Combining Generative Model Families

Volodymyr Kuleshov

Cornell Tech

Lecture 14

Announcements

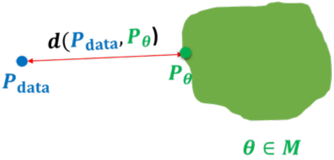
- Assignments 1 and 2 have been graded in Gradescope. Come see me if you have questions. Feel free to submit regarde requests.
- Will post Assignment solutions on Canvas. Stay tuned!
- Project proposals have been graded. Please make sure to read my feedback.
- Assignment 3 is due on Sunday

Lecture Outline

1. Combining Model Families
   - Autoregressive models + VAEs: PixelVAE
   - Autoregressive models + Flows: Autoregressive flows
   - Flows + VAEs: Flow-based posteriors
   - VAEs + RNNs: Variational RNNs
   - Flows + GANs: FlowGAN
   - GANs + VAEs: InfoGAN, InfoVAE, β-VAE

Summary

$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \dots, n$



Model family

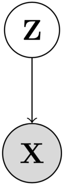
Story so far:

- 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.

Recall: Deep Gaussian Latent Variable Models (VAEs)

We may form deep latent variable models by parameterizing the $\mathbf{z} \to \mathbf{x}$ mapping using neural nets. Deep Gaussian models are an example:



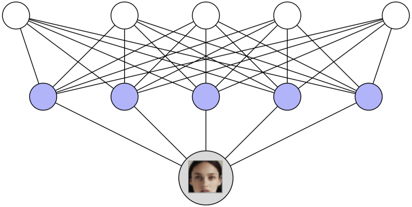


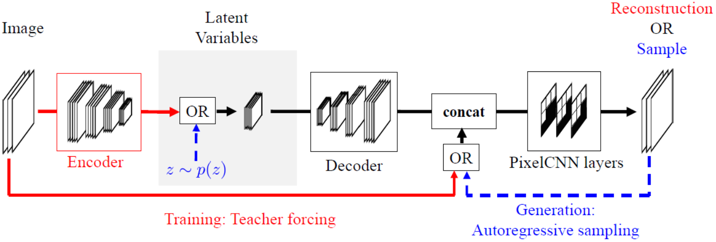
Image x

1. The prior $p(\mathbf{z})$ is a p-dimensional Gaussian $\mathcal{N}(0, I_p)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$ where μ_θ,Σ_θ are neural networks

The **z** form a smooth parameterization of faces **x**. Similar faces (same age, gender) have similar **z**. We can map **x** to **z**, interpolate between **z**, generate faces from new **z**, etc.

PixelVAE (Gulrajani et al.,2017)

The PixelVAE model combines autoregressive PixelCNNs and VAEs:



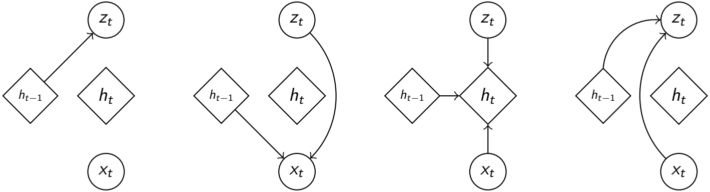
Gulrajani et. al, 2017

- $\mathbf{z}$ is transformed into a feature map with the same resolution as $\mathbf{x}$
- 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
- State-of-the art log-likelihood on some datasets; learns features (unlike PixelCNN); computationally cheaper than PixelCNN (shallower)

Variational RNN

- **Goal:** Learn a joint distribution over a sequence $p(x_1, \cdots, x_T)$
- VAE for sequential data, using latent variables $z_1, \cdots, z_T$. Instead of training separate VAEs $z_i \to x_i$, train a joint model:

$$p(x_{\leq T}, z_{\leq T}) = \prod_{t=1}^{T} p(x_t \mid z_{\leq t}, x_{<t}) p(z_t \mid z_{<t}, x_{<t})$$



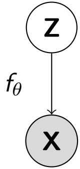
(a) Prior (b) Generation (c) Recurrence (d) Inference

Chung et al, 2016

- Use RNNs to model the conditionals (similar to PixelRNN)
- Use RNNs for inference $q(z_{\leq T} | x_{\leq T}) = \prod_{t=1}^{T} q(z_t \mid z_{<t}, x_{\leq t})$
- Train like VAE to maximize ELBO. Conceptually similar to PixelVAE.

Recall: Autoregressive Flows

Flows and autoregressive models are also closely related.



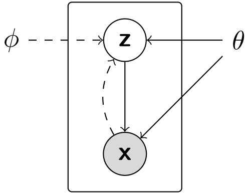
- Flow model, the marginal likelihood $p(\mathbf{x})$ is given by

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

  where $p_Z(\mathbf{z})$ is usually simple (e.g., 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 related to flows. Just another MAF layer.
- Autoregressive flow models can be naturally stacked.

VAE + Flow Model

Flows can improve the expressivity of variational posteriors in VAEs.



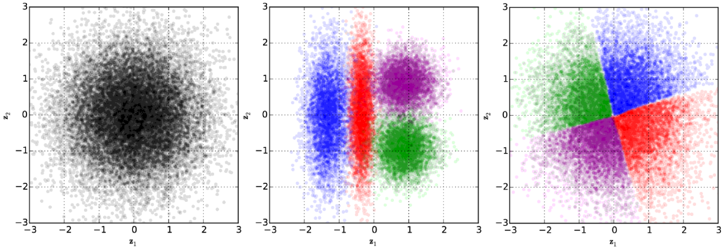
$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x};\phi) \log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z}|\mathbf{x};\phi)) = \underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}}$$

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

- $q(\mathbf{z}|\mathbf{x};\phi)$ 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

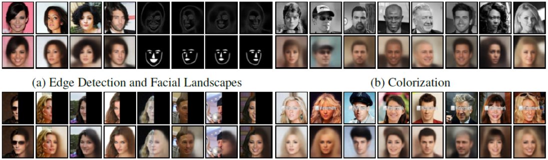
Consider a dataset with four datapoints, shown in different colors below.



- The VAE has a Gaussian prior (left figure).
- The approximate posterior $q(\mathbf{z}|\mathbf{x})$ is also Gaussian. Hence each colored "cloud" of samples for each datapoint $\mathbf{x}$ is also Gaussian (middle).
- Flows make $q(\mathbf{z}|\mathbf{x})$ non-Gaussian. Hence, they fit the prior better.
- Posterior approximation is more flexible, hence we can get tighter ELBO (closer to true log-likelihood).

Multimodal VAEs

Often, data features different modalities ("views"); we may fit these jointly.

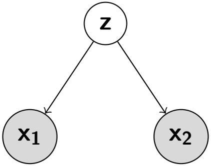


(a) Edge Detection and Facial Landscapes (b) Colorization

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

Wu and Goodman, 2018

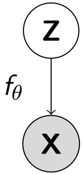
- **Goal:** Learn a joint distribution over the two domains $p(x_1, x_2)$, e.g., color and gray-scale images Can use a VAE style model:



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

Combining Losses

Consider a standard flow model $\mathbf{x} = f_\theta(\mathbf{z})$.

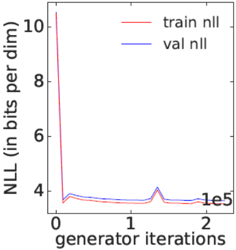


- The marginal likelihood $p(\mathbf{x})$ is given by

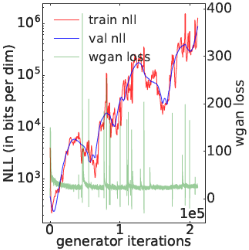
$$p_X(\mathbf{x};\theta) = p_Z\left(\mathbf{f}_\theta^{-1}(\mathbf{x})\right)\left|\det\left(\frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right)\right|.$$

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

FlowGAN



(a) MLE



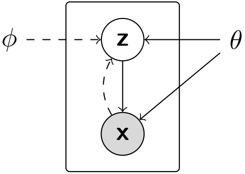
(b) ADV

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

Combining VAEs + GANs



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

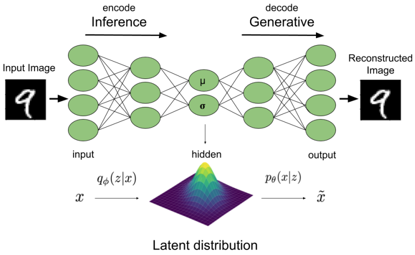
$$\underbrace{E_{\mathbf{x}\sim p_{data}}[\mathcal{L}(\mathbf{x};\theta,\phi)]}_{\approx\text{training obj.}} = E_{\mathbf{x}\sim p_{data}}\left[\log p(\mathbf{x};\theta) - D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right]$$

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

- 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.

Adversarial Autoencoders

Recall the autoencoder perspective on VAEs:



- VAE are latent models $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$ that optimize the ELBO

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}; \theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

- We may also match features using the JSD instead of the KLD. Matching $p(z)$ and $q_\phi(\mathbf{z}|\mathbf{x})$ is done in a GAN-style manner.

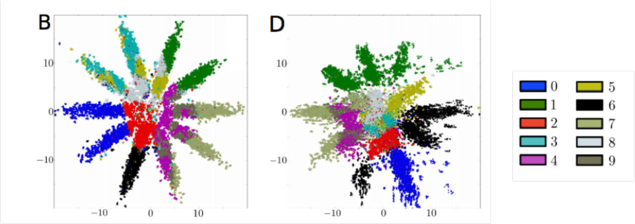
$$\mathcal{L}(\mathbf{x}; \theta, \phi) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}; \theta)] - D_{JSD}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

- This yields adversarial auto-encoders.

Adversarial Autoencoders

Adversarial autoencoders can produce better match the latent priors.

- In this example, the prior is a mixture of 10 Gaussians. Left figure shows an AAE posterior; right figure shows a VAE posterior.



Source: Makhzani et al., 2018

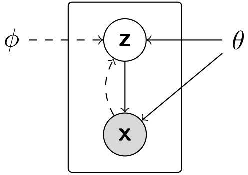
- Adversarial autoencoders don't need to assume $q(z|x)$ is Gaussian
- They can also better match the true shape of the prior (imagine it being an image).

A Reformulation of the ELBO

Next, consider the following reformulation of the ELBO:

$$
\begin{aligned}
\underbrace{E_{\mathbf{x}\sim p_{data}}[\mathcal{L}(\mathbf{x};\theta,\phi)]}_{\approx\text{training obj.}} &= E_{\mathbf{x}\sim p_{data}}\left[\log p(\mathbf{x};\theta) - D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right] \\
&\overset{\text{up to const.}}{\equiv} -D_{KL}(p_{data}(\mathbf{x})\|p(\mathbf{x};\theta)) - E_{\mathbf{x}\sim p_{data}}\left[D_{KL}(q(\mathbf{z}\mid\mathbf{x};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))\right] \\
&= -\sum_{\mathbf{x}} p_{data}(\mathbf{x})\left(\log\frac{p_{data}(\mathbf{x})}{p(\mathbf{x};\theta)} + \sum_{\mathbf{z}} q(\mathbf{z}\mid\mathbf{x};\phi)\log\frac{q(\mathbf{z}\mid\mathbf{x};\phi)}{p(\mathbf{z}|\mathbf{x};\theta)}\right) \\
&= -\sum_{\mathbf{x}} p_{data}(\mathbf{x})\left(\sum_{\mathbf{z}} q(\mathbf{z}\mid\mathbf{x};\phi)\log\frac{q(\mathbf{z}\mid\mathbf{x};\phi)p_{data}(\mathbf{x})}{p(\mathbf{z}|\mathbf{x};\theta)p(\mathbf{x};\theta)}\right) \\
&= -\sum_{\mathbf{x},\mathbf{z}} p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)\log\frac{p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)}{p(\mathbf{x};\theta)p(\mathbf{z}|\mathbf{x};\theta)} \\
&= -D_{KL}(\underbrace{p_{data}(\mathbf{x})q(\mathbf{z}\mid\mathbf{x};\phi)}_{q(\mathbf{z},\mathbf{x};\phi)} \,\|\, \underbrace{p(\mathbf{x};\theta)p(\mathbf{z}|\mathbf{x};\theta)}_{p(\mathbf{z},\mathbf{x};\theta)})
\end{aligned}
$$

Ignoring Latent Variables: A VAE Failure Mode



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

- ELBO is optimized as long as $q(\mathbf{z},\mathbf{x};\phi) = p(\mathbf{z},\mathbf{x};\theta)$. Many solutions are possible! For example,
  1. $p(\mathbf{z},\mathbf{x};\theta) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z};\theta) = p(\mathbf{z})p_{data}(\mathbf{x})$
  2. $q(\mathbf{z},\mathbf{x};\phi) = p_{data}(\mathbf{x})q(\mathbf{z}|\mathbf{x};\phi) = p_{data}(\mathbf{x})p(\mathbf{z})$
  3. Note: $\mathbf{x}$ and $\mathbf{z}$ are independent. $\mathbf{z}$ carries no information about $\mathbf{x}$. This happens in practice when $p(\mathbf{x}|\mathbf{z};\theta)$ is too flexible, like PixelCNN.
- **Issue:** Many more variables than constraints

InfoVAE and InfoGAN

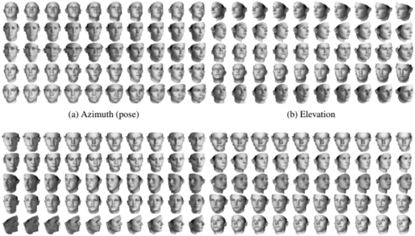
- Explicitly add a mutual information term to the objective

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

- MI intuitively measures how far $\mathbf{x}$ and $\mathbf{z}$ are from being independent

$$MI(\mathbf{x},\mathbf{z}) = D_{KL}\left(p(\mathbf{z},\mathbf{x};\theta) \| p(\mathbf{z})p(\mathbf{x};\theta)\right)$$

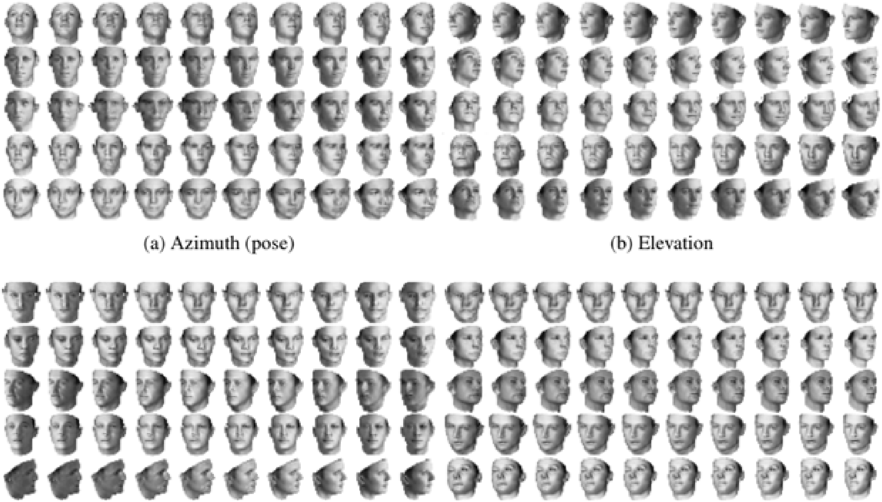
- InfoGAN (Chen et al, 2016) use mutual information to learn meaningful (disentangled) representations of the data



(a) Azimuth (pose) (b) Elevation

(c) Lighting (d) Wide or Narrow

InfoGAN



(a) Azimuth (pose)

(b) Elevation

(c) Lighting

(d) Wide or Narrow

β-VAE

Model proposed to learn disentangled features (Higgins, 2016)

$$-E_{q_\phi(\mathbf{x},\mathbf{z})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + \beta E_{\mathbf{x}\sim p_{data}}\left[D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\right]$$

It is a VAE with scaled up KL divergence term. This is equivalent (up to constants) to the following objective:

$$(\beta - 1)MI(\mathbf{x};\mathbf{z}) + \beta D_{KL}(q_\phi(\mathbf{z})\|p(\mathbf{z})) + E_{q_\phi(\mathbf{z})}[D_{KL}(q_\phi(\mathbf{x}|\mathbf{z})\|p_\theta(\mathbf{x}|\mathbf{z}))]$$

See *The Information Autoencoding Family: A Lagrangian Perspective on Latent Variable Generative Models* for more examples.

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