Variants and Combinations of Basic Models

Stefano Ermon, Yang Song

Stanford University

Lecture 16
Summary

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 (KL divergence)
- Each have Pros and Cons

Plan for today: Combining models
Variational Autoencoder

A mixture of an infinite number of Gaussians:

1. \( z \sim \mathcal{N}(0, I) \)
2. \( p(x \mid z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z)) \) where \( \mu_\theta, \Sigma_\theta \) are neural networks
3. \( p(x \mid z) \) and \( p(z) \) usually simple, e.g., Gaussians or conditionally independent Bernoulli vars (i.e., pixel values chosen independently given \( z \))
4. Idea: increase complexity using an autoregressive model
PixelVAE (Gulrajani et al., 2017)

- **z** is a feature map with the same resolution as the image **x**
- Autoregressive structure: \( p(\mathbf{x} \mid \mathbf{z}) = \prod_i p(x_i \mid x_1, \ldots, x_{i-1}, \mathbf{z}) \)
  - \( p(\mathbf{x} \mid \mathbf{z}) \) is a PixelCNN
  - Prior \( p(\mathbf{z}) \) can also be autoregressive
- Learns features (unlike PixelCNN); computationally cheaper than PixelCNN (shallower)
Flow model, the marginal likelihood $p(x)$ is given by

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

where $p_Z(z)$ is typically simple (e.g., a Gaussian). More complex prior?

Prior $p_Z(z)$ can be autoregressive $p_Z(z) = \prod_i p(z_i \mid z_1, \cdots, z_{i-1})$.

Autoregressive models are flows. Just another MAF layer.

See also neural autoregressive flows (Huang et al., ICML-18)
\[
\begin{align*}
\log p(x; \theta) & \geq \sum_z q(z|x; \phi) \log p(z, x; \theta) + H(q(z|x; \phi)) = \mathcal{L}(x; \theta, \phi) \\
\log p(x; \theta) & = \mathcal{L}(x; \theta, \phi) + \underbrace{D_{KL}(q(z|x; \phi)\| p(z|x; \theta))}_{\text{Gap between true log-likelihood and ELBO}}
\end{align*}
\]

- \(q(z|x; \phi)\) is often too simple (Gaussian) compared to the true posterior \(p(z|x; \theta)\), hence ELBO bound is loose
- **Idea:** Make posterior more flexible: \(z' \sim q(z'|x; \phi), z = f_{\phi'}(z')\) for an invertible \(f_{\phi'}\) (Rezende and Mohamed, 2015; Kingma et al., 2016)
- Still easy to sample from, and can evaluate density.
Posterior approximation is more flexible, hence we can get tighter ELBO (closer to true log-likelihood).
**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 \( q_\phi(z \mid x_1) \), \( q_\phi(z \mid x_2) \), \( q_\phi(z \mid x_1, x_2) \). Conceptually similar to semi-supervised VAE in HW2.
**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 $z_i \rightarrow x_i$, train a joint model:

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

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

Flow model, the marginal likelihood $p(x)$ is given by

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

Can also be thought of as the generator of a GAN

Should we train by $\min_\theta D_{KL}(p_{data}, p_\theta)$ or $\min_\theta JSD(p_{data}, p_\theta)$?
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 $z, x$ have same dimensions, can optimize $\min_{\theta} KL(p_{data}, p_{\theta}) + \lambda JSD(p_{data}, p_{\theta})$

<table>
<thead>
<tr>
<th>Objective</th>
<th>Inception Score</th>
<th>Test NLL (in bits/dim)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>2.92</td>
<td>3.54</td>
</tr>
<tr>
<td>ADV (\lambda = 1)</td>
<td>5.76</td>
<td>8.53</td>
</tr>
<tr>
<td>Hybrid (\lambda = 1)</td>
<td>3.90</td>
<td>4.21</td>
</tr>
</tbody>
</table>

Interpolates between a GAN and a flow model
Adversarial Autoencoder (VAE + GAN)

\[
\log p(x; \theta) = \mathcal{L}(x; \theta, \phi) + D_{KL}(q(z | x; \phi) \| p(z|x; \theta))
\]

\[
E_{x \sim p_{data}}[\mathcal{L}(x; \theta, \phi)] \approx \text{training obj.}
\]

\[
\approx \text{training obj.} \equiv -D_{KL}(p_{data}(x) \| p(x; \theta)) - E_{x \sim p_{data}}[D_{KL}(q(z | x; \phi) \| p(z|x; \theta))]
\]

- Note: regularized maximum likelihood estimation (Shu et al, *Amortized inference regularization*)

- Can add in a GAN objective \(-\text{JSD}(p_{data}, p(x; \theta))\) to get sharper samples, i.e., discriminator attempting to distinguish VAE samples from real ones.
An alternative interpretation

\[ E_{x \sim p_{\text{data}}} \left[ \mathcal{L}(x; \theta, \phi) \right] \approx \text{training obj.} \]

\[ = E_{x \sim p_{\text{data}}} \left[ \log p(x; \theta) - D_{KL}(q(z \mid x; \phi) \parallel p(z \mid x; \theta)) \right] \]

up to const.

\[ \equiv -D_{KL}(p_{\text{data}}(x) \parallel p(x; \theta)) - E_{x \sim p_{\text{data}}} \left[ D_{KL}(q(z \mid x; \phi) \parallel p(z \mid x; \theta)) \right] \]

\[ = - \sum_x p_{\text{data}}(x) \left( \log \frac{p_{\text{data}}(x)}{p(x; \theta)} + \sum_z q(z \mid x; \phi) \log \frac{q(z \mid x; \phi)}{p(z \mid x; \theta)} \right) \]

\[ = - \sum_x p_{\text{data}}(x) \left( \sum_z q(z \mid x; \phi) \log \frac{q(z \mid x; \phi) p_{\text{data}}(x)}{p(z \mid x; \theta) p(x; \theta)} \right) \]

\[ = - \sum_{x,z} p_{\text{data}}(x) q(z \mid x; \phi) \log \frac{p_{\text{data}}(x) q(z \mid x; \phi)}{p(x; \theta) p(z \mid x; \theta)} \]

\[ = -D_{KL}(p_{\text{data}}(x) q(z \mid x; \phi) \parallel p(x; \theta) p(z \mid x; \theta)) \]
An alternative interpretation

\[ E_{x \sim p_{data}} [\mathcal{L}(x; \theta, \phi)] \equiv -D_{KL}(p_{data}(x)q(z \mid x; \phi) \parallel p(x; \theta)p(z \mid x; \theta)) \]

- Optimizing ELBO is the same as matching (via KL) the inference distribution \( q(z, x; \phi) \) to the generative distribution \( p(z, x; \theta) = p(z)p(x \mid z; \theta) \)

**Intuition:** \( p(x; \theta)p(z \mid x; \theta) = p_{data}(x)q(z \mid x; \phi) \) if

1. \( p_{data}(x) = p(x; \theta) \)
2. \( q(z \mid x; \phi) = p(z \mid x; \theta) \) for all \( x \)
3. Hence we get the VAE objective:
   \[ -D_{KL}(p_{data}(x)\parallel p(x; \theta)) - E_{x \sim p_{data}} [D_{KL}(q(z \mid x; \phi)\parallel p(z \mid x; \theta))] \]

- Many other variants are possible! VAE + GAN:
  \[ -JSD(p_{data}(x)\parallel p(x; \theta)) - D_{KL}(p_{data}(x)\parallel p(x; \theta)) - E_{x \sim p_{data}} [D_{KL}(q(z \mid x; \phi)\parallel p(z \mid x; \theta))] \]
Adversarial Autoencoder (VAE + GAN)

\[
E_{x \sim p_{data}}[L(x; \theta, \phi)] \equiv -D_{KL}(p_{data}(x)q(z | x; \phi) \parallel p(x; \theta)p(z|x; \theta))
\]

- Optimizing ELBO is the same as matching the inference distribution \(q(z, x; \phi)\) to the generative distribution \(p(z, x; \theta)\)

- **Symmetry:** Using alternative factorization:
  \[
p(z)p(x|z; \theta) = q(z; \phi)q(x | z; \phi)
  \]
  if
  1. \(q(z; \phi) = p(z)\)
  2. \(q(x | z; \phi) = p(x|z; \theta)\) for all \(z\)
  3. We get an equivalent form of the VAE objective:
  \[
  -D_{KL}(q(z; \phi)\parallel p(z)) - E_{z \sim q(z;\phi)}[D_{KL}(q(x | z; \phi)\parallel p(x|z; \theta))]
  \]

- Other variants are possible. For example, can add \(-JSD(q(z; \phi)\parallel p(z))\) to match features in latent space (Zhao et al., 2017; Makhzani et al, 2018)
Information Preference

\[ E_{x \sim p_{data}}[\mathcal{L}(x; \theta, \phi)] \equiv -D_{KL}(p_{data}(x)q(z | x; \phi) \| p(x; \theta)p(z | x; \theta)) \]

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

- **Issue**: System of equations with many more variables than constraints
Explicitly add a mutual information term to the objective

\[
- D_{KL}(p_{\text{data}}(x) q(z \mid x; \phi) \| p(x; \theta) p(z \mid x; \theta)) + \alpha MI(x, z)
\]

MI intuitively measures how far \(x\) and \(z\) are from being independent

\[
MI(x, z) = D_{KL}(p(z, x; \theta) \| p(z) p(x; \theta))
\]

InfoGAN (Chen et al, 2016) used to learn meaningful (disentangled?) representations of the data

\[
MI(x, z) - E_{x \sim p_{\theta}} [D_{KL}(p_{\theta}(z \mid x) \| q_{\phi}(z \mid x))] - JSD(p_{\text{data}}(x) \| p_{\theta}(x))
\]
$\beta$-VAE

Model proposed to learn disentangled features / latent variables (Higgins, 2016)

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

It is a VAE with scaled up KL divergence term ($\beta > 1$). This is equivalent (up to constants) to the following objective:

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

See *The Information Autoencoding Family: A Lagrangian Perspective on Latent Variable Generative Models* for more examples.
We have covered several useful building blocks: autoregressive, latent variable models, flow models, GANs, EBMs, score-based models.

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.