Generative Adversarial Networks

Volodymyr Kuleshov

Cornell Tech

Lecture 9

Announcements

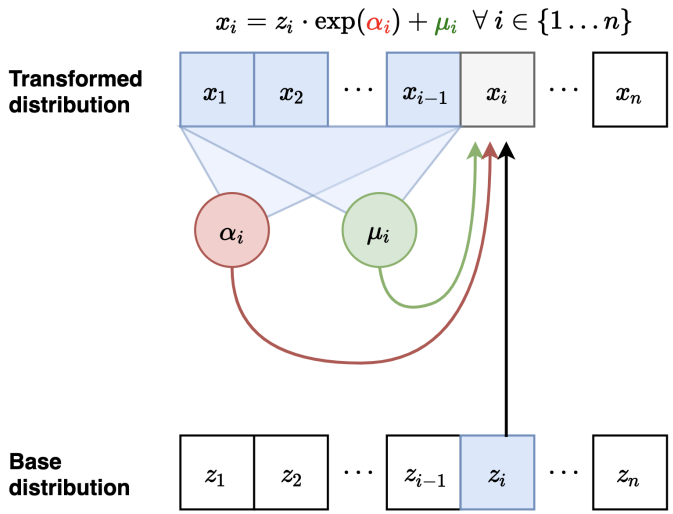
- Assignment 2 is out!
  - Programming assignment in PyTorch
  - Use starter code. Submit a PDF for other answers.
  - Code can run on CPU.
- Presentation topics list is on Piazza
- Have freed up two slots for presentations for the remaining teams.
- Submission link for the project proposal is open on Gradescope.

Lecture Outline

1. Recap of Normalizing Flows
   - IAF vs. MAF
   - Model Distillation and Parallel Wavenet
2. Towards Likelihood-Free Learning
   - Motivation
   - Two-Sample Tests
   - Unsupervised Learning as Supervised Learning
3. Generative Adversarial Networks
   - Definition
   - Objective Functions
   - Optimization Issues

Masked Autoregressive Flow (MAF)

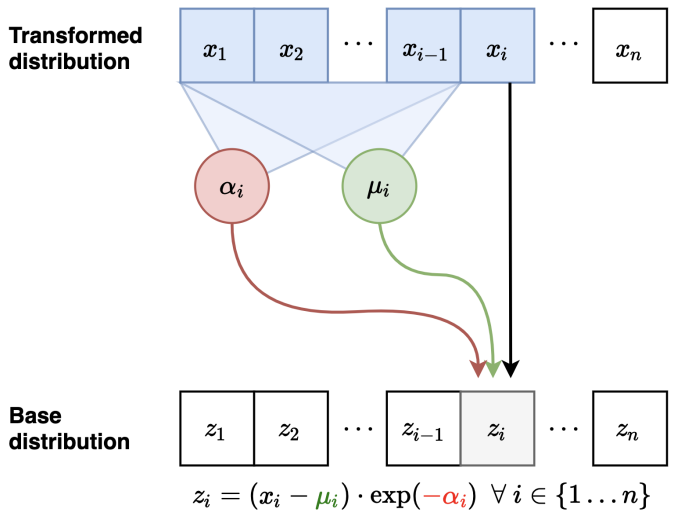
Masked Autoregressive Flow (MAF) is a bijective normalizing flow transformation $\mathbf{f} : X \to Z$ that implements this intuition:



- Forward mapping from $\mathbf{z} \mapsto \mathbf{x}$:
  - Let $x_1 = \exp(\alpha_1) z_1 + \mu_1$. Compute $\mu_2(x_1), \alpha_2(x_1)$
  - Let $x_2 = \exp(\alpha_2) z_2 + \mu_2$. Compute $\mu_3(x_1, x_2), \alpha_3(x_1, x_2)$
- Sampling is sequential and slow (like autoregressive): $O(n)$ time

Figure adapted from Eric Jang's blog

Masked Autoregressive Flow (MAF)

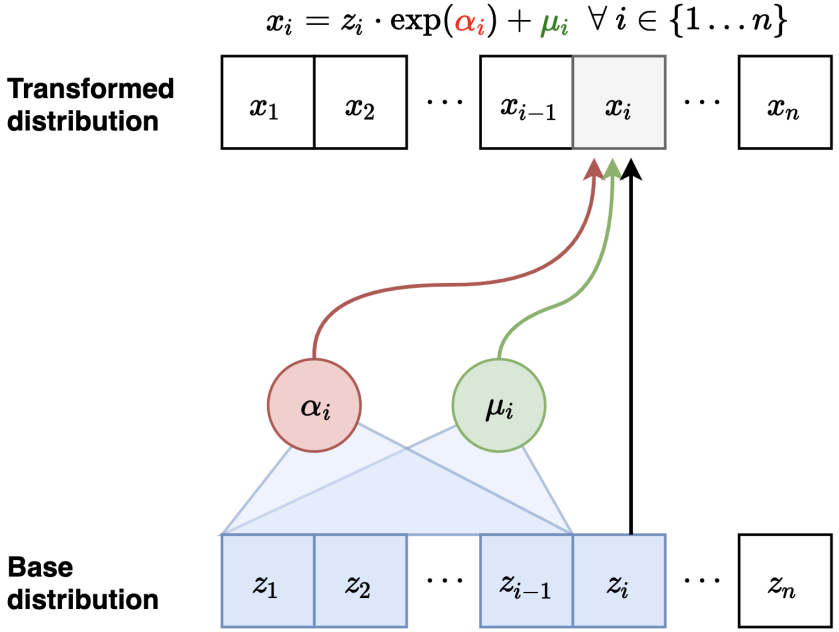


- Inverse mapping from $\mathbf{x} \mapsto \mathbf{z}$:
  - Compute all μ_i, α_i (can be done in parallel using e.g., MADE)
  - Let $z_1 = (x_1 - \mu_1) / \exp(\alpha_1)$ (scale and shift)
  - Let $z_2 = (x_2 - \mu_2) / \exp(\alpha_2)$
  - Let $z_3 = (x_3 - \mu_3) / \exp(\alpha_3)$ ...
- Jacobian is lower diagonal, hence determinant can be computed efficiently
- Likelihood evaluation is easy and parallelizable (like MADE)

Figure adapted from Eric Jang's blog

Inverse Autoregressive Flow (IAF)

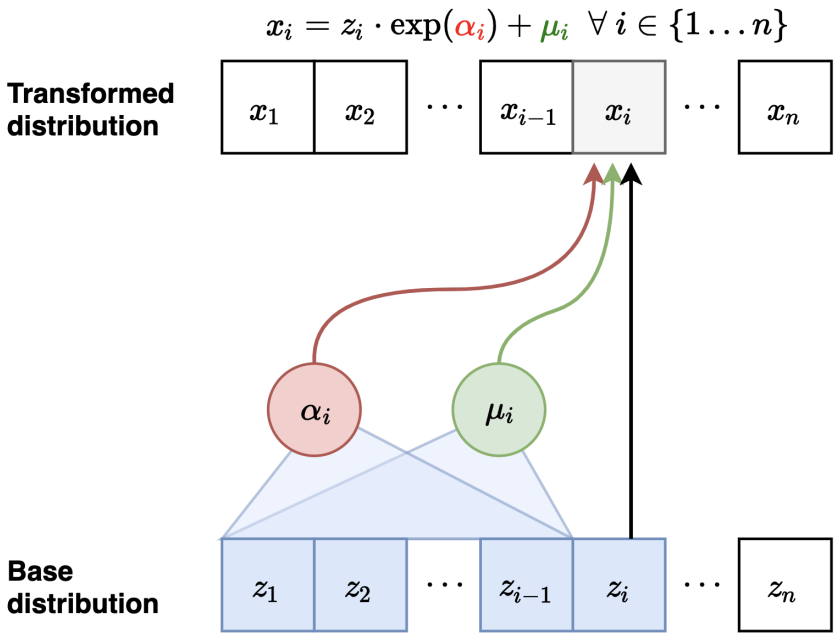
Inverse Autoregressive Flow (IAF) is a bijective normalizing flow transformation $\mathbf{f} : X \to Z$ that implements the opposite sampling approach:



- Forward mapping from $\mathbf{z} \mapsto \mathbf{x}$ (parallel):
  - Sample $z_i \sim \mathcal{N}(0, 1)$ for $i = 1, \cdots, n$
  - Compute all $\mu_i(z_{<i}), \alpha_i(z_{<i})$ (can be done in parallel)
  - Let $x_1 = \exp(\alpha_1) z_1 + \mu_1$
  - Let $x_2 = \exp(\alpha_2) z_2 + \mu_2$ ...

Figure adapted from Eric Jang's blog

Inverse Autoregressive Flow (IAF)



- Inverse mapping from $\mathbf{x} \mapsto \mathbf{z}$ (sequential):
  - Let $z_1 = (x_1 - \mu_1)/\exp(\alpha_1)$. Compute $\mu_2(z_1), \alpha_2(z_1)$
  - Let $z_2 = (x_2 - \mu_2)/\exp(\alpha_2)$. Compute $\mu_3(z_1, z_2), \alpha_3(z_1, z_2)$
- Fast to sample from, slow to evaluate likelihoods of data points (train)
- Note: Fast to evaluate likelihoods of a generated point (cache $z_1, z_2, \dots, z_n$)

Figure adapted from Eric Jang's blog

IAF is inverse of MAF

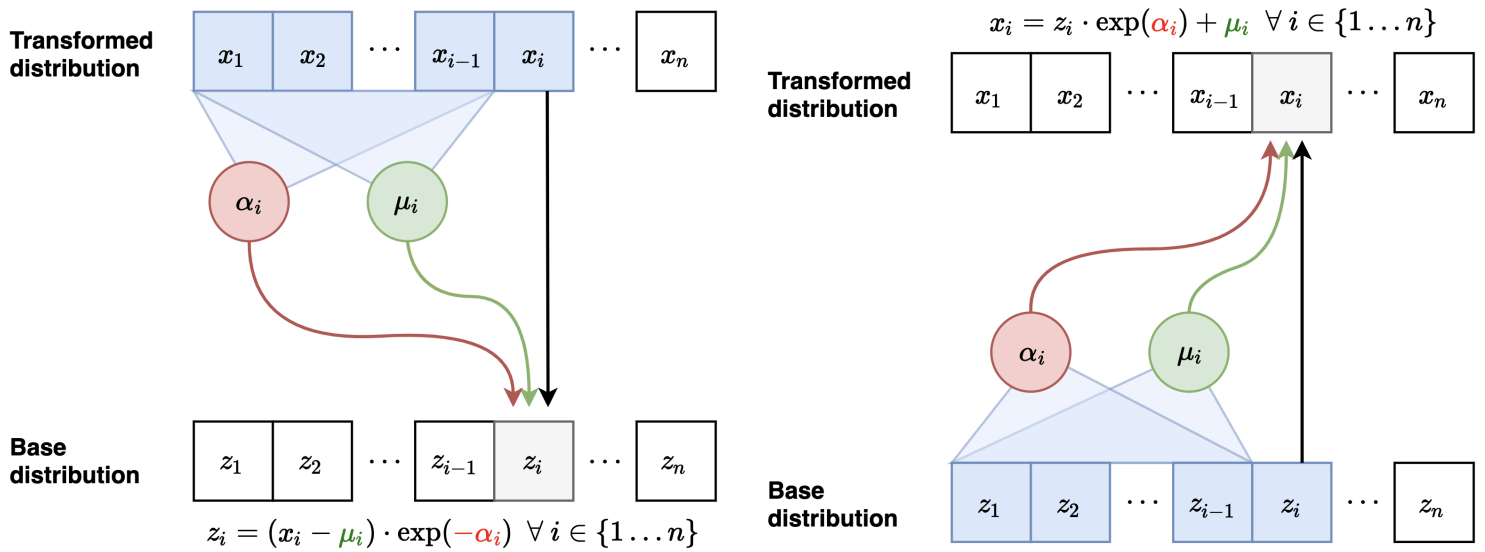


Figure: Inverse pass of MAF (**left**) vs. Forward pass of IAF (**right**)

- Interchanging **z** and **x** in the inverse transformation of MAF gives the forward transformation of IAF
- Similarly, forward transformation of MAF is inverse transformation of IAF

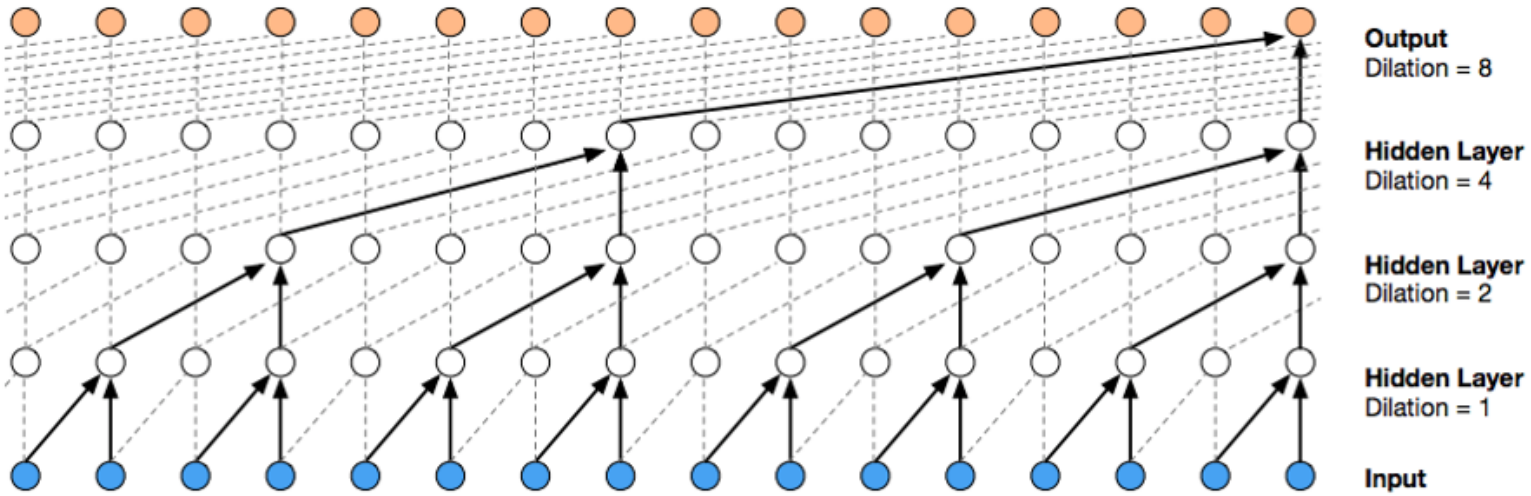
Figure adapted from Eric Jang's blog

IAF vs. MAF

- Computational tradeoffs
  - MAF: Fast likelihood evaluation, slow sampling
  - IAF: Fast sampling, slow likelihood evaluation
- MAF more suited for training based on MLE, density estimation
- IAF more suited for real-time generation
- Can we get the best of both worlds?

Recall: WaveNet (van den Oord et al., 2016)

State of the art model for speech:



Dilated convolutions increase the receptive field: kernel only touches the signal at every 2^d entries.

Accelerating Wavenet

- **Challenge:** How to make sampling fast?
- **Solution:** Two part training with a teacher and student model
  - Teacher is parameterized by MAF. Teacher can be efficiently trained via MLE
  - Once teacher is trained, initialize a student model parameterized by IAF. Student model cannot efficiently evaluate density for external datapoints but allows for efficient sampling
- **Key observation**: IAF can also efficiently evaluate densities of its own generations (via caching the noise variates $z_1, z_2, \ldots, z_n$)

Model Distillation

- **Probability density distillation**: Student distribution is trained to minimize the KL divergence between student (s) and teacher (t)

$$D_{\mathrm{KL}}(s, t) = E_{\mathbf{x}\sim s}[\log s(\mathbf{x}) - \log t(\mathbf{x})]$$

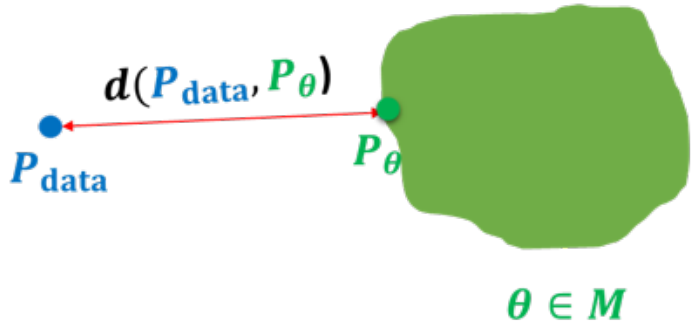
- Evaluating and optimizing Monte Carlo estimates of this objective requires:
  - Samples $\mathbf{x}$ from student model (IAF)
  - Density of $\mathbf{x}$ assigned by student model
  - Density of $\mathbf{x}$ assigned by teacher model (MAF)
- All operations above can be implemented efficiently

Parallel Wavenet: Overall algorithm

- Training
  - Step 1: Train teacher model (MAF) via MLE
  - Step 2: Train student model (IAF) to minimize KL divergence with teacher
- Test-time: Use student model for testing
- Improves sampling efficiency over original Wavenet (vanilla autoregressive model) by 1000x!

Recap

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

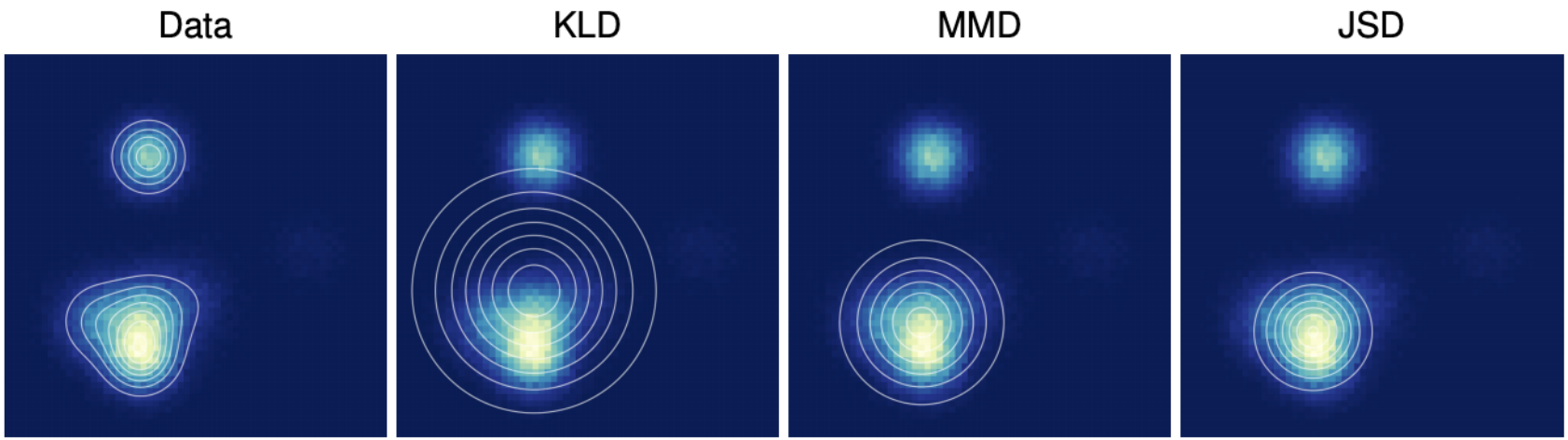


- Model families
  - Autoregressive Models: $p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i|\mathbf{x}_{<i})$
  - Variational Autoencoders: $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$
  - Normalizing Flow Models: $p_X(\mathbf{x}; \theta) = p_Z\left(\mathbf{f}_\theta^{-1}(\mathbf{x})\right) \left|\det\left(\frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right)\right|$
- All the above families are based on maximizing likelihoods (or approximations)
- Is the likelihood the right objective for measuring the similarity of a model to data?

Towards likelihood-free learning

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



Towards likelihood-free learning

- **Example:** Great test log-likelihoods, poor samples. E.g., For a discrete noise mixture model $p_\theta(\mathbf{x}) = 0.01 p_{\text{data}}(\mathbf{x}) + 0.99 p_{\text{noise}}(\mathbf{x})$
  - 99% of the samples are just noise
  - Taking logs, we get a lower bound

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

  - 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)

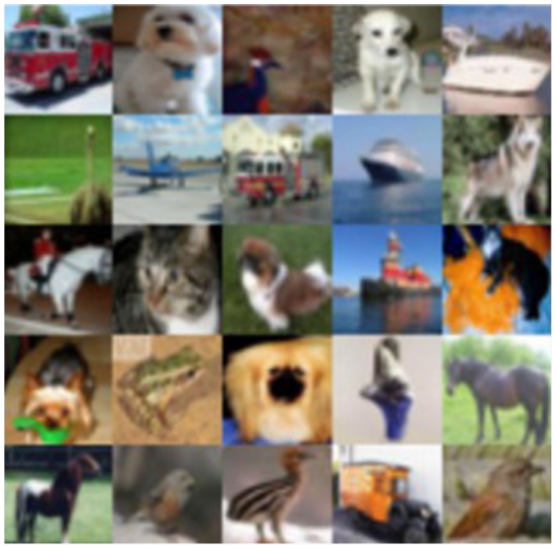
$$E_{p_{\text{data}}}[\log p_{\text{data}}(\mathbf{x}))] \geq E_{p_{\text{data}}}[\log p_\theta(\mathbf{x})]$$

  - 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

Towards likelihood-free learning

- **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

Comparing distributions via samples



vs.

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

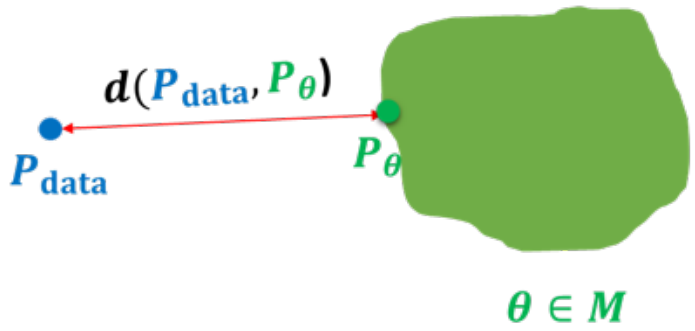
Given a finite set of samples from two distributions $S_1 = \{\mathbf{x} \sim P\}$ and $S_2 = \{\mathbf{x} \sim Q\}$, how can we tell if these samples are from the same distribution? (i.e., $P = Q$?)

Two-sample tests

- Given $S_1 = \{\mathbf{x} \sim P\}$ and $S_2 = \{\mathbf{x} \sim Q\}$, a **two-sample test** considers the following hypotheses
  - Null hypothesis H_0: $P = Q$
  - Alternate hypothesis H_1: $P \neq Q$
- Test statistic T compares S_1 and S_2 e.g., difference in means, variances of the two sets of samples
- If T is less than a threshold α, then accept H_0 else reject it
- **Key observation:** Test statistic is **likelihood-free** since it does not involve the densities P or Q (only samples)

Generative modeling and two-sample tests

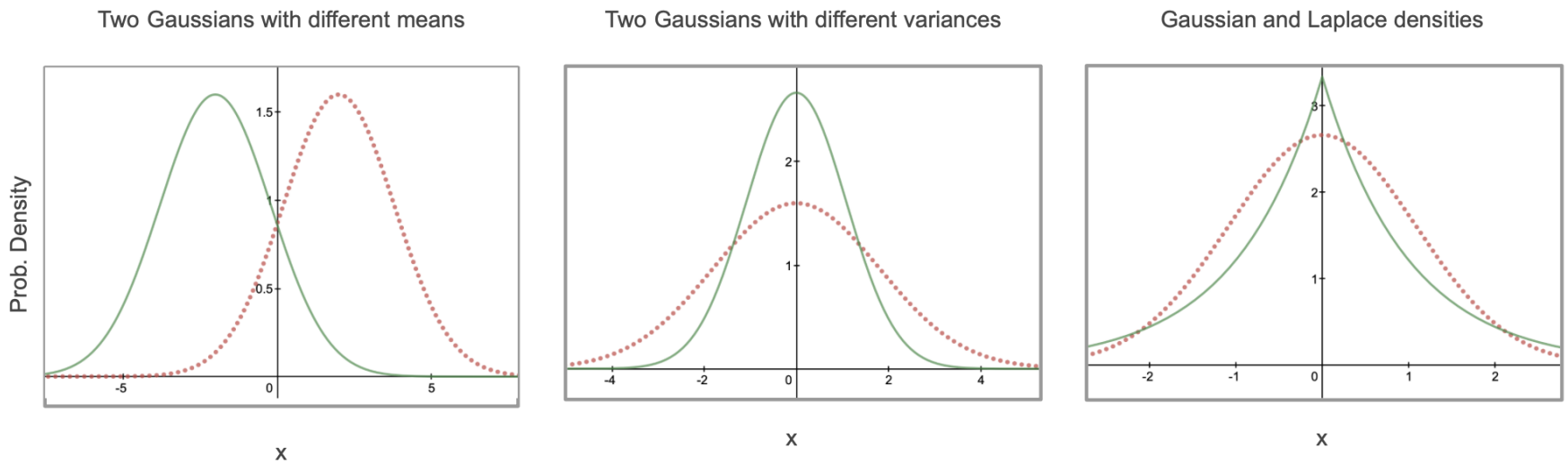
$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



- Apriori we assume direct access to $S_1 = \mathcal{D} = \{\mathbf{x} \sim p_{\text{data}}\}$
- In addition, we have a model distribution p_θ
- Assume that the model distribution permits efficient sampling (e.g., directed models). Let $S_2 = \{\mathbf{x} \sim p_\theta\}$
- **Alternate notion of distance between distributions:** Train the generative model to minimize a two-sample test objective between S_1 and S_2

Two-Sample Test via a Discriminator

- Finding a two-sample test objective in high dimensions is hard



- In the generative model setup, we know that S_1 and S_2 come from different distributions p_{data} and p_θ respectively
- **Key idea: Learn** a statistic that **maximizes** a suitable notion of distance between the two sets of samples S_1 and S_2

Unsupervised Learning as Supervised Learning

Consider balanced mixture of distributions $P(X|Y=0)$ and $P(X|Y=1)$.

- We have

$$P(Y=1|X) = \frac{P(X|Y=1)P(Y=1)}{P(X)} = \frac{P(X|Y=1)}{P(X|Y=0)+P(X|Y=1)}$$
$$= \frac{1}{1+\frac{P(X|Y=0)}{P(X|Y=1)}} = \sigma\left(\log \frac{P(X|Y=0)}{P(X|Y=1)}\right)$$

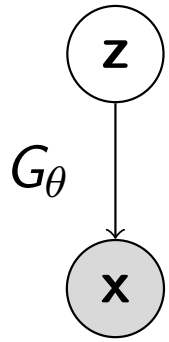
- Hence, we can use logistic regression trained on $(X, Y) \sim P$ pairs to estimate the log odds

$$\log \frac{P(X|Y=0)}{P(X|Y=1)}.$$

- Old idea: can be used for outlier detection, density estimation, noise-contrastive learning, etc.

Generative Adversarial Networks

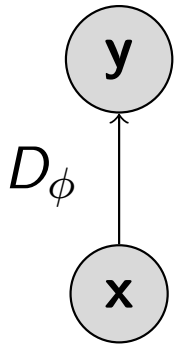
- A two player minimax game between a **generator** and a **discriminator**



- **Generator**
  - Directed, latent variable model with a deterministic mapping between **z** and **x** given by G_θ
  - Minimizes a two-sample test objective (in support of the null hypothesis $p_{\text{data}} = p_\theta$)

Generative Adversarial Networks

- A two player minimax game between a generator and a discriminator



- **Discriminator**
  - Any function (e.g., neural network) which tries to distinguish "real" samples from the dataset and "fake" samples generated from the model
  - Maximizes the two-sample test objective (in support of the alternate hypothesis $p_{\text{data}} \neq p_\theta$)

Example of GAN objective

- **Training objective for discriminator**:

$$\max_D V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

- For a fixed generator G, the discriminator is performing binary classification with the cross entropy objective
  - Assign probability 1 to true data points $\mathbf{x} \sim p_{\text{data}}$
  - Assing probability 0 to fake samples $\mathbf{x} \sim p_G$
- Optimal discriminator

$$D^*_G(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} = \sigma\left(\log \frac{p_{\text{data}}(\mathbf{x})}{p_G(\mathbf{x})}\right)$$

Example of GAN objective

- **Training objective for generator**:

$$\min_G V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

- For the optimal discriminator $D^*_G(\cdot)$, we have

$$\begin{aligned}
& V(G, D^*_G(\mathbf{x})) \\
&= E_{\mathbf{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}\right] + E_{\mathbf{x} \sim p_G}\left[\log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}\right] \\
&= E_{\mathbf{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}}\right] + E_{\mathbf{x} \sim p_G}\left[\log \frac{p_G(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}}\right] - \log 4 \\
&= \underbrace{D_{KL}\left[p_{\text{data}}, \frac{p_{\text{data}} + p_G}{2}\right] + D_{KL}\left[p_G, \frac{p_{\text{data}} + p_G}{2}\right]}_{2 \times \text{Jenson-Shannon Divergence (JSD)}} - \log 4 \\
&= 2 D_{JSD}[p_{\text{data}}, p_G] - \log 4
\end{aligned}$$

Jenson-Shannon Divergence

- Also called as the symmetric KL divergence

$$D_{JSD}[p, q] = \frac{1}{2}\left(D_{KL}\left[p, \frac{p+q}{2}\right] + D_{KL}\left[q, \frac{p+q}{2}\right]\right)$$

- Properties
  - $D_{JSD}[p, q] \geq 0$
  - $D_{JSD}[p, q] = 0$ iff $p = q$
  - $D_{JSD}[p, q] = D_{JSD}[q, p]$
  - $\sqrt{D_{JSD}[p, q]}$ satisfies triangle inequality $\rightarrow$ Jenson-Shannon Distance
- Optimal generator for the JSD/Negative Cross Entropy GAN

$$p_G = p_{\text{data}}$$

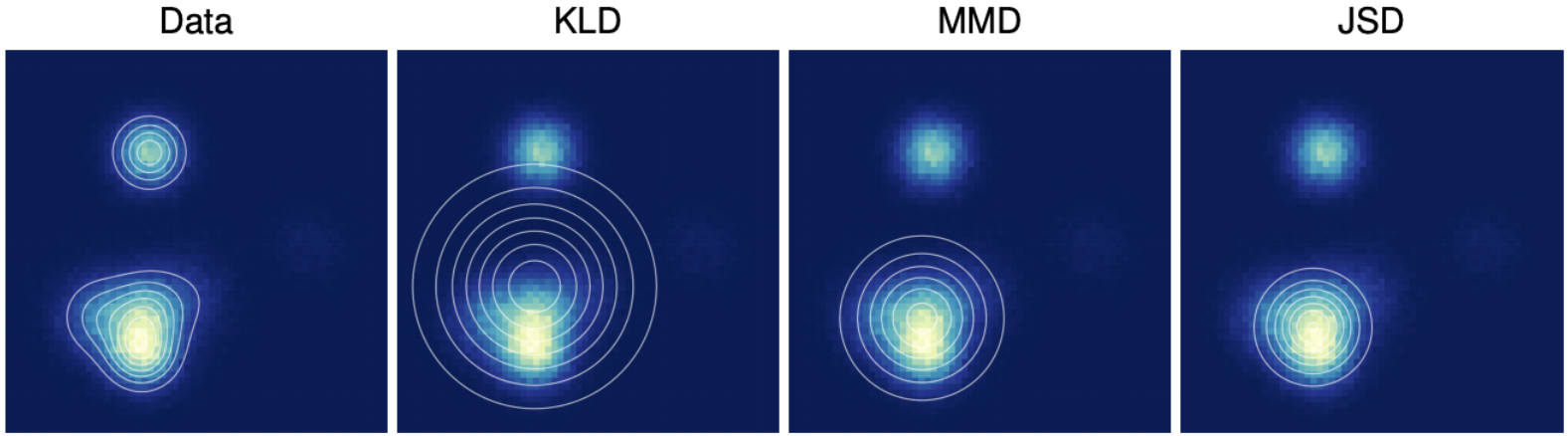
- For the optimal discriminator $D^*_{G^*}(\cdot)$ and generator $G^*(\cdot)$, we have

$$V(G^*, D^*_{G^*}(\mathbf{x})) = -\log 4$$

Jenson-Shannon Divergence

The Jenson-Shannon divergence is mode-seeking.

- Consider a multi-modal data distribution that we are trying to approximating with a uni-model estimator.



- The KL divergence (log-likelihood objective) tries to average both modes. The JSD objective favors fitting one mode well. Recall:

$$\mathbf{D}(P_{\text{data}}||P_\theta) = \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log\left(\frac{P_{\text{data}}(\mathbf{x})}{P_\theta(\mathbf{x})}\right)\right] = \sum_{\mathbf{x}} P_{\text{data}}(\mathbf{x}) \log \frac{P_{\text{data}}(\mathbf{x})}{P_\theta(\mathbf{x})}$$

The GAN training algorithm

- Sample minibatch of m training points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$ from $\mathcal{D}$
- Sample minibatch of m noise vectors $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)}$ from p_z
- Update the generator parameters θ by stochastic gradient **descent**

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m} \nabla_\theta \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))$$

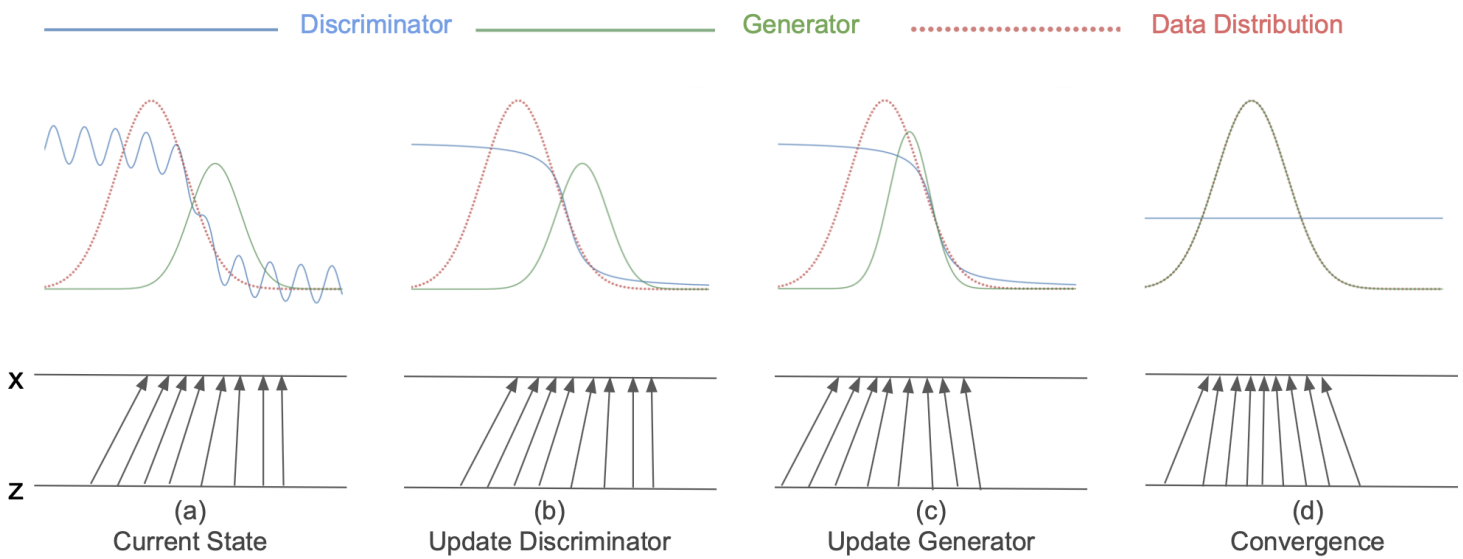
- Update the discriminator parameters ϕ by stochastic gradient **ascent**

$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m} \nabla_\phi \sum_{i=1}^{m} [\log D_\phi(\mathbf{x}^{(i)}) + \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))]$$

- Repeat for fixed number of epochs

Alternating optimization in GANs

$$\min_{\theta} \max_{\phi} V(G_\theta, D_\phi) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$



Frontiers in GAN research

- GANs have been successfully applied to several domains and tasks
- However, working with GANs can be very challenging in practice
  - Unstable optimization
  - Mode collapse
  - Evaluation
- Many bag of tricks applied to train GANs successfully

Image Source: Ian Goodfellow. Samples from Goodfellow et al., 2014, Radford et al., 2015, Liu et al., 2016, Karras et al., 2017, Karras et al., 2018

Optimization challenges

- **Theorem (informal):** If the generator updates are made in function space and discriminator is optimal at every step, then the generator is guaranteed to converge to the data distribution
- **Unrealistic assumptions**!
- In practice, the generator and discriminator loss keeps oscillating during GAN training

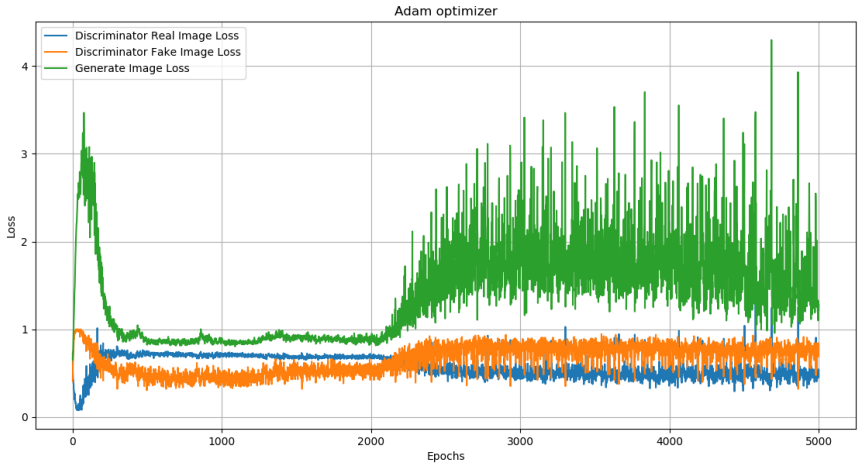
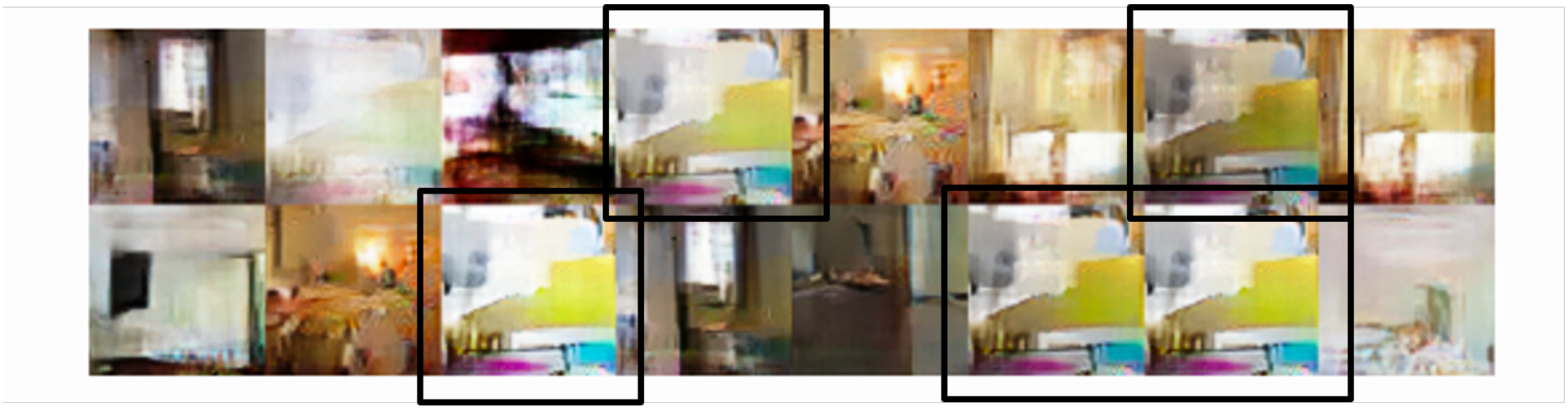


Figure: *

Source: Mirantha Jayathilaka

Mode Collapse

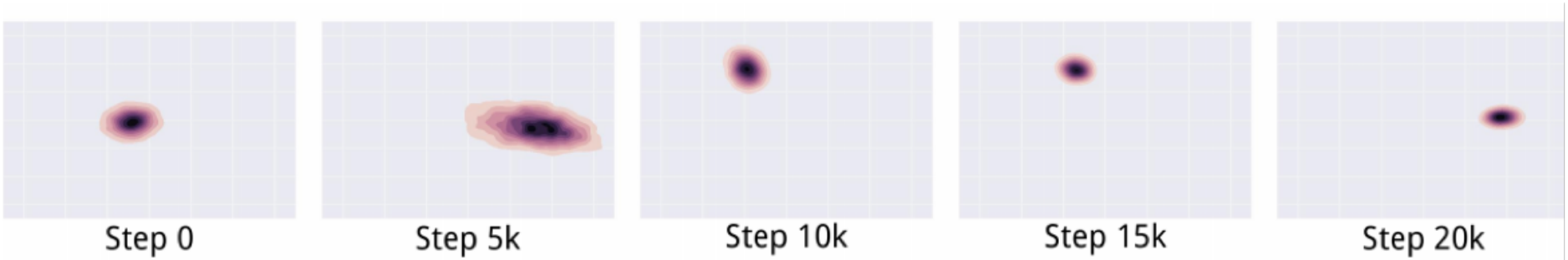
- GANs are notorious for suffering from **mode collapse**
- Intuitively, this refers to the phenomena where the generator of a GAN collapses to one or few samples (dubbed as "modes")



Arjovsky et al., 2017

Mode Collapse

- True distribution is a mixture of Gaussians



Source: Metz et al., 2017

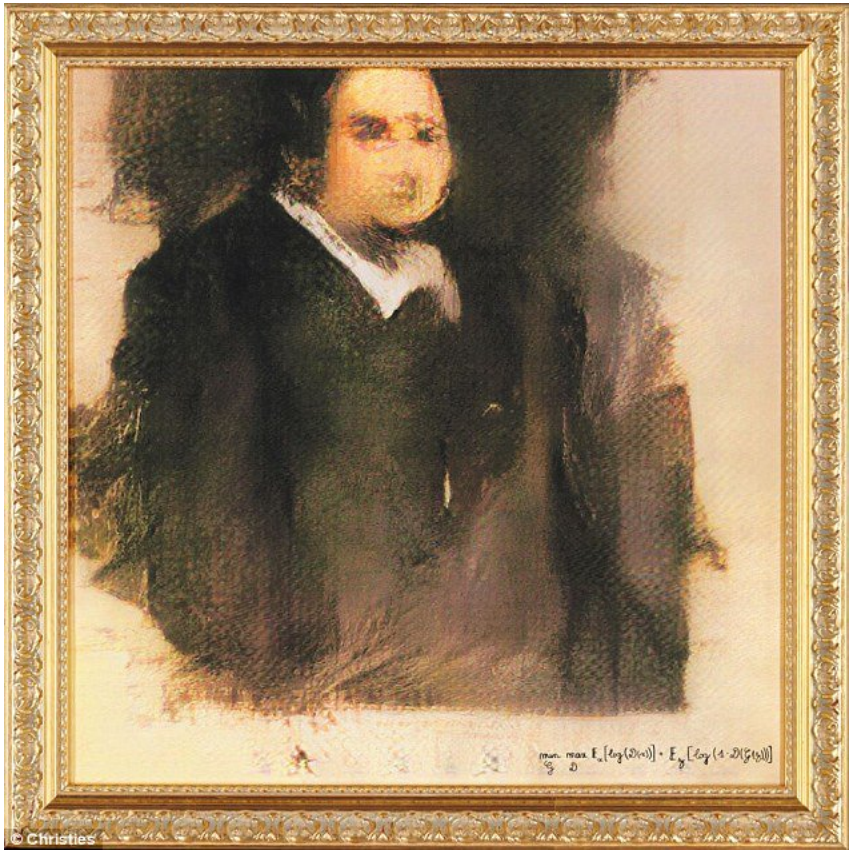
- The generator distribution keeps oscillating between different modes

Mode Collapse

Source: Metz et al., 2017

- Fixes to mode collapse are mostly empirically driven: alternate architectures, adding regularization terms, injecting small noise perturbations etc.
- `https://github.com/soumith/ganhacks`
  How to Train a GAN? Tips and tricks to make GANs work by Soumith Chintala

Beauty lies in the eyes of the discriminator



GAN generated art auctioned at Christie's.

Expected Price: $\$7,000 - \$10,000$

True Price: $\$432,500$

Summary of GAN Models

- GAN Pros:
  - Very high-quality samples.
  - Can optimize a wide range of divergences between probabilities (next lecture)
  - Broadly applicable: only need sampling from G!
- GAN Cons:
  - Only works for continuous variables
  - Difficult to train
  - Suffers from mode collapse