as matching the inference distribution $q(\mathbf{z},\mathbf{x};\phi)$ to the generative distribution $p(\mathbf{z},\mathbf{x};\theta)$
- **Symmetry:** Using alternative factorization: $p(\mathbf{z})p(\mathbf{x}|\mathbf{z};\theta) = q(\mathbf{z};\phi)q(\mathbf{x}\mid\mathbf{z};\phi)$ if
  1. $q(\mathbf{z};\phi) = p(\mathbf{z})$
  2. $q(\mathbf{x}\mid\mathbf{z};\phi) = p(\mathbf{x}|\mathbf{z};\theta)$ for all $\mathbf{z}$
  3. We get an *equivalent* form of the VAE objective: $-D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z})) - E_{\mathbf{z}\sim q(\mathbf{z};\phi)}\left[D_{KL}(q(\mathbf{x}\mid\mathbf{z};\phi)\|p(\mathbf{x}|\mathbf{z};\theta))\right]$
- Other variants are possible. For example, can add $-JSD(q(\mathbf{z};\phi)\|p(\mathbf{z}))$ to match features in latent space (Zhao et al., 2017; Makhzani et al, 2018)

Slide by Stefano Ermon, Aditya Grover

Conclusion

- We have covered several useful building blocks: autoregressive, latent variable models, flow models, GANs.
- Can be combined in many ways to achieve different tradeoffs: many of the models we have seen today were published in top ML conferences in the last couple of years
- Lots of room for exploring alternatives in your projects!
- Which one is best? Evaluation is tricky. Still largely empirical