

VAE and GANs

Refs:

- [1] D. Kingma and M. Welling, Auto-encoding variational Bayes, ICLR 2014. (arXiv 1312.6114).
- [2] Matthew Bernstein, Blog posts on variational autoencoders variational inference, mbernste.github.io/posts/vae, March 2023.
- [3] Infinite norm (blog by Chris Ormrod), Deriving the ELBO, chrisorm.github.io/VI-ELBO.html.
- [4] Colin Raffel (blog), GANs and Divergence Minimization, December 2018, colinraffel.com/blog/gans-and-divergence-minimization.html.
- [5] Yufan Jiang (blog post), ELBO - What and Why, Jan 2021. yufanj.com/blog/2021/01/11/ELBO.html
- [6] Agustinus Kristiadi (blog post), KL Divergence: Forward or Reverse?, Dec. 2016, [agustinus.kristia.de/blog/...](https://agustinus.kristia.de/blog/...-forward-reverse-kl/)
... forward-reverse-kl/
- [7] I. Goodfellow et. al., Generative Adversarial Nets, NeurIPS 2014
- [8] S. Nowozin et. al., F-GAN: Training generative neural samplers using variational divergence minimization, arXiv 1606.00709.

Setting: Given samples $\{x_1, x_2, \dots, x_n\}$
from a ground-truth distribution $p_*(x)$, $x \in \mathbb{R}^d$,
(In this note, I use $p_*(\cdot)$ for ground-truth instead of
a super-script $*$, as we use $(\cdot)^*$ for the Legendre-Fenchel dual).

① Learn a "good" approximation to $p_*(x)$:

Plausible approach: (i) Parameterize a class of distribution
functions $\mathcal{Q} = \{q_\theta(x), \theta \in \Theta\}$, where
 Θ is the set of model weights, and $q_\theta(\cdot)$ are the
associated prob. dist. functions.

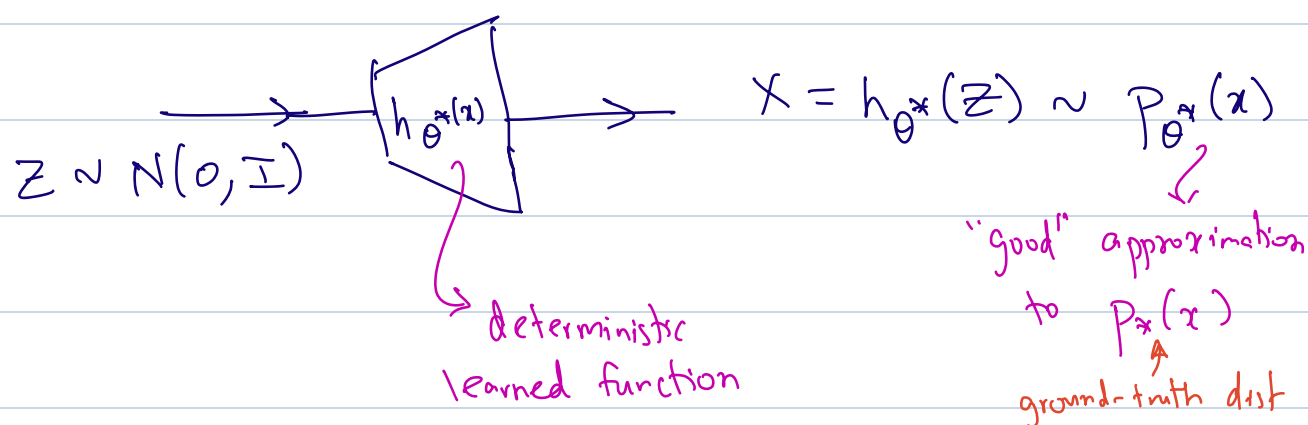
• Optimize ML to learn: $\ln q_*(x) \in \max_{\theta \in \Theta} \ln q_\theta(x)$

• use (yet to discuss) method to learn $\nabla \ln q_*(x)$
(also known as **score**)

(ii) Use this learned dist., along with a sampling
algorithm (e.g., M-H, Langevin, other SDEs) to
generate **new samples** $x_i \sim q_*(x)$.

We will study this approach in DDPM, flow models.

⑥ Directly learn a sampler, without explicitly learning the distribution. Thus, the learned model will take as input, a sample from a known distribution (e.g., a sample of a Gaussian) and output a sample from the desired distribution $p_*(x)$.



In the language of optimal transport, this is also referred to as the push-forward distribution, and the associated transport map is $T_{\#}^{\theta^*}(\mu_0) = p_{\theta^*}$

Annotations:

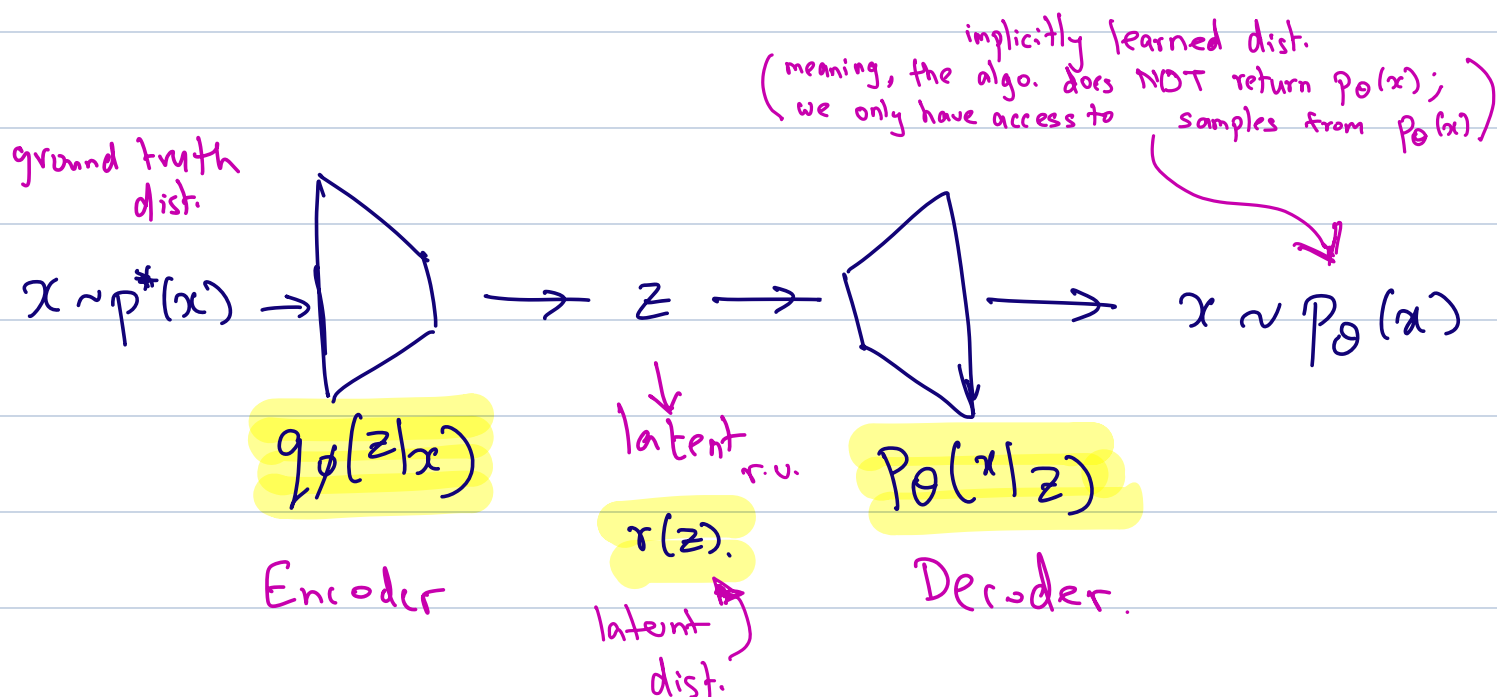
- maps measures to measures
- Gaussian measure = $N(0, I)$

We can also be somewhere in between, where we learn some conditionals, but do not marginalize to explicitly get $p_{\theta^*}(x)$; as we will see below in the VAE.

I Variational Inference and VAE

Goal: Given samples $\{x_1, x_2, \dots, x_n\} \sim p^*(x)$,
learn a model that generates new samples

The Variational Auto Encoder pipeline:



Design Choices: $(q_\phi(\cdot|\cdot), \pi(\cdot), p_\theta(\cdot|\cdot))$

Jointly train two models - (encoder, decoder) - parameterized by (ϕ, θ) ; the model architectures are design choices.

In addition to (ϕ, θ) that parameterize the encoder and decoder conditional distributions, the choice of the latent distribution, $r(z)$, is a design choice.

$x \in \mathbb{R}^d$, $z \in \mathbb{R}^m$ (typically $m < d$ resulting in a compressed latent space; see remark below.)

$r(\cdot)$ is chosen to be an easily sampleable distribution; typically $r(z) = N(0, I)$.

Remark: Though we are focussed on generation, the design of the VAE trades-off between two competing objectives:

① Generation of new samples: We sample $z \sim r(\cdot)$.

Then, conditioned on $z = z$, we further sample

$$\tilde{x} \sim P_{\theta^*}(\cdot | z).$$

Here, θ^* corresponds to the parameters of the

decoder resulting from training using the VAE loss (ELBO, see below).

As an example, suppose that the chosen decoder architecture is $p_{\theta}(x|z) = N(g_{\theta}(z), \sigma_{\text{dec}}^2 I)$ (i.e., a Gaussian model, with σ_{dec}^2 a hyper-parameter). Then, the resulting $\hat{x}$ would be generated by first computing the conditional mean $g_{\theta}(z)$, and then resampling from a Gaussian with this mean and covariance of $\sigma_{\text{dec}}^2 I$.

Another choice could be $p_{\theta}(x|z) = \delta_{h_{\theta}(z)}$,

i.e., $\hat{x} = h_{\theta}(z)$, $z \sim q(z)$. This decoder does not require a resampling step (e.g. Stable Diffusion).

The choice of the decoder architecture will limit the types of resulting distributions, $p_{\theta^*}(x)$, that can be modeled using this pipeline. In general, one will not expect, even in the infinite

training samples regime, to have $KL(p_{\theta^*}(x) || p_{\theta^*}(x)) = 0$.

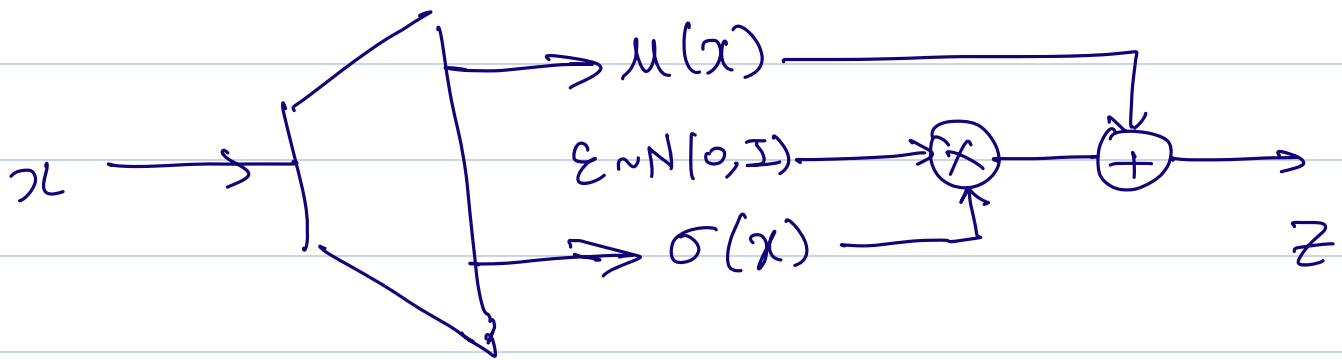
⑥ Compressed latent space: $x \in \mathbb{R}^d$ is mapped to $z \in \mathbb{R}^m$, typically a lower dimensional space.

This compression is important for applications, but will potentially result in some approximation error in terms of fidelity of the learned dist. w.r.t. the ground-truth distribution.

For example, in diffusion models, with Stable Diffusion, (the original $\sim 1B$ parameter model), x corresponds to $(512 \times 512 \times 3) = 786,432$ dimensions (512x512 pixels, in RGB). The latent space dimension is $(64 \times 64 \times 8)$. The diffusion model operates in this latent space, where gradients, etc. can be computed tractably.

This trade-off between fidelity and compression is an important design choice.

Encoder: The encoder $q_{\phi}(z|x)$ is explicitly parameterized as follows:



$$z = \mu(x) + \text{diag}(\sigma(x))\epsilon$$

where $\epsilon \sim N(0, I)$.

In words, $q_{\phi}(z|x) = N\left(\mu_{\phi}(x), \text{diag}(\sigma_{\phi}(x))\right)$

↙
diagonal matrix

The reparameterization of the conditional distribution of the latent variable z through an independent Gaussian $\epsilon \sim N(0, I)$, equivalently, modeling the parameters of the cond. dist. $q_{\phi}(\cdot|\cdot)$ through $\phi \equiv (\mu_{\phi}, \sigma_{\phi})$ is important to enable end-to-end training via gradient descent, as we will see later.

Deriving the VAE Training Loss

Recall that there are two sets of learnable parameters: (i) ϕ parameterizes the encoder distribution $q_{\phi}(z|x)$, and (ii) θ parameterizes the decoder distribution $p_{\theta}(x|z)$.

The VAE Training Loss (ELBO):
 $\hookrightarrow$ Evidence Lower Bound

Maximum likelihood:
objective for learning θ

$$\theta^* \in \arg \max_{\theta} \ln p_{\theta}(x)$$

Instead of directly learning a dist over x , augment to have a latent space z , and learn the joint over (x, z) . Concretely,

$$\ln p_{\theta}(x) = \ln \int_{\mathbf{z}} p_{\theta}(x, z) dz$$

$\nearrow$ log-likelihood fn

$$= \ln \int_{\mathbf{z}} p_{\theta}(x, z) \cdot \frac{q_{\phi}(z|x)}{q_{\phi}(z|x)} dz = \ln E_{z \sim q_{\phi}(\cdot|x)} \left[\frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \right]$$

Jensen's Ineq.

$$\geq \underbrace{E_{z \sim q_\phi(\cdot|x)} \left[\ln \left(\frac{p_\theta(x,z)}{q_\phi(z|x)} \right) \right]}_{\text{ELBO}(x)} \rightarrow \text{ELBO}$$

$$= E_{z \sim q_\phi(\cdot|x)} \left[\ln p_\theta(x,z) \right] + \underbrace{\int \ln \left(\frac{1}{q_\phi(z|x)} \right) q_\phi(z|x) dz}_{H(q_\phi(z|x))}$$

Thus,

$$\text{ELBO}(x) = \int_z \ln \left(\frac{p_\theta(x,z)}{q_\phi(z|x)} \right) \cdot q_\phi(z|x) dz$$

$\text{ELBO}(x; \theta, \phi)$

learnable parameters

$$= \int q_\phi(z|x) \ln \left(\frac{p_\theta(x|z) \cdot \underbrace{\tau(z)}_{N(0,I)}}{q_\phi(z|x)} \right) dz$$

$$= \underbrace{E_{z \sim q_\phi(z|x)} \left[\ln p_\theta(x|z) \right]}_{\text{(i) Fidelity of generation}} - \underbrace{KL(q_\phi(z|x) \parallel \tau(z))}_{\text{(ii) Compression of latent}}$$

(i) Fidelity of generation

$\mathcal{L}_1(x)$

(ii) Compression of latent

$\mathcal{L}_2(x)$

$$(i) \quad \alpha_1(x) = E_{z \sim q_\phi(z|x)} \left[\ln p_\theta(x|z) \right]$$

This focuses on **fidelity** of generation. To see this, note that $p_\theta(x|\cdot)$ is a density function that is maximized when $z = x$, and $p_\theta(\cdot|x)$ is a dirac delta function (i.e., z matches x , and the latent is simply a copy of the data).

Furthermore recall that this loss function is averaged over $\{x_1, \dots, x_n\} \sim p_*(x)$, i.e., $E_{x \sim p_*(x)} [\alpha_1(x)]$

in the infinite sample limit. Thus, optimizing this alone over (θ, ϕ) , and ignoring $\alpha_2(x)$, leads to the density of z being the same as the ground-truth $p_*(x)$.

$$(ii) \quad \alpha_2(x) = KL(q_\phi(z|x) \parallel r(z))$$

This term optimizes for **compression** of the latent. Recall $r(z) \sim N(0, \Sigma)$. Thus, this term tries

to make every $q_\phi(\cdot|x)$ the same, independent of x . In other words, all the conditional dist. are pushed close to each other, thus compressing the latent space.

Example: A specific ELBO Calculation

Suppose:

$$p_\theta(x|z) \triangleq N(g_\theta(z), \sigma_{\text{dec}}^2 \cdot I)$$

$\downarrow$ learnable function. $\hookrightarrow$ hyper-parameter

$$\text{i.e., } \ln p_\theta(x|z) = C - \frac{\|x - g_\theta(z)\|_2^2}{2\sigma_{\text{dec}}^2}$$

In this case:

$$\text{ELBO}(x; \theta, \phi) = \left(\begin{aligned} &C - E_{z \sim q_\phi(\cdot|x)} \left[\frac{\|x - g_\theta(z)\|_2^2}{2\sigma_{\text{dec}}^2} \right] \\ &- \text{KL}(q_\phi(z|x) \parallel \underbrace{\tau(z)}_{\hookrightarrow N(0, I)}) \end{aligned} \right)$$

Training the VAE and reparameterization of the encoder ϕ through (μ_ϕ, σ_ϕ)

Recall the VAE objectives from samples:

VAE Training Objective :

$$\max_{\theta, \phi} \sum_{i=1}^n \text{ELBO}(x_i)$$

n iid samples given from $x_i \sim p_x(x)$

Training with ELBO loss requires gradient w.r.t. (ϕ, θ) . The gradient w.r.t ϕ seems tricky because ϕ parameterizes the dist. with respect to which we are taking expectation.

The following reparameterization (discussed in the beginning of this note) is useful:

$$Z \sim q_\phi(z|x) \stackrel{d}{=} \mu_\phi(x) + \text{diag}(\sigma_\phi(x)) \cdot \epsilon$$

where $\epsilon \sim N(0, I)$.

We can now change variable to an expectation over $\mathcal{Z}$. With this change, and with

$$\mu_{\phi}(x) = h_{\phi}^{(1)}(x), \quad z \sim \mathcal{N}(0, I)$$
$$\ln \sigma_{\phi}(x) = h_{\phi}^{(2)}(x) \quad \rightarrow \in \mathbb{R}^d$$

we have: $KL(q_{\phi}(z|x) || p(z))$

$$= -\frac{1}{2} \sum_{j=1}^d \left(1 + h_{\phi}^{(2)}(x)_j - (h_{\phi}^{(1)}(x)_j)^2 - e^{h_{\phi}^{(2)}(x)_j} \right)$$

(see [2] for derivation).

Generative Adversarial Networks (GANs)

We next proceed to study GANs. At a high level, the VAE used the ELBO bound using Jensen's inequality. This lead to an approximation error in the objective (w.r.t. to the original intent of maximum likelihood), that

is difficult to control/characterize. The GAN, through a dual formulation, explicitly allows one to control the approximation.

Before studying GANs, we study properties of the KL Divergence.

(II) Forward vs. Reverse KL.

First, a basic connection between ML and KL Divergence.

Suppose $p^*(x)$ is the ground-truth dist., and $q_\theta(x)$ is a parametric family of dist, where θ is the learnable parameter. Then,

$$KL(p^*(x) \parallel q_\theta(x)) = \int p^*(x) \ln \frac{p^*(x)}{q_\theta(x)} dx$$

$$= \int p^*(x) \ln p^*(x) dx - \int p^*(x) \ln q_\theta(x) dx$$

$$= E_{x \sim p^*} [\ln p^*(x)] - E_{x \sim p^*} [\ln q_\theta(x)]$$

↗ does not have θ dependence.

$$\therefore \min_{\theta} KL(p^*(x) \| q_\theta(x)) = \max_{\theta} E_{x \sim p^*} [\ln q_\theta(x)]$$

$$\approx \max_{\theta} \frac{1}{n} \sum_{i=1}^n \ln q_\theta(x_i)$$

given n iid samples
 $\{x_1, \dots, x_n\} \sim p^*(x)$,
 approximate the expectation
 by the empirical average

maximum likelihood
 estimation

The above KL divergence is called the **forward KL**, as the ground-truth dist. is the averaging dist.

The issue with the forward KL is that in an under-parameterized model class, this can lead to poor model fit.



Suppose $p^*(x)$ is a mixture of two Gaussians as above, but $\mathcal{Q} = \{q_\theta : q_\theta = \mathcal{N}(\theta, 1)\}$

(i.e., one parameter class, with θ being the mean)

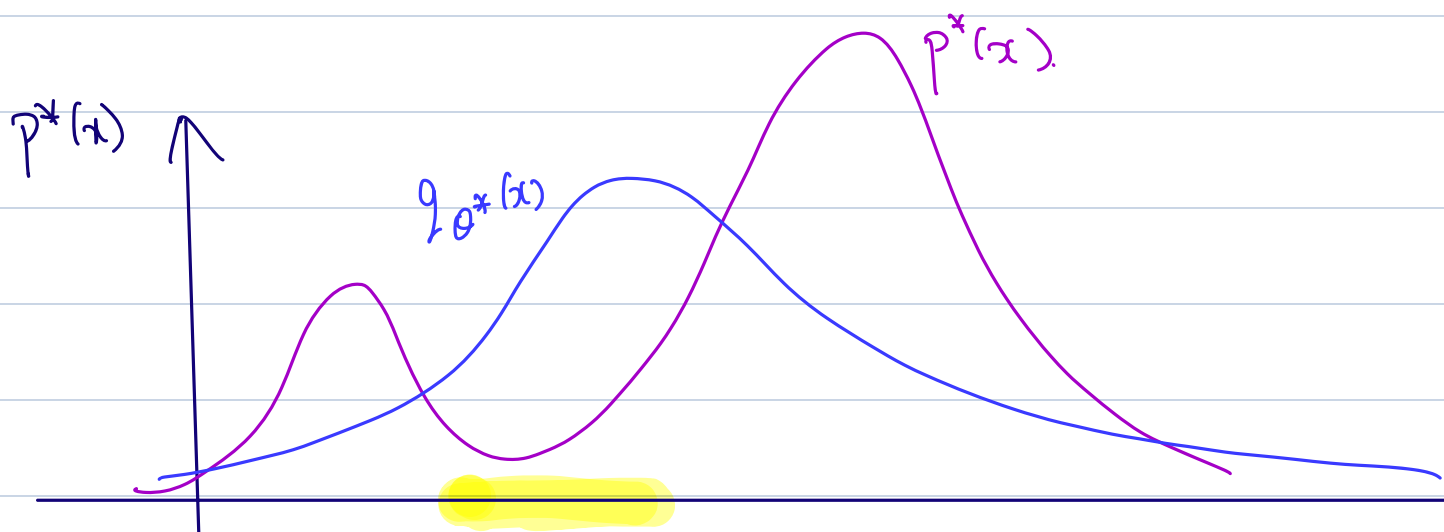
$$\text{Now, } KL(p^*(x) \| q_\theta(x)) = \int p^*(x) \ln \frac{p^*(x)}{q_\theta(x)} dx.$$

Observe that:

(i) If $q_\theta(x)$ has low prob. over regions where $p^*(x)$ is high, then the loss is large
($\because \frac{p^*(x)}{q_\theta(x)}$ is large)

(ii) If $q_\theta(x)$ has high prob. over regions where $p^*(x)$ is small, it does not matter.

$$\left(\because p^*(x) \approx 0 \Rightarrow p^*(x) \cdot \ln \left(\frac{p^*(x)}{q_\theta(x)} \right) \approx 0 \right)$$



↳ low prob. region of $p^*(x)$, but high prob with $q_{\theta^*}(x)$.

Then, the loss $\min_{\theta} KL(p^*(x) \| p_{\theta}(x))$ will

choose θ^* s.t. there is sufficient coverage over both modes, even though there is a significant mass on low-probability regions of $p^*(x)$. This can result in a large number of atypical samples, when $q_{\theta^*}(x)$ is used as a generative model.

Reverse KL : $KL(q_{\theta}(x) \| p^*(x))$

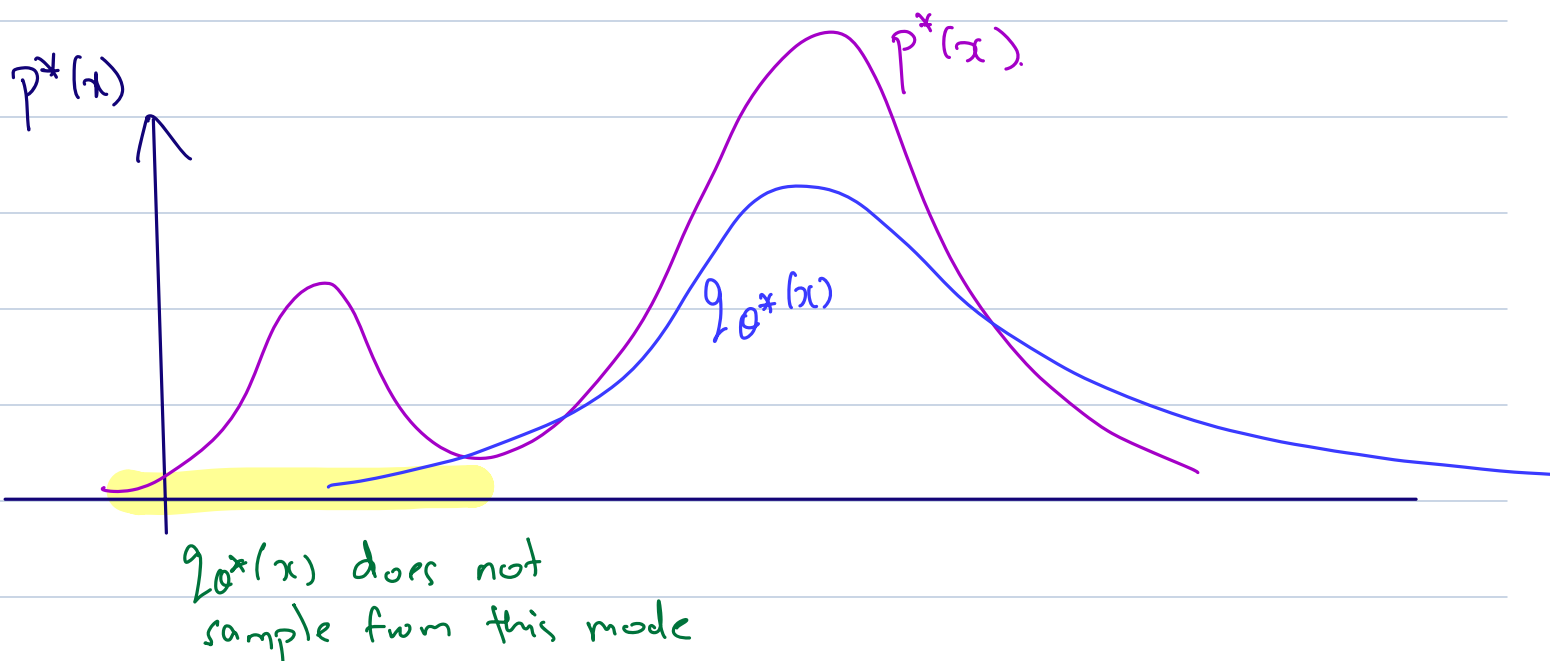
$$= \int q_{\theta}(x) \cdot \ln \frac{q_{\theta}(x)}{p^*(x)} dx.$$

Suppose that we use the reverse KL as the loss for training the generative model. Observe that:

(i) If $q_\theta(x)$ has high prob. over regions of $p^*(x)$ with low prob, then the loss is high
(because $\frac{q_\theta(x)}{p^*(x)}$ is large)

(ii) If $q_\theta(x)$ has low prob. over regions of $p^*(x)$ with high prob, it does not matter

$$\left(\because q_\theta(x) \approx 0 \Rightarrow q_\theta(x) \ln \left(\frac{q_\theta(x)}{p^*(x)} \right) \approx 0 \right)$$



In this case, $q_{\theta^*}(x)$ will result in typical samples, BUT will rarely sample from the smaller mode. (mode collapse)

Thus, if one wants to prioritize typical samples (e.g., realistic images) over diversity of images (multiple modes), the reverse KL is the loss function of choice.

However, the reverse KL has a problem: We do not have function access to $p^*(x)$; we only have samples $x_1, x_2, \dots, x_n \sim p^*(x)$.

Thus it is unclear how to compute:

$$KL(q_{\theta}(x) \| p^*(x)) = \int q_{\theta}(x) \ln \left(\frac{q_{\theta}(x)}{p^*(x)} \right) dx$$

we do not know this, and UNLIKE the forward KL, we cannot easily approximate the reverse KL loss by empirical averages from samples of $p^*(x)$

As we will see next, the GAN is an approach to resolve this issue of implementing a reverse KL loss.

III

Generative Adversarial Network (GAN)

(notes follow discussion in [7])

Beyond forward and reverse KL, we could use a variety of divergence measures including the Jensen-Shannon divergence (a symmetrized KL), the GAN divergence (developed in Goodfellow et. al.), etc.

$P_*(x)$: ground-truth dist.

$q_\theta(x)$: θ -parameterized distribution ; $\theta \in \Theta$.

Goal: Given $\{x_1, x_2, \dots, x_n\} \sim P_*(x)$ iid samples
learn θ^* to minimize

$$\theta^* \in \underset{\theta}{\operatorname{argmin}} D_f(P_* \| q_\theta) \longrightarrow \textcircled{A}$$

where $D_f(P_* \parallel q_0) = \int q_0(x) f\left(\frac{P_*(x)}{q_0(x)}\right) dx$

$\hookrightarrow$ convex

Examples:

a. $f(u) = u \ln u \rightarrow D_f(P_* \parallel q_0) = \int P_*(x) \ln \frac{P_*(x)}{q_0(x)} dx$
(KL div.)

b. $f(u) = -\ln u \rightarrow D_f(\cdot \parallel \cdot) = \int q_0(x) \ln \frac{q_0(x)}{P_*(x)} dx$
(reverse KL div.)

c. $f(u) = (\sqrt{u} - 1)^2 \rightarrow D_f(\cdot \parallel \cdot) = \int (\sqrt{P_*(x)} - \sqrt{q_0(x)})^2 dx$
(squared Hellinger)

d. $f(u) = -(u+1) \ln\left(\frac{1+u}{2}\right) + u \ln u$

(Jensen-Shannon) $\rightarrow D_f(\cdot \parallel \cdot) = \frac{1}{2} \text{KL}\left(P_*(\cdot) \parallel \frac{P_*(\cdot) + q_0(\cdot)}{2}\right) + \frac{1}{2} \text{KL}\left(q_0(\cdot) \parallel \frac{P_*(\cdot) + q_0(\cdot)}{2}\right)$
symmetrized KL.

(There are several others, see [7]).

Legendre - Fenchel Duality

$f(x) \in \mathcal{F}$, a convex lower semi continuous function

$$f: \mathbb{R}_+ \mapsto \mathbb{R}.$$

(this can be generalized to other domains. These notes only present the simplest version.)

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

↳ convex conjugate aka Legendre-Fenchel dual of f .

Facts:

- f^* is also convex and lower-semicontinuous
- $f^{**} = f$

$$\therefore f(x) = \sup_{t \in \text{Domain}(f^*)} (tx - f^*(t)).$$

This duality applies to f -divergences. Specifically,

$$D_f(p_* \| q_0) = \int x q_0(x) f\left(\frac{p_*(x)}{q_0(x)}\right) dx$$

$$= \int_{\mathcal{X}} q_0(x) \cdot \sup_{t \in \text{Domain}(f^*)} \left(t \cdot \frac{p_*(x)}{q_0(x)} - f^*(t) \right) dx$$

$\downarrow$
 depends on x

Let $\mathcal{T} = \left\{ T : \mathcal{X} \mapsto \mathbb{R} \right\}$ collection of functions, sufficiently rich

$\downarrow$
 $\subseteq \mathbb{R}$

(See N. Nguyen, M. Wainwright, Estimating divergence functionals and the likelihood ratio by convex risk minimization, IEEE Trans IT, 2010.)

$$\geq \sup_{T \in \mathcal{T}} \left(\int_{\mathcal{X}} p_*(x) T(x) dx - \int_{\mathcal{X}} q_0(x) f^*(T(x)) dx \right)$$

$$= \sup_{T \in \mathcal{T}} \left(E_{x \sim p_*} [T(x)] - E_{y \sim q_0} [T(y)] \right)$$

Furthermore, under reasonable conditions, the inequality above is tight, when $T_*(x) = f' \left(\frac{p_*(x)}{q_0(x)} \right)$.

Summary: $\min_{\theta \in \Theta} D_f (P_{\theta}(x) \parallel q_{\theta}(x))$

$$= \min_{\theta \in \Theta} \max_{T \in \mathcal{T}} \left(E_{x \sim P_{\theta}} [T(x)] - E_{x \sim q_{\theta}} [T(x)] \right).$$

$\hookrightarrow \textcircled{B}$

Examples: (of the maximizing T , for various f).

1. (KL) $f(u) = u \ln u$ $T^*(x) = 1 + \ln \frac{P_{\theta}(x)}{q_{\theta}(x)}$

2. (reverse KL) $f(u) = -\ln u$ $T^*(x) = -\frac{q_{\theta}(x)}{P_{\theta}(x)}$

3. (Squared Hellinger) $f(u) = (\sqrt{u} - 1)^2$ $T^*(x) = \left(1 - \sqrt{\frac{q_{\theta}(x)}{P_{\theta}(x)}} \right)$

4. (Jensen-Shannon) $f(u) = -(u+1) \ln \left(\frac{1+u}{2} \right) + u \ln u$

$$T^*(x) = \ln \frac{2 P_{\theta}(x)}{P_{\theta}(x) + q_{\theta}(x)}$$

(see [7] for more)

$$P_{\theta}(x) + q_{\theta}(x)$$

Recall our primary difficulty with reverse KL
— we **do not** have function access to $p_*(x)$

Hence, we use the dual form to address (A) (the primal optimization problem to learn the generative model $q_\theta(\cdot)$). Motivated by (B) (dual equivalence), we parameterize $T(x)$ by ϕ :

$$(\theta^*, \phi^*) = \min_{\theta} \max_{\phi} \left(E_{x \sim p_*} [T_{\phi}(x)] - E_{\gamma \sim q_{\theta}} [T_{\phi}(\gamma)] \right)$$

Note that the above min-max optimization does not require function access to p_* , as the expectation can be estimated from samples.

There are several approaches to solve the minmax optimization. We refer to [7, 8] and many other tutorials on algorithms (e.g., alternating gradient updates).