Deep Generative Models

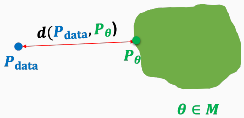
Lecture 10: Evaluating Generative Models

Aditya Grover

UCLA

Mid-quarter summary

$x^{(j)} \sim P_{\text{data}}$
$j = 1, 2, \ldots, m$



Model family:

- Probability density/mass functions
    - Autoregressive models. $p_\theta(\mathbf{x}) = \prod_{i=1}^{d} p_\theta(\mathbf{x}_i \mid \mathbf{x}_{<i})$.
    - Normalizing flow models. $p_\theta(\mathbf{x}) = p(\mathbf{z})|\det(J_{f_\theta}(\mathbf{x}))|$, where $\mathbf{z} = f_\theta(\mathbf{x})$.
    - Latent variable models (e.g., variational autoencoders). $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})d\mathbf{z}$.
- Sample generation processes
    - Generative adversarial networks (GANs). $\mathbf{z} \sim p(\mathbf{z}), \mathbf{x} = g_\theta(\mathbf{z})$.

Mid-quarter summary

Distances of probability distributions:

- KL divergence (maximum likelihood)
  - Autoregressive models.
  - Normalizing flow models.
  - ELBO in Variational autoencoders.
- f-divergences, Wasserstein distances
  - Generative adversarial networks (f-GANs, WGANs).

Plan for today: Evaluating generative models

Evaluation

- In any research field, evaluation drives progress. How do we evaluate generative models?
- Evaluating generative models is highly non-trivial.
- **Key question**: What is the task that you care about?
    - Density estimation
    - Compression
    - Sampling/generation
    - Latent representation learning
    - More than one task? Custom downstream task? E.g., Semisupervised learning, image translation, compressive sensing etc.

Evaluation - Density Estimation or Compression

- 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
- **Caveat:** Not all models have tractable likelihoods e.g., VAEs, GANs, EBMs
- For VAEs, we can compare evidence lower bounds (ELBO) to log-likelihoods. How about GANs? How to estimate the model likelihood if we only have samples?

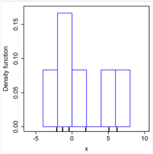
In general, unbiased estimation of density functions from samples is impossible. Approximation methods are necessary. We can use kernel density estimates via samples alone.

Kernel Density Estimation

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

$\mathbf{x}^{(1)}$	$\mathbf{x}^{(2)}$	$\mathbf{x}^{(3)}$	$\mathbf{x}^{(4)}$	$\mathbf{x}^{(5)}$	$\mathbf{x}^{(6)}$
-2.1	-1.3	-0.4	1.9	5.1	6.2

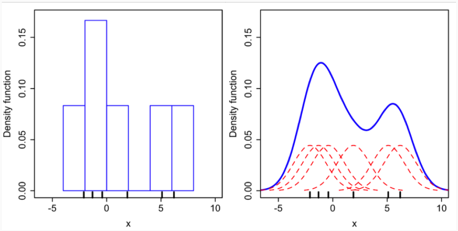
- What is $p_\theta(-0.5)$?
- **Answer 1:** Since $-0.5 \notin \mathcal{S}$, $p_\theta(-0.5) = 0$
- **Answer 2:** Compute a histogram by binning the samples



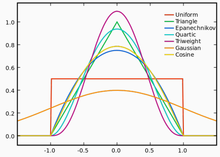
- Bin width$=$ 2, min height$=$ $1/12$ (area under histogram should equal 1). What is $p_\theta(-0.5)$? $1/6$ $p_\theta(-1.99)$? $1/6$ $p_\theta(-2.01)$? $1/12$

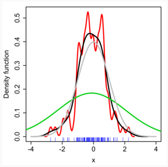
Kernel Density Estimation

- **Answer 3:** Compute kernel density estimate (KDE) over $\mathcal{S}$
$$\hat{p}(\mathbf{x}) = \frac{1}{n} \sum_{\mathbf{x}^{(i)} \in \mathcal{S}} K\left(\frac{\mathbf{x} - \mathbf{x}^{(i)}}{\sigma}\right)$$
where σ 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



Kernel Density Estimation



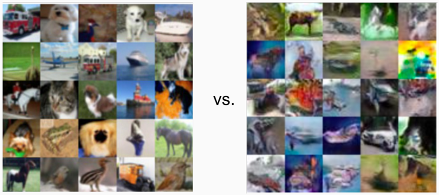


- A **kernel** K is any non-negative function satisfying:
  - Normalization: $\int_{-\infty}^{\infty} K(u)\mathrm{d}u = 1$ (ensures KDE is also normalized)
  - Symmetric: $K(u) = K(-u)$ for all u
- Intuitively, a kernel is a measure of similarity between pairs of points (function is higher when the difference in points is low)
- **Bandwidth** σ controls the smoothness (see right figure)
  - 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

Importance Sampling for latent variable models

- **Likelihood weighting:**
$$p(\mathbf{x}) = E_{p(\mathbf{z})}[p(\mathbf{x}|\mathbf{z})]$$
Can have high variance if $p(\mathbf{z})$ is far from $p(\mathbf{z}|\mathbf{x})$!
- **Annealed importance sampling:** Construct a sequence of intermediate distributions that gradually interpolate from $p(\mathbf{z})$ to the unnormalized estimate of $p(\mathbf{z}|\mathbf{x})$
- General purpose technique to estimate ratios of normalizing constants Z_2/Z_1 of any two unnormalized distributions
- For estimating $p(\mathbf{x})$, first distribution is $p(\mathbf{z})$ (with $Z_1 = 1$) and second distribution is $p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ (with $Z_2 = p(\mathbf{x}) = \int_{\mathbf{x}} p(\mathbf{x}, \mathbf{z})\mathrm{d}\mathbf{z}$)
- Gives unbiased estimates of likelihoods, but biased estimates of log-likelihoods
- A good implementation available in Tensorflow probability `tfp.mcmc.sample_annealed_importance_chain`

Evaluation - Sample quality



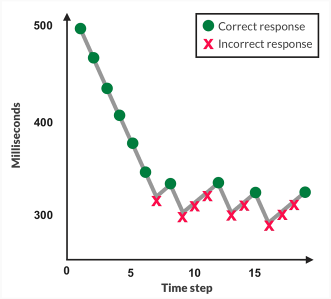
$S_1 = \{\mathbf{x} \sim P\}$ $S_2 = \{\mathbf{x} \sim Q\}$

- Which of these two sets of generated samples "look" better?
- Human evaluations (e.g., Mech Turk) are the gold standard.

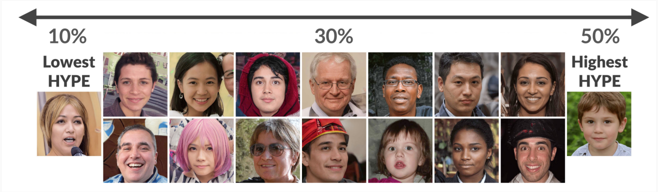
HYPE: Human eYe Perceptual Evaluation (Zhou et al., 2019)

- $\text{HYPE}_{\text{time}}$: the minimum time people needed to make accurate classifications. The larger, the better.
- HYPE_{∞}: The percentage of samples that deceive people under unlimited time. The larger, the better.
- https://stanfordhci.github.io/gen-eval/

Evaluation - Sample quality

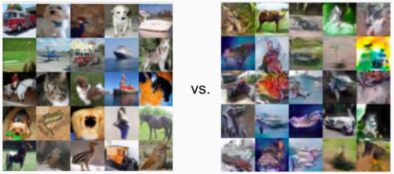


The process of determining HYPE$_{\text{time}}$ scores.



HYPE$_{\infty}$ scores for samples generated from a StyleGAN.

Evaluation - Sample quality

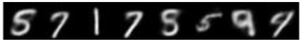


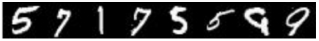
$S_1 = \{\mathbf{x} \sim P\}$ $S_2 = \{\mathbf{x} \sim Q\}$

- 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 $c(y|\mathbf{x})$ for predicting the label y for any point $\mathbf{x}$
- We want samples from a good generative model to satisfy two criteria: sharpness and diversity
- **Sharpness (S)**


Low sharpness

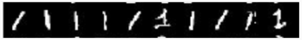

High sharpness

$$S = \exp\left(E_{\mathbf{x}\sim p}\left[\int c(y|\mathbf{x}) \log c(y|\mathbf{x}) \mathrm{d}y\right]\right)$$

- High sharpness implies classifier is confident in making predictions for generated images
- That is, classifier's predictive distribution $c(y|\mathbf{x})$ has low entropy

Inception Scores

- **Diversity (D)**



Low diversity

High diversity

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

where $c(y) = E_{\mathbf{x}\sim p}[c(y|\mathbf{x})]$ 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

$$IS = D \times S$$

- Higher IS corresponds to better quality.
- 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_θ and do not take into account the desired data distribution p_{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_θ and the test dataset
- Computing FID:
  - Let $\mathcal{G}$, $\mathcal{T}$ denote the generated samples and 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 their mean and covariances
  - FID: Wasserstein-2 distance between these two Gaussians:

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

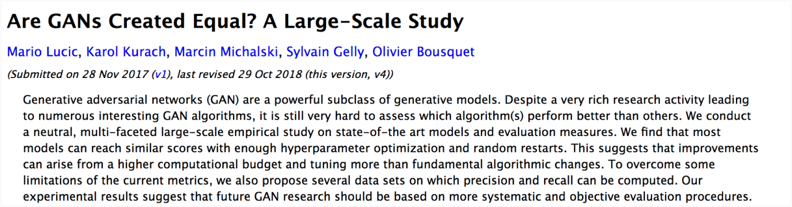
- Lower FID implies better sample quality

Kernel Inception Distance

- **Maximum Mean Discrepancy (MMD)** is a two-sample test statistic that compares samples from two distributions p and q based on differences in their moments (mean, variances etc.)
- Key idea: Use a suitable kernel e.g., Gaussian to measure similarity between points

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

Evaluating sample quality - Best practices


Are GANs Created Equal? A Large-Scale Study

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet

(Submitted on 28 Nov 2017 (v1), last revised 29 Oct 2018 (this version, v4))

Generative adversarial networks (GAN) are a powerful subclass of generative models. Despite a very rich research activity leading to numerous interesting GAN algorithms, it is still very hard to assess which algorithm(s) perform better than others. We conduct a neutral, multi-faceted large-scale empirical study on state-of-the art models and evaluation measures. We find that most models can reach similar scores with enough hyperparameter optimization and random restarts. This suggests that improvements can arise from a higher computational budget and tuning more than fundamental algorithmic changes. To overcome some limitations of the current metrics, we also propose several data sets on which precision and recall can be computed. Our experimental results suggest that future GAN research should be based on more systematic and objective evaluation procedures.

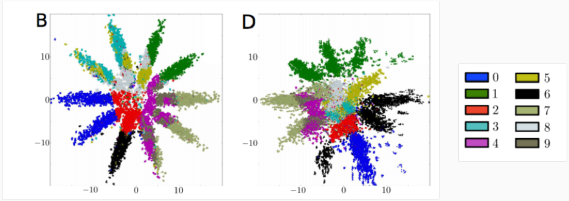
- Spend time tuning your baselines (architecture, learning rate, optimizer etc.). Be amazed (rather than dejected) at how well they can perform
- Use random seeds for reproducibility
- Report results averaged over multiple random seeds along with confidence intervals

Evaluating latent representations

- 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
- For unsupervised tasks, there is no one-size-fits-all
- Three commonly used notions for evaluating unsupervised latent representations
    - Clustering
    - Compression
    - Disentanglement

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

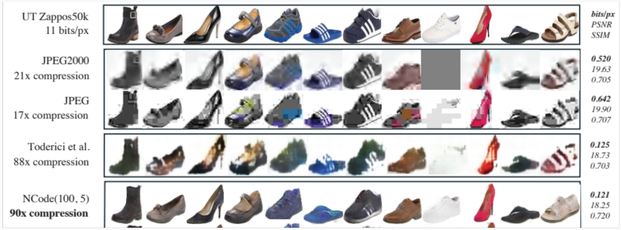
- 2D representations learned by two generative models for MNIST digits with colors denoting true labels. Which is better? B or D?

Clustering

- For labelled datasets, there exists many quantitative evaluation metrics. Note labels are only used for evaluation, not obtaining clusters itself (i.e., clustering is unsupervised)
- `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

Lossy Compression or Reconstruction

- Latent representations can be evaluated based on the maximum compression they can achieve without significant loss in reconstruction accuracy

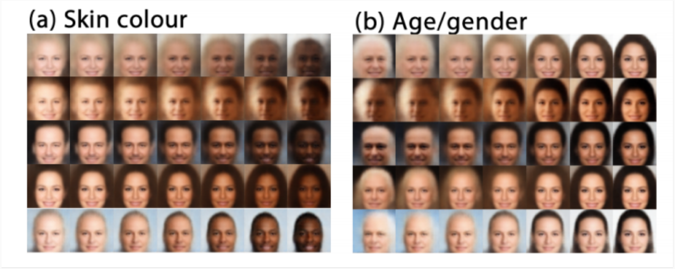


Source: Santurkar et al., 2018

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

Disentanglement

- Intuitively, we want representations that disentangle **independent and interpretable** attributes of the observed data



Source: Higgins et al., 2018

- 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

Disentanglement

- Many quantitative evaluation metrics
  - Beta-VAE metric (Higgins et al., 2017): Accuracy of a linear classifier that predicts a fixed factor of variation
  - Many other metrics: Factor-VAE metric, Mutual Information Gap, SAP score, DCI disentanglement, Modularity
  - Check `disentanglement_lib` for implementations of these metrics
- Disentangling generative factors is theoretically impossible without additional assumptions

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