Statistical Models & Computing Methods

Lecture 18: Generative Models – II

Cheng Zhang

School of Mathematical Sciences, Peking University

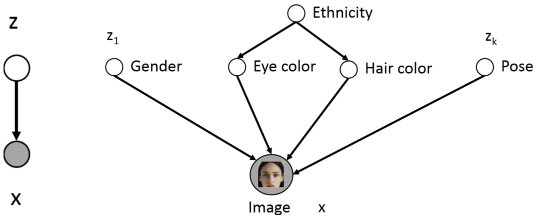
December 08, 2023

Recap of Autoregressive Models

- Autoregressive models:
  - Chain rule based factorization is fully general
  - Compact representation via conditional independence and /or neural parameterization
- Pros:
  - Easy to evaluate likelihoods
  - Easy to train
- Cons:
  - Requires an ordering
  - Generation is sequential
  - Cannot learn features in an unsupervised way

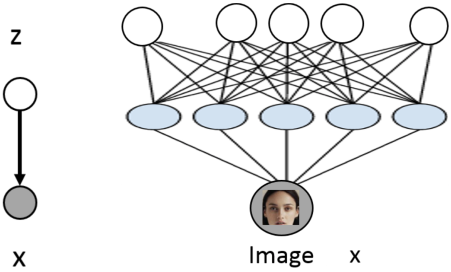
Latent Variable Models: Motivation

- Lots of variability in images x due to gender, eye color, hair color, pose, etc. However, unless images are annotated, these factors of variation are not explicitly available (latent)
- Idea: explicitly model these factors using latent variables z



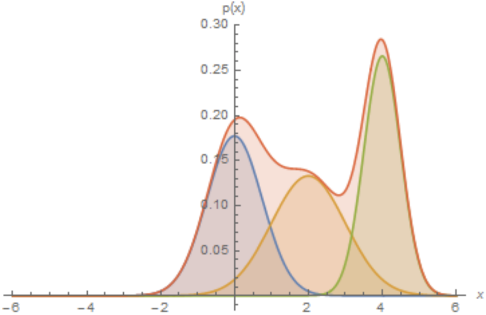
- Only shaded variables x are observed in the data (pixel values)
- Latent variables z correspond to high level features
  - If z chosen properly, $p(x|z)$ could be much simpler than $p(x)$
  - If we had trained this model, then we could identify features via $p(z|x)$, e.g., $p(\text{EyeColor} = \text{Blue}|x)$
- **Challenge**: Very difficult to specify these conditionals by hand

Deep Latent Variable Models

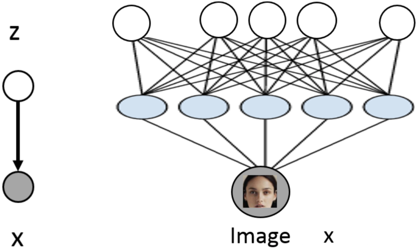


- $z \sim \mathcal{N}(0, I)$
- $p(x|z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))$ where $\mu_\theta, \Sigma_\theta$ are neural networks
- Hope that after training, z will correspond to meaningful latent factors of variation (features). Unsupervised representation learning
- As before, features can be computed via $p(z|x)$

Combine simple models into a more complex and expressive one



$$p(x) = \sum_z p(x,z) = \sum_z p(z)p(x|z) = \sum_{k=1}^{K} p(z=k)\mathcal{N}(x;\mu_k,\Sigma_k)$$



A mixture of infinite many Gaussians

- $z \sim \mathcal{N}(0, I)$
- $p(x|z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))$ where $\mu_\theta, \Sigma_\theta$ are neural networks
- Even though $p(x|z)$ is simple, the marginal $p(x)$ could be very complex/flexible

$$p_\theta(x) = \int_z p_\theta(x, z) dz = \int_z p_\theta(x|z) p(z) dz$$

- Allow us to define complex models $p(x)$ in terms of simple building blocks $p(x|z)$
- Natural for unsupervised learning tasks (clustering, unsupervised representation learning, etc)
- No free lunch: much more difficult to learn compared to fully observed autoregressive models

First Attempt: Naive Monte Carlo

$$p_\theta(x) = \mathbb{E}_{z\sim p(z)} p_\theta(x|z), \quad \nabla_\theta p_\theta(x) = \mathbb{E}_{z\sim p(z)} \nabla_\theta p_\theta(x|z)$$

We can use Monte Carlo estimate for the marginal likelihood and its gradient

- Sample $z^{(1)}, \cdots, z^{(k)}$ from the prior $p(z)$
- Approximate expectation with sample average

$$p_\theta(x) \approx \frac{1}{k}\sum_{i=1}^{k} p_\theta(x|z^{(i)}), \quad \nabla_\theta p_\theta(x) \approx \frac{1}{k}\sum_{i=1}^{k} \nabla_\theta p_\theta(x|z^{(i)})$$

Remark: work in theory but not in practice. For most $z \sim p(z)$, $p_\theta(x|z)$ is very low, i.e., mismatch between the prior and posterior. This leads to large variance for the Monte Carlo estimates. We need a clever way to select $z^{(i)}$ to reduce the variance of the estimator.

Second Attempt: Importance Sampling

We can use importance sampling to reduce the variance

$$p_\theta(x) = \int_z p_\theta(x|z)p(z)dz = \int_z q(z)\frac{p_\theta(x,z)}{q(z)}dz = \mathbb{E}_{z\sim q(z)}\frac{p_\theta(x,z)}{q(z)}$$

Similarly, we can use Monte Carlo estimate

- Sample $z^{(1)}, \cdots, z^{(k)}$ from the important distribution $q(z)$
- Approximate expectation with sample average

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

Remark: What is a good choice for $q(z)$?

- Evidence Lower Bound (ELBO)

$$
\begin{aligned}
\log p_\theta(x) &\geq \mathbb{E}_{z\sim q(z)} \log \frac{p_\theta(x,z)}{q(z)} \\
&= \mathbb{E}_{z\sim q(z)} \log p_\theta(x,z) - \mathbb{E}_{z\sim q(z)} \log q(z) \\
&= \mathbb{E}_{z\sim q(z)} \log p_\theta(x,z) + H(q)
\end{aligned}
$$

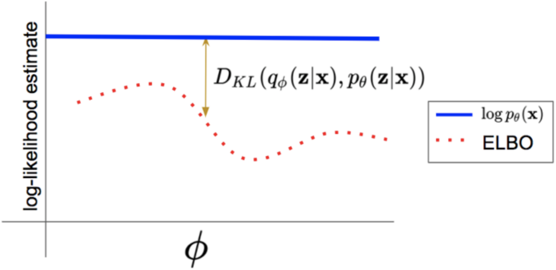
- Equality holds when $q(z) = p(z|x;\theta)$

$$\log p_\theta(x) = \mathbb{E}_{z\sim p(z|x;\theta)} \log p_\theta(x,z) + H(p(z|x;\theta))$$

  This is the E-step in EM!

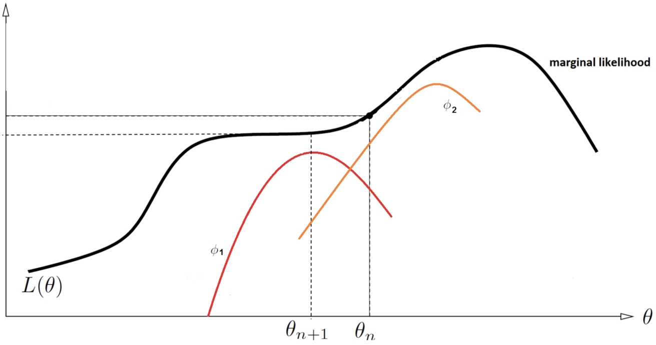
- In practice, $p(z|x,\theta)$ is usually intractable. We can find the "best" $q(z)$ by maximizing the ELBO in a parameterized family of $\{q_\phi(z) : \phi \in \Phi\}$

The Evidence Lower Bound



$$
\begin{aligned}
\log p_\theta(x) &\geq \int_z q_\phi(z|x) \log \frac{p_\theta(x,z)}{q_\phi(z|x)} = \mathcal{L}(x;\theta,\phi) \\
&= \mathcal{L}(x;\theta,\phi) + \mathrm{KL}(q_\phi(z|x) \| p(z|x;\theta))
\end{aligned}
$$

The better $q_\phi(z|x)$ can approximate the posterior $p(z|x;\theta)$, the closer ELBO will be to the $\log p_\theta(x)$. We then jointly optimize over θ and ϕ to maximize the ELBO over a dataset.



$\mathcal{L}(x;\theta,\phi_1)$ and $\mathcal{L}(x;\theta,\phi_2)$ are both lower bounds, we want to jointly optimize θ and ϕ.

- For each data point x, ELBO holds

$$\log p_\theta(x) \geq \int_z q_\phi(z|x) \log p_\theta(x, z) + H(q_\phi(z|x)) = \mathcal{L}(x; \theta, \phi)$$

- Maximum likelihood learning over the entire dataset

$$\ell(\theta; \mathcal{D}) = \sum_{x^i \in \mathcal{D}} \log p_\theta(x^i) \geq \sum_{x^i \in \mathcal{D}} \mathcal{L}(x^i; \theta, \phi^i)$$

- Therefore

$$\max_\theta \ell(\theta; \mathcal{D}) \geq \max_{\theta, \phi^1, \cdots, \phi^M} \sum_{i=1}^{M} \mathcal{L}(x^i; \theta, \phi^i)$$

- Note that we use different *variational parameters* ϕ^i for every data point x^i, because the true posterior $p_\theta(z|x^i)$ is different across data points x^i

- Assume $p_\theta(z, x^i)$ is close to $p_{\text{data}}(z, x^i)$. Suppose z captures information such as digit identity (label), style, etc. For simplicity, assume $z \in \{0, 1, \ldots, 9\}$
- Suppose $q_{\phi^i}(z)$ is a probability distribution over the hidden variable z parameterized by $\phi^i = (p_0, \ldots, p_9)$
- If $\phi^i = (0, 0, 0, 1, \ldots, 0)$, is $q_{\phi^i}(z)$ a good approximation of $p_\theta(z|x^1)$(x^1 is the leftmost datapoint)? Yes
- If $\phi^i = (0, 0, 0, 1, \ldots, 0)$, is $q_{\phi^i}(z)$ a good approximation of $p_\theta(z|x^3)$(x^3 is the rightmost datapoint)? No
- For each x^i, need to find a good $\phi^{i,*}$ via optimization, can be expensive

Learning via SVI

- Optimizing $\sum_{x^i \in \mathcal{D}} \mathcal{L}(x^i; \theta, \phi^i)$ as a function of $\theta, \phi^1, \ldots, \phi^M$ using stochastic gradient ascent

$$L(\mathcal{D}; \theta, \phi^{1:M}) = \sum_{i=1}^{M} \mathbb{E}_{q_{\phi^i}(z^i)} \left(\log p_\theta(x^i, z) - \log q_{\phi^i}(z^i) \right)$$

1. Initialize $\theta, \phi^1, \cdots, \phi^M$
2. Randomly sample a data point x^i from $\mathcal{D}$
3. Optimize $\mathcal{L}(x^i; \theta, \phi^i)$ as a function of ϕ^i, e.g., local gradient update
4. Compute $\nabla_\theta \mathcal{L}(x^i; \theta, \phi^{i,*})$
5. Update θ in the gradient direction. Go to step 2

- How to compute the gradients? Often no close form solution for the expectations. Use Monte Carlo estimates!

$$\mathcal{L}(x;\theta,\phi) = \mathbb{E}_{q_\phi(z)}\left(\log p_\theta(x,z) - \log q_\phi(z)\right)$$

- Similarly as in VI, we assume $q_\phi(z)$ is tractable, i.e., easy to sample from and evaluate
- Suppose $z^1, \ldots, z^k$ are samples from $q_\phi(z)$
- The gradient with respect to θ is easy

$$\begin{aligned}
\nabla_\theta \mathcal{L}(x;\theta,\phi) &= \nabla_\theta \mathbb{E}_{q_\phi(z)}\left(\log p_\theta(x,z) - \log q_\phi(z)\right) \\
&= \mathbb{E}_{q_\phi(z)} \nabla_\theta \log p_\theta(x,z) \\
&\approx \frac{1}{k}\sum_{i=1}^{k} \nabla_\theta \log p_\theta(x,z^i)
\end{aligned}$$

- The gradient with respect to ϕ is more complicated because the expectation depends on ϕ
- We can use score function estimator (or REINFORCE) with *control variates*. When $q_\phi(z)$ is reparameterizable, we can also use the reparameterization trick.
- If these exists g_ϕ and q_ϵ, s.t. $z = g_\phi(\epsilon), \epsilon \sim q_\epsilon \Rightarrow z \sim q_\phi(z)$

$$\begin{aligned}\nabla_\phi \mathcal{L}(x;\theta,\phi) &= \nabla_\phi \mathbb{E}_{q_\epsilon(\epsilon)} \left(\log p_\theta(x, g_\phi(\epsilon)) - \log q_\phi(g_\phi(\epsilon))\right) \\ &= \mathbb{E}_{q_\epsilon(\epsilon)} \left(\nabla_\phi \log p_\theta(x, g_\phi(\epsilon)) - \nabla_\phi \log q_\phi(g_\phi(\epsilon))\right) \\ &\approx \frac{1}{k}\sum_{i=1}^{k} \left(\nabla_\phi \log p_\theta(x, g_\phi(\epsilon^i)) - \nabla_\phi \log q_\phi(g_\phi(\epsilon^i))\right)\end{aligned}$$

where $\epsilon^i \sim q_\epsilon(\epsilon), i = 1, \ldots, k$

- Example: $z = \mu + \sigma\epsilon, \epsilon \sim \mathcal{N}(0,1) \Leftrightarrow z \sim \mathcal{N}(\mu, \sigma^2) = q_\phi(z)$

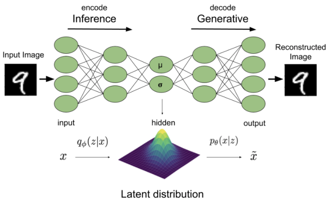
$$\max_{\theta} \ell(\theta; \mathcal{D}) \geq \max_{\theta, \phi^{1:M}} \sum_{i=1}^{M} \mathcal{L}(x^i; \theta, \phi^i)$$

- So far we have used a set of variational parameters ϕ^i for each data point x^i. Unfortunately, this does not scale to large datasets.
- **Amortization**: Learn a single parameteric function f_λ that maps each x to a set of variational parameters. Like doing regression $x^i \mapsto \phi^{i,*}$
  - For example, if $q(z|x^i)$ are Gaussians with different means $\mu^1, \ldots, \mu^m$, we learn a single neural network f_λ mapping x^i to μ^i
- We approximate the posteriors $q(z|x^i)$ using this distribution $q_\lambda(z|x^i)$

Amortized Inference

- Assume $p_\theta(z, x^i)$ is close to $p_{\text{data}}(z, x^i)$. Suppose z captures information such as digit identity (label), style, etc.
- Suppose $q_{\phi^i}(z)$ is a probability distribution over the hidden variable z parameterized by ϕ^i
- For each x^i, need to find a good $\phi^{i,*}$ via optimization, expensive for large dataset
- Amortized Inference: learn how to map x^i to a good set of parameters ϕ^i via $q(z; f_\lambda(x^i))$. f_λ learns how to solve the optimization problem for you, jointly across all datapoints.
- In the literature, $q(z; f_\lambda(x^i))$ often denoted as $q_\phi(z|x^i)$

Autoencoder Perspective



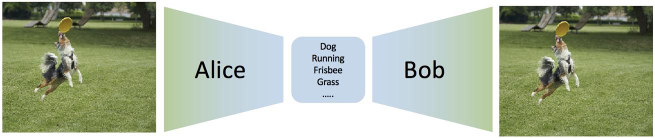
$$
\begin{aligned}
\mathcal{L}(x;\theta,\phi) &= \mathbb{E}_{q_\phi(z|x)}\left(\log p_\theta(x,z) - \log q_\phi(z|x)\right) \\
&= \mathbb{E}_{q_\phi(z|x)}\left(\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)\right) \\
&= \mathbb{E}_{q_\phi(z|x)} \log p(x|z;\theta) - \mathrm{KL}\left(q_\phi(z|x) \| p(z)\right)
\end{aligned}
$$

Take a data point x^i $\rightarrow$ Map it to $\hat{z}$ by sampling from $q_\phi(z|x^i)$ (encoder) $\rightarrow$ Reconstruct $\hat{x}$ by sampling from $p(x|\hat{z};\theta)$ (decoder)

What does the training objective $\mathcal{L}(x;\theta,\phi)$ do?

- First term encourages $\hat{x} \approx x^i$ (x^i likely under $p(x|\hat{z};\theta)$)
- Second term encourages $\hat{z}$ to be likely under the prior $p(z)$

Variational AutoEncoder

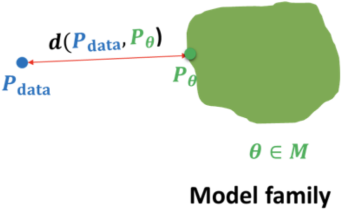


- Alice goes on a space mission and needs to send images to Bob. Given an image x^i, she (stochastically) compress it using $\hat{z} \sim q_\phi(z|x^i)$ obtaining a message $\hat{z}$. Alice sends the message $\hat{z}$ to Bob
- Given $\hat{z}$, Bob tries to reconstruct the image using $p_\theta(x|\hat{z})$
  - This scheme works well if $\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z)$ is large
  - The term $\mathrm{KL}\left(q_\phi(z|x) \| p(z)\right)$ forces the distribution over messages to have a specific shape $p(z)$. If Bob knows $p(z)$, he can generate realistic messages $\hat{z} \sim p(z)$ and the corresponding image, as if he had received them from Alice!

- Combine simple models to get a more flexible one (e.g., mixture of Gaussians)
- Directed model permits ancestral sampling (efficient generation): $z \sim p(z), \quad x \sim p_\theta(x|z)$
- However, log-likelihood is generally intractable, hence learning is difficult (compared to autoregressive models)
- Joint learning of a model (θ) and an amortized inference component ϕ to achieve tractability via ELBO optimization
- Latent representations for any x can be inferred via $q_\phi(z|x)$

Recap on Deep Generative Models

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



- Model families
  - Autoregressive Models: $p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_{<i})$
  - Variational Autoencoders: $p_\theta(x) = \int_z p_\theta(x, z) dz$
  - Normalizing Flow Models:
    $$p_X(x; \theta) = p_Z(f_\theta^{-1}(x)) \left| \det \left(\frac{\partial f_\theta^{-1}(x)}{\partial x} \right) \right|$$
- All the above families are based on maximizing likelihoods (or approximations, e.g., lower bound)
- Is the likelihood a good indicator of the quality of samples generated by the model?

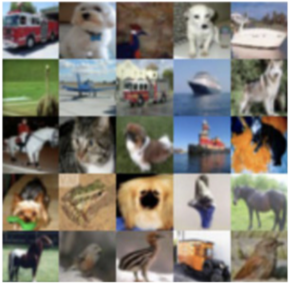
Sample Quality and Likelihood

- Optimal generative model will give best sample quality and highest test log-likelihood. However, in practice, high log-likelihoods $\neq$ good sample quality (Theis et al., 2016)
- Case 1: great test log-likelihoods, poor samples. Consider a mixture model $p_\theta(x) = 0.01 p_{\text{data}}(x) + 0.99 p_{\text{noise}}(x)$, we have

  $$\mathbb{E}_{p_{\text{data}}} \log p_{\text{data}}(x) \geq \mathbb{E}_{p_{\text{data}}} \log p_\theta(x) \geq \mathrm{E}_{p_{\text{data}}} \log p_{\text{data}}(x) - \log 100$$

  This means $\mathbb{E}_{p_{\text{data}}} \log p_\theta(x) \approx \mathrm{E}_{p_{\text{data}}} \log p_{\text{data}}(x)$ when the dimension of x is large.
- Case 2: great samples, poor test log-likelihoods. E.g., memorizing training set: samples look exactly like the training set; test set will have zero probability
- The above cases suggest that it might be useful to disentangle likelihoods and samples $\Rightarrow$ likelihood-free learning!

Comparing Distributions via Samples



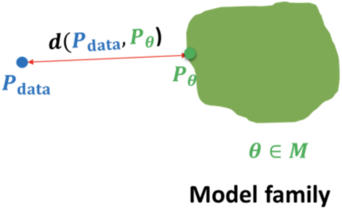
vs.

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

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

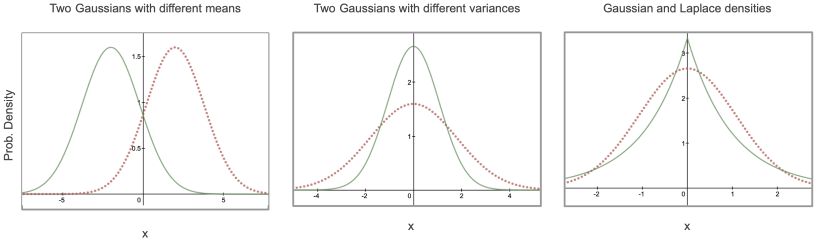
- Given $S_1 = \{x \sim P\}$ and $S_2 = \{x \sim Q\}$, a two-sample test considers the following hypotheses
  - Null hypothesis $H_0 : P = Q$
  - Alternative hypothesis $H_1 : p \neq Q$
- Test statistic T compares S_1 and S_2, e.g., difference in means, variances of the two sets of samples
- If T is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

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



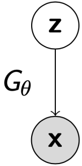
- Suppose we have direct access to the data set $S_1 = \mathcal{D} = \{x \sim p_{\text{data}}\}$
- Now assume that the model distribution p_θ permits efficient sampling (e.g., directed models). Let $S_2 = \{x \sim p_\theta\}$
- Use a two-sample test objective to measure the distance between distributions and train the generative model p_θ to minimize this distance between S_1 and S_2

Two-Sample Test via a Discriminator



- Finding a two-sample test objective in high dimensions is non-trivial
- 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

The **generator** and **discriminator** play a minimax game!

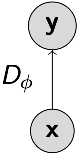


Generator

- 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 $p_{\text{data}} = p_\theta$

A Two Player Game

The **generator** and **discriminator** play a minimax game!



Discriminator

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

Discriminator Training Objective

- Training objective for discriminator:

$$\max_D V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x \sim p_G} \log(1 - D(x))$$

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

$$D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

- Training Objective for generator:

$$\min_G V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x\sim p_G} \log(1 - D(x))$$

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

$$\begin{aligned}
V(G, D_G^*) &= \mathbb{E}_{x\sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} + \mathbb{E}_{x\sim p_G} \log \frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)} \\
&= \mathbb{E}_{x\sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x)+p_G(x)}{2}} + \mathbb{E}_{x\sim p_G} \log \frac{p_G(x)}{\frac{p_{\text{data}}(x)+p_G(x)}{2}} - \log 4 \\
&= \mathrm{KL}\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_G}{2}\right) + \mathrm{KL}\left(p_G \middle\| \frac{p_{\text{data}} + p_G}{2}\right) - \log 4
\end{aligned}$$

- The sum of KL in the above equation is known as Jensen-Shannon divergence (JSD)

Jensen-Shannon Divergence

$$\mathrm{JSD}(p, q) = \mathrm{KL}\left(p \,\middle\|\, \frac{p+q}{2}\right) + \mathrm{KL}\left(q \,\middle\|\, \frac{p+q}{2}\right)$$

- Properties
  - $\mathrm{JSD}(p, q) \geq 0$
  - $\mathrm{JSD}(p, q) = 0$ iff $p = q$
  - $\mathrm{JSD}(p, q) = \mathrm{JSD}(q, p)$
  - $\sqrt{\mathrm{JSD}(p, q)}$ satisfies triangle inequality
- Optimal generator for the JSD GAN

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

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

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

Alternating Optimization in GAN

$$\min_{\theta} \max_{\phi} V(G_\theta, D_\phi) = \mathbb{E}_{x \sim p_{\text{data}}} \log D_\phi(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\phi(G_\theta(z)))$$

- sample m training points $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ from $\mathcal{D}$
- sample m noise vectors $z^{(1)}, z^{(2)}, \ldots, z^{(m)}$ from p_z
- generator parameters θ update: stochastic gradient descent

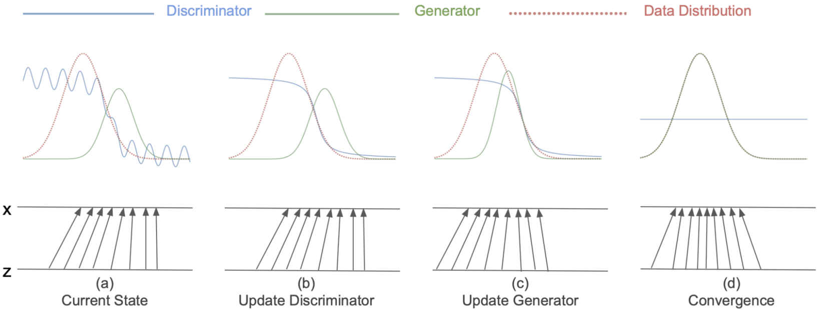
$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m} \nabla_\theta \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(z^{(i)})))$$

- discriminator parameters ϕ update: stochastic gradient ascent

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

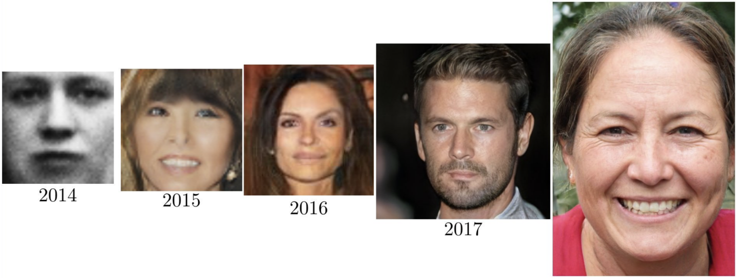
- Repeat for fixed number of epochs

A Toy Example



Adapted from Goodfellow, 2014

Frontiers in GAN Research

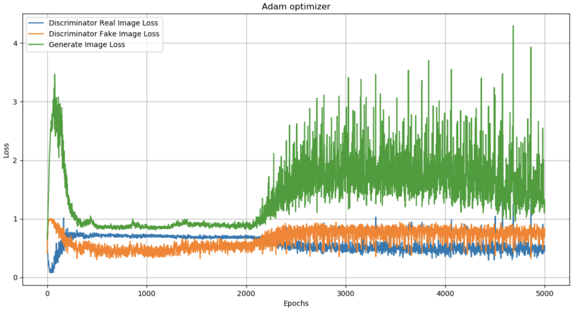


2014 2015 2016 2017 2018

- 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

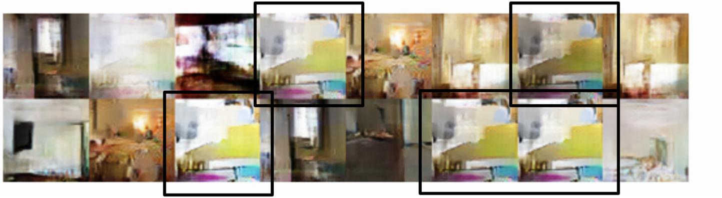
Image source: Ian Goodfellow. Samples from Goodfellow et al., 2014, Radford et al., 2015, Liu et al., 2016, Karras et al., 2017, Karras et al., 2018

- Theorem: 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



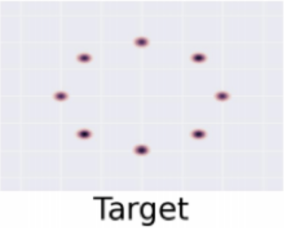
- No robust stopping criteria in practice (unlike MLE)

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

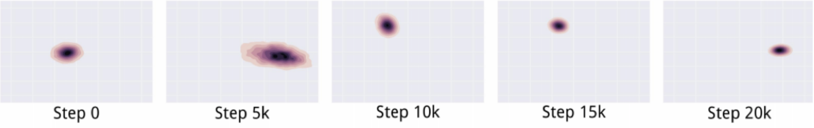


Arjovsky et al., 2017

Mode Collapse



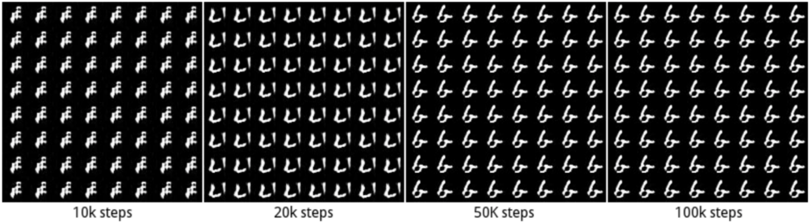
- True distribution is a mixture of Gaussians



Source: Metz et al., 2017

- The generator distribution keeps oscillating between different models

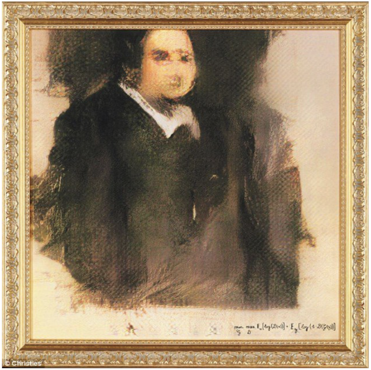
Mode Collapse



Source: Metz et al., 2017

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

GAN Generated Artworks



Source: Robbie Barrat, Obvious

GAN generated art auctioned at Christie's.
Expected Price: $7,000 – $10,000
True Price: $432,500

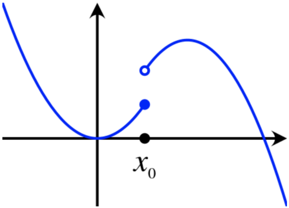
- The GAN Zoo:
  `https://github.com/hindupuravinash/the-gan-zoo`
- Examples
  - Rich class of likelihood-free objectives
  - Combination with latent representations
  - Application: Image-to-image translation, etc.

- Given two densities p and q, the $f-$ divergence is given by

$$D_f(p\|q) = \mathbb{E}_{x\sim q}\, f\left(\frac{p(x)}{q(x)}\right)$$

where f is any convex, lower-semicontinuous function with $f(1) = 0$

- Lower-semicontinuous: function value at any pint x_0 is close to $f(x_0)$ or greater than $f(x_0)$



- Example: KL divergence with $f(u) = u \log u$

f Divergence

Many more f-divergence!

Name	$D_f(P\|Q)$	Generator $f(u)$
Total variation	$\frac{1}{2}\int \lvert p(x)-q(x)\rvert\,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$
Reverse Kullback-Leibler	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\mathrm{d}x$	$(\sqrt{u}-1)^2$
Jeffrey	$\int (p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x$	$(u-1)\log u$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2}+u\log u$
Jensen-Shannon-weighted	$\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)}+(1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$	$\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$	$u\log u-(u+1)\log(u+1)$
α-divergence ($\alpha\notin\{0,1\}$)	$\frac{1}{\alpha(\alpha-1)}\int\left(p(x)\left[\left(\frac{q(x)}{p(x)}\right)^{\alpha}-1\right]-\alpha(q(x)-p(x))\right)\mathrm{d}x$	$\frac{1}{\alpha(\alpha-1)}\left(u^{\alpha}-1-\alpha(u-1)\right)$

Source: Nowozin et al., 2016

- To use f-divergences as a two-sample test objective for likelihood-free learning, we need to be able to estimate it only via samples
- Fenchel conjugate: For any function $f(\cdot)$, its convex conjugate is defined as

$$f^*(t) = \sup_{u \in \text{dom}_f} ut - f(u)$$

- Duallity: $f^{**} = f$. When $f(\cdot)$ is convex, lower semicontinuous, so is $f^*(\cdot)$

$$f(u) = \sup_{t \in \text{dom}_{f^*}} tu - f^*(t)$$

- We can obtain a lower bound to any f-divergence via its Fenchel conjugate

$$
\begin{aligned}
D_f(p\|q) &= \mathbb{E}_{x\sim q}\, f\left(\frac{p(x)}{q(x)}\right) \\
&= \mathbb{E}_{x\sim q} \sup_{t\in \mathrm{dom}_{f^*}} \left(t\frac{p(x)}{q(x)} - f^*(t)\right) \\
&\geq \mathbb{E}_{x\sim q}\, t(x)\frac{p(x)}{q(x)} - f^*(t(x)) \\
&= \int_{\mathcal{X}} t(x)p(x) - f^*(t(x))q(x)dx \\
&= \mathbb{E}_{x\sim p}\, t(x) - \mathbb{E}_{x\sim q}\, f^*(t(x))
\end{aligned}
$$

for any function $t : \mathcal{X} \mapsto \mathrm{dom}_{f^*}$

- Variational lower bound

$$D_f(p\|q) \geq \sup_{t \in \mathcal{T}} (\mathbb{E}_{x\sim p}\, t(x) - \mathbb{E}_{x\sim q}\, f^*(t(x)))$$

- Choose any f-divergence
- Let $p = p_{\text{data}}$ and $q = p_G$
- Parameterize t by ϕ and G by θ
- Consider the following f-GAN objective

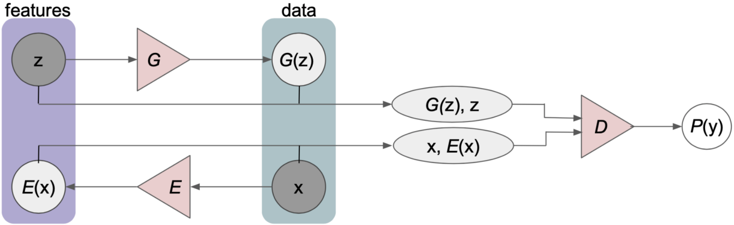
$$\min_\theta \max_\phi F(\theta, \phi) = \mathbb{E}_{x\sim p_{\text{data}}}\, t_\phi(x) - \mathbb{E}_{x\sim p_{G_\theta}} f^*(t_\phi(x))$$

- Generator G_θ tries to minimize the divergence estimate and discriminator t_ϕ tries to tighten the lower bound

- The generator of a GAN is typically a directed, latent variable model with latent variable z and observed variables x. How can we infer the latent feature representations in a GAN?
- Unlike a normalizing flow model, the mapping $G : z \mapsto x$ need not to be invertible
- Unlike a variational autoencoder, there is no inference network $q(\cdot)$ which can learn a variational posterior over latent variables
- Solution 1: For any point x, use the activations of the prefinal layer of a discriminator as a feature representation
- Intuition: similar to supervised deep neural networks, the discriminator would have learned useful representations for x while distinguishing real and fake x

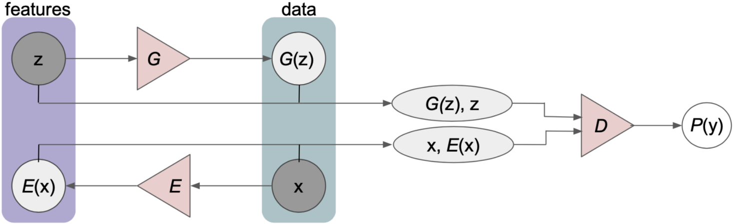
- If we want to directly learn the latent representation of x, we need a different learning algorithm
- A regular GAN optimizes a two-sample test objective that compares samples of x from the generator and the data distribution
- Solution 2: To infer latent representations, we will compare samples of x, z from joint distributions of observed and latent variables as per the model and the data distribution
- For any x generated via the model, we have access to z (sampled from a simple prior $p(z)$)
- For any x from the data distribution, the z is however unobserved (latent)

Bidirectional GAN



- In a BiGAN, we have an encoder network E in addtion to the generator network G
- The encoder network only observes $x \sim p_{\text{data}}(x)$ during training to learn a mapping $E : x \mapsto z$
- As before, the generator network only observes the samples from the prior $z \sim p(z)$ during training to learn a mapping $G : z \mapsto x$

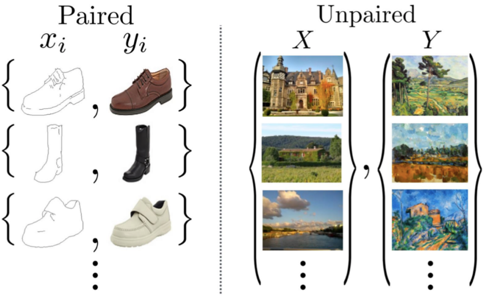
Bidirectional GAN



- The discriminator D observes samples from the generative model $z, G(z)$ and encoding distribution $E(x), x$
- The goal of the discriminator is the maximize the two-sample test objective between $z, G(z)$ and $E(x), x$
- After training is complete, new samples are generated via G and latent representations are inferred via E

Translating Across Domains

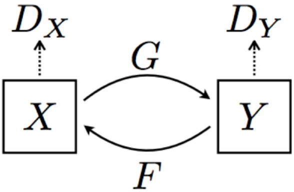
- Image-to-image translation: we are given image from two domains, $\mathcal{X}$ and $\mathcal{Y}$
- Paired vs. unpaired examples



Source: Zhu et al., 2016

- Paired examples can be expensive to obtain. Can we translate from $\mathcal{X} \Leftrightarrow \mathcal{Y}$ in an unsupervised manner?

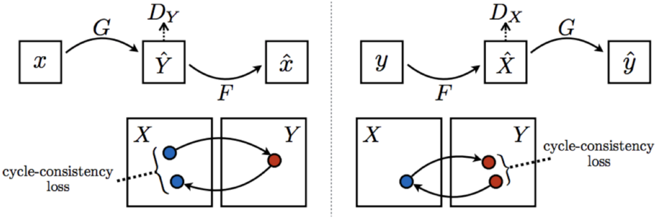
- To match the two distributions, we learn two parameterized conditional generative models $G : \mathcal{X} \mapsto \mathcal{Y}$ and $F : \mathcal{Y} \mapsto \mathcal{X}$
- G maps an element of $\mathcal{X}$ to an element of $\mathcal{Y}$. A discriminator $D_{\mathcal{Y}}$ compares the observed dataset Y and the generated samples $\hat{Y} = G(X)$
- Similarly, F maps an element of $\mathcal{Y}$ to an element of $\mathcal{X}$. A discriminator $D_{\mathcal{X}}$ compares the observed dataset X and the generated samples $\hat{X} = F(Y)$



Source: Zhu et al., 2016

CycleGAN

- Cycle consistency: If we can go from X to $\hat{Y}$ via G, then it should also be possible to go from $\hat{Y}$ back to X via F
  - $F(G(X)) \approx X$
  - Similarly, vice versa: $G(F(Y)) \approx Y$

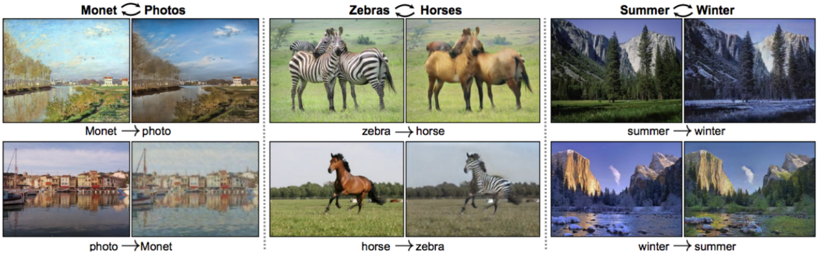


Source: Zhu et al., 2016

- Overall loss function

$$\mathcal{L}_{\text{GAN}}(G, D_{\mathcal{Y}}, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_{\mathcal{X}}, X, Y)$$
$$+\lambda(\mathbb{E}_X \|F(G(X)) - X\|_1 + \mathbb{E}_Y \|G(F(Y)) - Y\|_1)$$

CycleGAN in Practice



Source: Zhu et al., 2016

- Key observation: Samples and likelihoods are not correlated in practice
- Two-sample test objectives allow for learning generative mdoels only via samples (likelihood-free)
- Wide range of two-sample test objectives covering f-divergences (and more)
- Latent representations can be inferred via BiGAN (and other GANs with similar autoencoder structures)
- Cycle-consistent domain translations via CycleGAN and other variants

References

- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.
- Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In Proceedings of the Cognitive Science Society, volume 36, 2014.
- R. Shu, H. H. Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon. Amortized inference regularization. In Advances in Neural Information Processing Systems, pages 4393–4402, 2018.
- Naesseth, C. A., Linderman, S. W., Ranganath, R., and Blei, D. M. Variational sequential Monte Carlo. In International Conference on Artificial Intelligence and Statistics, 2018.

References

- C.J. Maddison, D. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Whye Teh. Filtering variational objectives. In Advances in Neural Information Processing Systems, 2017.
- Le, T. A., Igl, M., Rainforth, T., Jin, T., and Wood, F. (2018). Auto-Encoding Sequential Monte Carlo. International Conference on Learning Representations.
- L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models. International Conference on Learning Representations, 2016.
- Zhao, S., Song, J., and Ermon, S. Infovae: Information maximizing variational autoencoders. arXiv preprint arXiv:1706.02262, 2017.

- L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In ICLR, 2016
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.
- Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.
- Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.

- L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163, 2016.
- S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In Advances in neural information processing systems, pages 271–279, 2016.
- Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. arXiv preprint arXiv:1605.09782, 2016.
- Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In ICCV, 2017.