

# Evaluating Generative Models

Volodymyr Kuleshov

Cornell Tech

Lecture 14

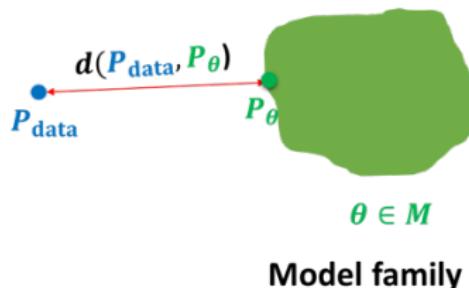
# Announcements

- Project progress report (3 pages) due on March 31.
- I gave everyone feedback on their project proposals in Gradescope.
- Please make sure to have signed up for a presentation and reviewing slot **by Wed.** I will take off one point per day after that.

# Recap So Far



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



We have seen several generative models and several tasks:

- ① **Generation:** Autoregressive, Latent Variable Models, GANs, Score-Based & Diffusion Models
- ② **Density Estimation and Outlier Detection:** Autoregressive Models, Normalizing Flows, Energy-Based Models
- ③ **Representation Learning:** Latent Variable Models

How do we evaluate models and decide which one to use for each task?

# Evaluation

- Evaluating generative models can be very tricky
- **Key question:** What is the task that you care about?
  - Classical generative modeling tasks?
    - Density estimation
    - Sampling/generation
    - Latent representation learning
  - More than one task? Custom downstream task?
    - Semisupervised learning
    - Image translation
    - Compressive sensing
- In any research field, evaluation drives progress. How do we evaluate generative models?

# Lecture Outline

## ① Density Estimation

- Overfitting & Underfitting
- Importance Sampling
- Kernel Density Estimation

## ② Evaluating Sample Quality

- Inception Scores
- Frechet Inception Distance
- Kernel Inception Distance

## ③ Evaluating Latent Variables

- Supervised Clustering
- Compression
- Disentanglement

# Evaluating Density Estimation

Density estimation seeks to learn a good model  $p_\theta$  that approximates  $p_{\text{data}}$ . We can use  $p_\theta$  for outlier detection, generation, etc.

- In practice, we evaluate models using the KL divergence, which is equivalent (up to a constant) to maximum likelihood.

It's straightforward to evaluate models with tractable likelihoods (autoregressive models, normalizing flows)

- Split dataset into train, validation, test sets
- Fit the model on the train set
- Tune hyperparameters (e.g., learning rate, neural network architecture) based on validation set
- Evaluate generalization by reporting likelihoods on test set

# Density Estimation Without Tractable Likelihoods

Not all models have tractable likelihoods e.g., VAEs, GANs, energy based models. What do we do in such cases?

- For **VAEs**, we can compare evidence lower bounds (ELBO) to log-likelihoods.
- For **energy-based models** and **VAEs**, we can perform (annealed) importance sampling to estimate the density directly.
- For implicit models (e.g., **GANs**), we can estimate the density from samples using non-parametric kernel density estimation.

**Note:** These alternatives often make strong assumptions and don't always work easily and reliably in practice.

# Importance Sampling

Importance sampling is a clever way of applying Monte Carlo. First, we express  $p_\theta(\mathbf{x})$  as a different kind of expectation:

$$p_\theta(\mathbf{x}) = \sum_{\mathbf{z} \in \mathcal{Z}} p_\theta(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z} \in \mathcal{Z}} \frac{q(\mathbf{z})}{q(\mathbf{z})} p_\theta(\mathbf{x}, \mathbf{z}) = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} \left[ \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right]$$

where  $q$  can be any distribution over  $\mathbf{z}$ . We then apply Monte Carlo:

- ① Sample  $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(k)}$  from  $q(\mathbf{z})$  (instead of uniformly at random!)
- ② Approximate expectation with sample average

$$p_\theta(\mathbf{x}) \approx \frac{1}{k} \sum_{j=1}^k \frac{p_\theta(\mathbf{x}, \mathbf{z}^{(j)})}{q(\mathbf{z}^{(j)})}$$

What is a good choice for  $q(\mathbf{z})$ ? Using  $p(\mathbf{z}|\mathbf{x})$  yields correct density in one sample. In practice we choose tractable  $q(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$ .

- ① The same technique to any  $f(\mathbf{z})$  such as  $\log p_\theta(\mathbf{x}, \mathbf{z})$  in VAEs.
- ② *Annealed* importance sampling is an advanced extension that works for energy-based models.

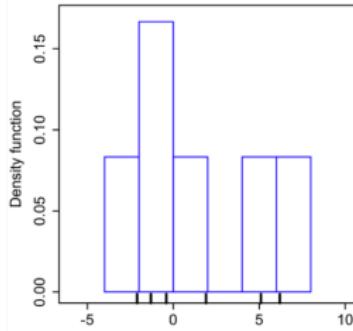
# Kernel Density Estimation

What if we have a model  $p_\theta(\mathbf{x})$  from which we can only sample?

- Let  $\mathcal{S} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(6)}\}$  be 6 data points drawn from  $p_\theta$ .

$\mathbf{x}^{(1)}$	$\mathbf{x}^{(2)}$	$\mathbf{x}^{(3)}$	$\mathbf{x}^{(4)}$	$\mathbf{x}^{(5)}$	$\mathbf{x}^{(6)}$
-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  $p_\theta(-0.5)$ ?  $1/6$   $p_\theta(-1.99)$ ?  $1/6$   $p_\theta(-2.01)$ ?  $1/12$

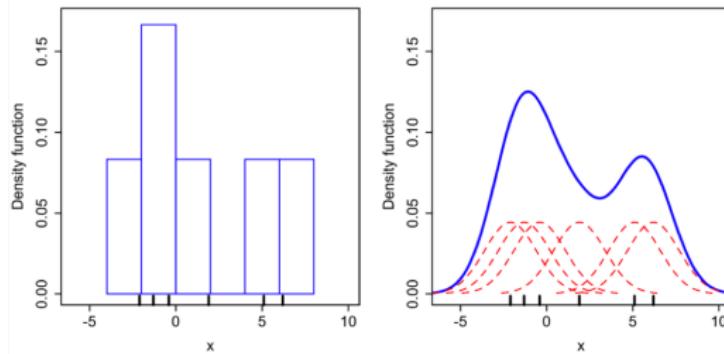
# Kernel Density Estimation

- **Answer 3:** Compute kernel density estimate (KDE) over  $\mathcal{S}$

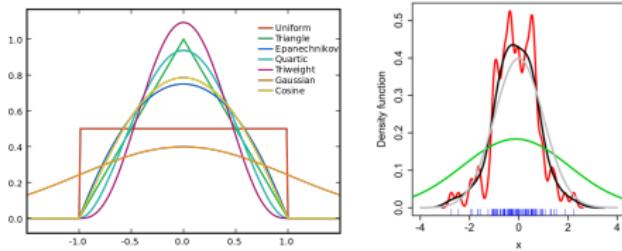
$$\hat{p}(\mathbf{x}) = \frac{1}{n} \sum_{\mathbf{x}^{(i)} \in \mathcal{S}} K\left(\frac{\mathbf{x} - \mathbf{x}^{(i)}}{\sigma}\right)$$

where  $\sigma$  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



# Kernel Density Estimation



- 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

# Lecture Outline

## ① Density Estimation

- Overfitting & Underfitting
- Importance Sampling
- Kernel Density Estimation

## ② Evaluating Sample Quality

- Inception Scores
- Frechet Inception Distance
- Kernel Inception Distance

## ③ Evaluating Latent Variables

- Supervised Clustering
- Compression
- Disentanglement

# High Likelihood Does Not Imply Great Samples

**Example:** Great test log-likelihoods, poor samples. Consider a mixture model  $p_\theta(\mathbf{x}) = 0.01p_{\text{data}}(\mathbf{x}) + 0.99p_{\text{noise}}(\mathbf{x})$  of data and noise.

- 99% of the samples are just noise
- Taking logs, we get a lower bound

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

- For expected likelihoods, we know that
  - Lower bound

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

- Upper bound (via non-negativity of KL)

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

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

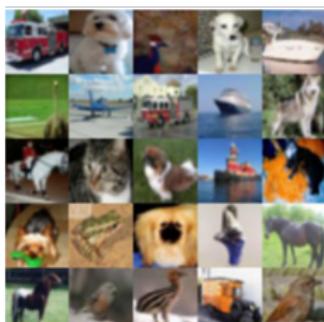
# Great Samples Don't Imply High Likelihood

**Example:** Great samples, poor test log-likelihoods. Consider a model that simply memorizes the 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.

# Evaluation of Sample Quality

Instead of using likelihood, we can try to evaluate samples directly.



vs.



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

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

- 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)**

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

- 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)**



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

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

$$IS = D \times S = \mathbb{E}_{x \sim p} [KL(c(y|x) || c(y))]$$

- 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

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

- 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

# Lecture Outline

## ① Density Estimation

- Overfitting & Underfitting
- Importance Sampling
- Kernel Density Estimation

## ② Evaluating Sample Quality

- Inception Scores
- Frechet Inception Distance
- Kernel Inception Distance

## ③ Evaluating Latent Variables

- Supervised Clustering
- Compression
- Disentanglement

# Evaluating Latent Representations on Downstream Tasks

- 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.

Model	Number of incorrectly predicted test examples for a given number of labeled samples			
	20	50	100	200
DGN [22]			$333 \pm 14$	
Virtual Adversarial [23]			212	
CatGAN [14]			$191 \pm 10$	
Skip Deep Generative Model [24]			$132 \pm 7$	
Ladder network [25]			$106 \pm 37$	
Auxiliary Deep Generative Model [24]			$96 \pm 2$	
Our model	$1677 \pm 452$	$221 \pm 136$	$93 \pm 6.5$	$90 \pm 4.2$
Ensemble of 10 of our models	$1134 \pm 445$	$142 \pm 96$	$86 \pm 5.6$	$81 \pm 4.3$

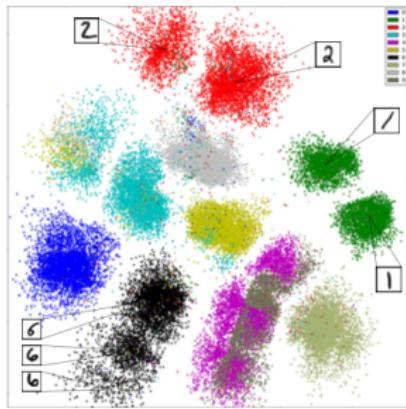
Table 1: Number of incorrectly classified test examples for the semi-supervised setting on permutation invariant MNIST. Results are averaged over 10 seeds.

# Evaluating Latents Without Downstream Tasks

- 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

# Clustering in Latent Space

- Clusters in latent space often reveal useful structure about the data:



- These clusters can be computed automatically by applying k-means or any other algorithm in the latent space of generative model
- When labels are available (e.g., MNIST digits), we can evaluate clusters quantitatively using standard clustering metrics e.g., completeness score, homogeneity score, v-measure.

# Evaluating Clustering When Labels Are Available

- 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`

# Compression

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

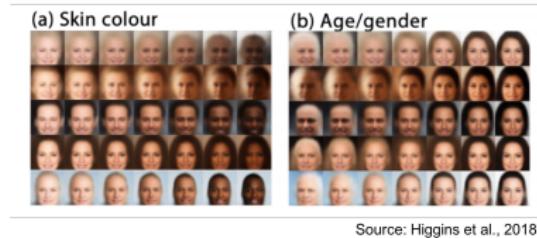


Source: Santurkar et al., 2018

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

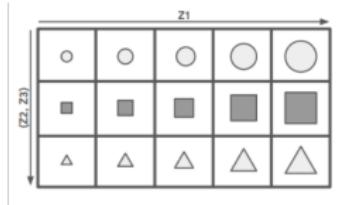
# Disentanglement

- 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



Source: Shu et al., 2019

- 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

# Recall: InfoVAE and InfoGAN

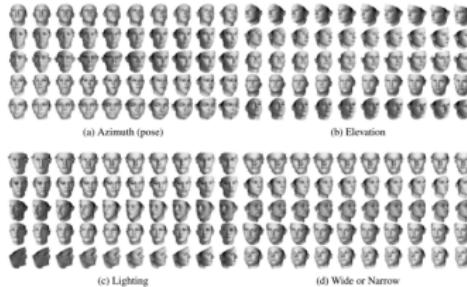
- Explicitly add a mutual information term to the objective

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

- 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) learns disentangled representations of the data



- Many quantitative metrics: Beta-VAE metric (Higgins et al., 2017), Factor-VAE metric, Mutual Information Gap, SAP score, DCI disentanglement, Modularity. None is perfect yet.

# 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