Evaluating Generative Models

Volodymyr Kuleshov

Cornell Tech

Lecture 13
Announcements

- Please start emailing me your presentation topics
- Presentation guidelines have been posted online and on Piazza.
- Proposed format:
  - Motivation for the problem being studied; why is it interesting?
  - Context and previous work in this area;
  - Summary of the novel ideas and contributions in the presented papers;
  - Explanation of the technical material in the papers;
  - Experimental or theoretical results;
  - Discussion of the results;
  - Conclusion and open-ended questions.
Background

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

Plan for today: Evaluating generative models
Evaluating generative models can be very tricky

**Key question**: What is the task that you care about?

- Density estimation
- Sampling/generation
- Latent representation learning
- More than one task? Custom downstream task? E.g., Semisupervised learning, image translation, compressive sensing etc.

In any research field, evaluation drives progress. How do we evaluate generative models?
Evaluation - Density Estimation

Straightforward for models which have tractable likelihoods

- Split dataset into train, validation, test sets
- Evaluate gradients based on train set
- Tune hyperparameters (e.g., learning rate, neural network architecture) based on validation set
- Evaluate generalization by reporting likelihoods on test set
There is an inherent **bias-variance trade off** when selecting the hypothesis class. Error in learning due to both things: bias and variance.

Hypothesis space: linear relationship

- Does it fit well? Underfits

Hypothesis space: high degree polynomial

- Overfits

Hypothesis space: low degree polynomial

- Right tradeoff
**Caveat:** Not all models have tractable likelihoods e.g., VAEs, GANs

For VAEs, we can compare evidence lower bounds (ELBO) to log-likelihoods

In general, we can use kernel density estimates only via samples (non-parametric)

**Note:** These alternatives often make strong assumptions and can be misleading in practice
Kernel Density Estimation

- Given: A model $p_\theta(x)$ with an intractable/ill-defined density
- Let $S = \{x^{(1)}, x^{(2)}, \ldots, x^{(6)}\}$ be 6 data points drawn from $p_\theta$.

<table>
<thead>
<tr>
<th>$x^{(1)}$</th>
<th>$x^{(2)}$</th>
<th>$x^{(3)}$</th>
<th>$x^{(4)}$</th>
<th>$x^{(5)}$</th>
<th>$x^{(6)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.1</td>
<td>-1.3</td>
<td>-0.4</td>
<td>1.9</td>
<td>5.1</td>
<td>6.2</td>
</tr>
</tbody>
</table>

- What is $p_\theta(-0.5)$?
- **Answer 1:** Since $-0.5 \notin S$, $p_\theta(-0.5) = 0$
- **Answer 2:** Compute a histogram by binning the samples

Bin width= 2, min height= 1/12 (area under histogram should equal 1). What is $p_\theta(-0.5)$? 1/6 $p_\theta(-1.99)$? 1/6 $p_\theta(-2.01)$? 1/12
Kernel Density Estimation

**Answer 3:** Compute kernel density estimate (KDE) over $S$

$$
\hat{p}(x) = \frac{1}{n} \sum_{x(i) \in S} K \left( \frac{x - x(i)}{\sigma} \right)
$$

where $\sigma$ is called the bandwidth parameter and $K$ is called the kernel function.

- Example: Gaussian kernel, $K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} u^2\right)$
- Histogram density estimate vs. KDE estimate with Gaussian kernel
A kernel $K$ is any non-negative function satisfying two properties

- Normalization: $\int_{-\infty}^{\infty} K(u) du = 1$ (ensures KDE is also normalized)
- Symmetric: $K(u) = K(-u)$ for all $u$

Intuitively, a kernel is a measure of similarity between pairs of points (function is higher when the difference in points is close to 0)

- **Bandwidth** $\sigma$ controls the smoothness (see right figure above)
  - Optimal sigma (black) is such that KDE is close to true density (grey)
  - Low sigma (red curve): undersmoothed
  - High sigma (green curve): oversmoothed
  - Tuned via crossvalidation

- **Con**: KDE is very unreliable in higher dimensions
Importance Sampling

- **Likelihood weighting:**

  \[ p(x) = E_{p(z)}[p(x|z)] \]

  Can have high variance if \( p(z) \) is far from \( p(z|x) \)!

- **Annealed importance sampling:** General purpose technique to estimate ratios of normalizing constants \( \mathcal{N}_2/\mathcal{N}_1 \) of any two distributions via importance sampling

- A good implementation available in Tensorflow probability
  
  \[ \text{tfp.mcmc.sample_annealed_importance_chain} \]
Annealed Importance Sampling

General purpose technique to estimate ratios of normalizing constants $\mathcal{N}_2/\mathcal{N}_1$ of any two distributions via importance sampling

- Main idea: construct a sequence of intermediate distributions that gradually interpolate between $p_1$ and $p_2$.

$$f_j = p_1^{\beta_j} \cdot p_2^{(1-\beta_j)}$$

$$w = \frac{f_{n-1}(x_{n-1})}{f_n(x_{n-1})} \cdot \frac{f_{n-2}(x_{n-2})}{f_{n-1}(x_{n-2})} \cdot ...$$

- We interpolate from $p(z)$ to the unnormalized estimate of $p(z|x)$
- For estimating $p(x)$, first distribution is $p(z)$ (with $\mathcal{N}_1 = 1$) and second distribution is $p(z|x)$ (with $\mathcal{N}_2 = p(x) = \int_x p(x, z) \, dz$)
- Gives unbiased estimates of likelihoods, but biased estimates of log-likelihoods
What are some pros and cons of using likelihood?

- Optimal generative model will give best **sample quality** and highest test **log-likelihood**
- For imperfect models, achieving high log-likelihoods might not always imply good sample quality, and vice-versa (Theis et al., 2016)
- Likelihood is only one possible metric to define the distance between two distributions
Example: Great test log-likelihoods, poor samples. E.g., For a discrete noise mixture model $p_\theta(x) = 0.01 p_{\text{data}}(x) + 0.99 p_{\text{noise}}(x)$
- 99% of the samples are just noise
- Taking logs, we get a lower bound

$$\log p_\theta(x) = \log [0.01 p_{\text{data}}(x) + 0.99 p_{\text{noise}}(x)]$$
$$\geq \log 0.01 p_{\text{data}}(x) = \log p_{\text{data}}(x) - \log 100$$

For expected likelihoods, we know that
- Lower bound

$$E_{p_{\text{data}}} [\log p_\theta(x)] \geq E_{p_{\text{data}}} [\log p_{\text{data}}(x)] - \log 100$$

- Upper bound (via non-negativity of KL)

$$E_{p_{\text{data}}} [\log p_{\text{data}}(x))] \geq E_{p_{\text{data}}} [\log p_\theta(x)]$$
- As we increase the dimension of $x$, absolute value of $\log p_{\text{data}}(x)$ increases proportionally but $\log 100$ remains constant. Hence, $E_{p_{\text{data}}} [\log p_\theta(x)] \approx E_{p_{\text{data}}} [\log p_{\text{data}}(x)]$ in very high dimensions
Log-Likelihood vs. Sample Quality

- **Example:** Great samples, poor test log-likelihoods. E.g., Memorizing training set
  - Samples look exactly like the training set (cannot do better!)
  - Test set will have zero probability assigned (cannot do worse!)
- The above cases suggest that it might be useful to disentangle likelihoods and samples
- **Likelihood-free learning** consider objectives that do not depend directly on a likelihood function
Which of these two sets of generated samples “look” better?

Human evaluations (e.g., Mechanical Turk) are expensive, biased, hard to reproduce

Generalization is hard to define and assess: memorizing the training set would give excellent samples but clearly undesirable

Quantitative evaluation of a qualitative task can have many answers

Popular metrics: Inception Scores, Frechet Inception Distance, Kernel Inception Distance
Inception Scores

- **Assumption 1:** We are evaluating sample quality for generative models trained on labelled datasets
- **Assumption 2:** We have a good probabilistic classifier \(c(y|x)\) for predicting the label \(y\) for any point \(x\)
- We want samples from a good generative model to satisfy two criteria: sharpness and diversity
- **Sharpness (S)**

\[
S = \exp \left( E_{x \sim p} \left[ \int c(y|x) \log c(y|x) \, dy \right] \right)
\]

- High sharpness implies classifier is confident in making predictions for generated images
- That is, classifier’s predictive distribution \(c(y|x)\) has low entropy
Inception Scores

- **Diversity (D)**

  \[
  D = \exp \left( -E_{x \sim p} \left[ \int c(y|x) \log c(y) dy \right] \right)
  \]

  where \( c(y) = E_{x \sim p}[c(y|x)] \) is the classifier’s marginal predictive distribution

- High diversity implies \( c(y) \) has high entropy

- Inception scores (IS) combine the two criteria of sharpness and diversity into a simple metric

  \[
  IS = D \times S = E_{x \sim p} [KL (p(y|x) || p(y))]
  \]

  Correlates well with human judgement in practice

- If classifier is not available, a classifier trained on a large dataset, e.g., Inception Net trained on the ImageNet dataset
Frechet Inception Distance

- Inception Scores only require samples from $p_{\theta}$ and do not take into account the desired data distribution $p_{\text{data}}$ directly (only implicitly via a classifier).

- **Frechet Inception Distance (FID)** measures similarities in the feature representations (e.g., those learned by a pretrained classifier) for datapoints sampled from $p_{\theta}$ and the test dataset.

- **Computing FID:**
  - Let $\mathcal{G}$ denote the generated samples and $\mathcal{T}$ denote the test dataset.
  - Compute feature representations $F_{\mathcal{G}}$ and $F_{\mathcal{T}}$ for $\mathcal{G}$ and $\mathcal{T}$ respectively (e.g., prefinal layer of Inception Net).
  - Fit a multivariate Gaussian to each of $F_{\mathcal{G}}$ and $F_{\mathcal{T}}$. Let $\mu_{\mathcal{G}}, \Sigma_{\mathcal{G}}$ and $\mu_{\mathcal{T}}, \Sigma_{\mathcal{T}}$ denote the mean and covariances of the two Gaussians.
  - FID is defined as

$$\text{FID} = \|\mu_{\mathcal{T}} - \mu_{\mathcal{G}}\|^2 + \text{Tr}(\Sigma_{\mathcal{T}} + \Sigma_{\mathcal{G}} - 2(\Sigma_{\mathcal{T}}\Sigma_{\mathcal{G}})^{1/2})$$

- Lower FID implies better sample quality.
Kernel Inception Distance

- **Maximum Mean Discrepancy (MMD)** is a two-sample test statistic that compares samples from two distributions $p$ and $q$ by computing differences in their moments (mean, variances etc.)

- Key idea: Use a suitable kernel e.g., Gaussian to measure similarity between points

$$MMD(p, q) = E_{x,x' \sim p}[K(x, x')] + E_{x,x' \sim q}[K(x, x')] - 2E_{x \sim p, x' \sim q}[K(x, x')]$$

- Intuitively, MMD is comparing the “similarity” between samples within $p$ and $q$ individually to the samples from the mixture of $p$ and $q$

- **Kernel Inception Distance (KID):** compute the MMD in the feature space of a classifier (e.g., Inception Network)

- FID vs. KID
  - FID is biased (can only be positive), KID is unbiased
  - FID can be evaluated in $O(n)$ time, KID evaluation requires $O(n^2)$ time
What does it mean to learn “good” latent representations?

For a downstream task, the representations can be evaluated based on the corresponding performance metrics e.g., accuracy for semi-supervised learning, reconstruction quality for denoising.

Example: Improved GANs by Salimans et al.

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of incorrectly predicted test examples for a given number of labeled samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>DGN [22]</td>
<td>333 ± 14</td>
</tr>
<tr>
<td>Virtual Adversarial [23]</td>
<td>212</td>
</tr>
<tr>
<td>CatGAN [14]</td>
<td>191 ± 10</td>
</tr>
<tr>
<td>Ladder network [25]</td>
<td>106 ± 37</td>
</tr>
<tr>
<td>Our model</td>
<td>1677 ± 452</td>
</tr>
<tr>
<td>Ensemble of 10 of our models</td>
<td>1134 ± 445</td>
</tr>
</tbody>
</table>

Table 1: Number of incorrectly classified test examples for the semi-supervised setting on permutation invariant MNIST. Results are averaged over 10 seeds.
Evaluating latent representations

- What does it mean to learn “good” latent representations?
- For unsupervised tasks, there is no one-size-fits-all
- Three commonly used notions for evaluating unsupervised latent representations
  - Clustering
  - Compression
  - Disentanglement
Recall: Adversarial Autoencoder (VAE + GAN)

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

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

**Symmetry:** Using alternative factorization:
\( p(z)p(x \mid z; \theta) = q(z; \phi)q(x \mid z; \phi) \) if

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

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

- Representations that can group together points based on some semantic attribute are potentially useful (e.g., semi-supervised classification)

- Clusters can be obtained by applying k-means or any other algorithm in the latent space of generative model

Source: Makhzani et al., 2018

- 2D representations learned by two generative models for MNIST digits with colors denoting true labels. Which is better? B or D?
For labeled datasets, there exists many quantitative evaluation metrics. Note labels are only used for evaluation, not obtaining clusters itself (i.e., clustering is unsupervised).

```python
from sklearn.metrics.cluster import completeness_score, homogeneity_score, v_measure_score
```

**Completeness score** (between $[0, 1]$): maximized when all the data points that are members of a given class are elements of the same cluster.

```
completeness_score(labels_true=[0, 0, 1, 1], labels_pred=[0, 1, 0, 1]) % 0
```

**Homogeneity score** (between $[0, 1]$): maximized when all of its clusters contain only data points which are members of a single class.

```
homogeneity_score(labels_true=[0, 0, 1, 1], labels_pred=[1, 1, 0, 0]) % 1
```

**V measure score** (also called normalized mutual information, between $[0, 1]$): harmonic mean of completeness and homogeneity score.

```
v_measure_score(labels_true=[0, 0, 1, 1], labels_pred=[1, 1, 0, 0]) % 1
```
Latent representations can be evaluated based on the maximum compression they can achieve without significant loss in reconstruction accuracy.

Standard metrics such as Mean Squared Error (MSE), Peak Signal to Noise Ratio (PSNR), Structure Similarity Index (SSIM).

Source: Santurkar et al., 2018
Intuitively, we want representations that disentangle independent and interpretable attributes of the observed data.

Provide user control over the attributes of the generated data.

When $Z_1$ is fixed, size of the generated object never changes.
When $Z_1$ is changed, the change is restricted to the size of the generated object.

Source: Higgins et al., 2018
Source: Shu et al., 2019
Recall: Information Preference

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

- 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:** Many more variables than constraints
Recall: InfoGAN

- Explicitly add a mutual information term to the objective

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

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

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

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

\[MI(x, z) - E_{x \sim p_{\theta}}[D_{KL}(p_{\theta}(z|x) \parallel q_{\phi}(z|x))] - JSD(p_{\text{data}}(x) \parallel p_{\theta}(x))\]
Many quantitative evaluation metrics

- **Beta-VAE metric (Higgins et al., 2017):** Accuracy of a linear classifier that predicts a fixed factor of variation
  - Sample from true generative process in which one disentangled factor is fixed.
  - Reconstruct latents. One factor should have less variation.
  - Use classifier to detect it.

- Many other metrics: Factor-VAE metric, Mutual Information Gap, SAP score, DCI disentanglement, Modularity
Summary

- Quantitative evaluation of generative models is a challenging task.
- For downstream applications, one can rely on application-specific metrics.
- For unsupervised evaluation, metrics can significantly vary based on end goal: density estimation, sampling, latent representations.