Large Scale ML: Generative models

Gabriel V Cardoso
gabriel.victorino_cardoso@minesparis.psl.eu

Equipe Géostatistiques, Ecole des Mines, Univ. PSL, France

What this lesson is about

Figure: Images generated by several different prompts using the FLUX model.[1]

[1] `https://huggingface.co/black-forest-labs/FLUX.1-schnell`

The probabilistic point of view

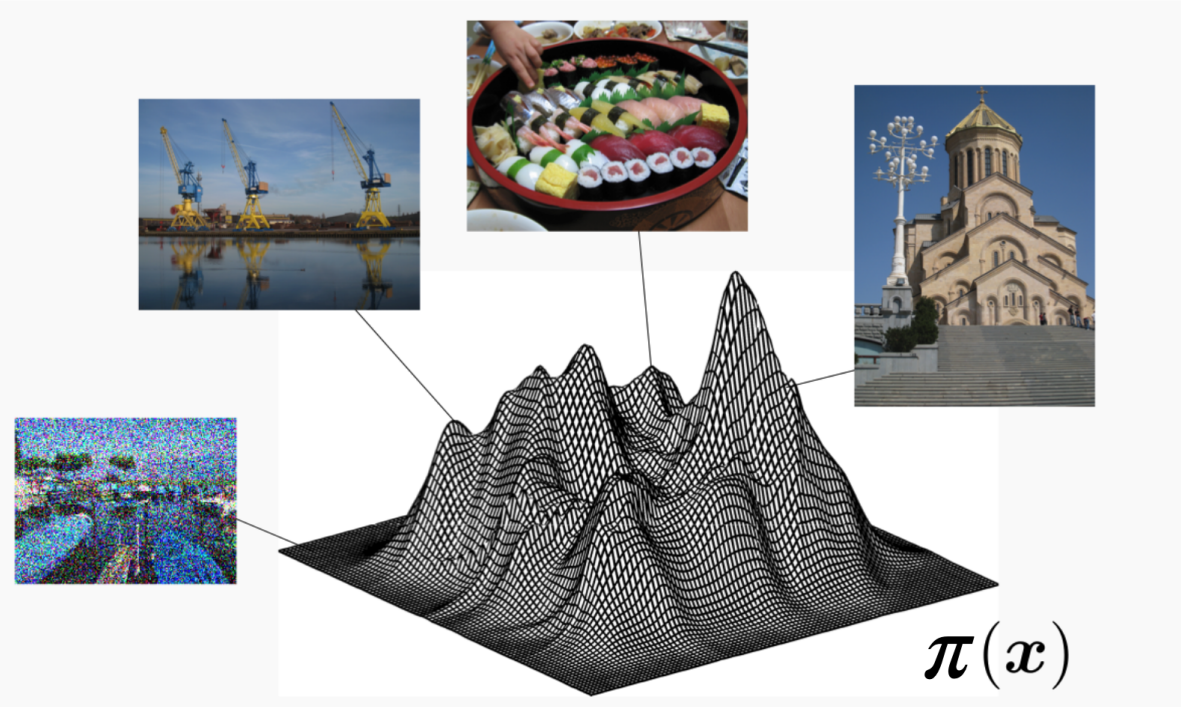


Figure: Landscape of the underlying data distribution [2].

[2] Source: https://www.charles-deledalle.fr/pages/files/ucsd_ece285_mlip/

A bit of context

- π lives in a high-dimensional space $d_x > 10^5$.
- We have access to big datasets $\mathcal{D} = \{X_1, \cdots, X_n\}$, $n > 10^5$, of samples of π.
- We have no parametric model for our data. [3]

Problem: For general π, density estimation in high-dimensions requires an exponential amount of data.

[3] or saying it differently, we can't code a program that generates real images from a set of fixed parameters.

A bit of context

- π lives in a high-dimensional space $d_x > 10^5$.
- We have access to big datasets $\mathcal{D} = \{X_1, \cdots, X_n\}$, $n > 10^5$, of samples of π.
- We have no parametric model for our data. [3]

Problem: For general π, density estimation in high-dimensions requires an exponential amount of data.

[3] or saying it differently, we can't code a program that generates real images from a set of fixed parameters.

The original miracle!

Figure: $X_0 \sim \pi$.

Figure: $X_\sigma = X_0 + \sigma Z$. [4]1

- D_{θ_0} untrained CNN,
- $L_\sigma(\theta) = \|D_\theta(X_\sigma) - X_\sigma\|^2$, $L_0(\theta) = \|D_\theta(X_\sigma) - X_0\|^2$,
- $\theta_t = \theta_{t-1} - \nabla L_\sigma(\theta_{t-1})$

What happens to $\{L_\sigma(\theta_t)\}_{t=1}^N$ and $\{L_0(\theta_t)\}_{t=1}^N$?

[1] $Z \sim \mathcal{N}(0, \mathrm{I})$, $X_0 \perp Z$.

The original miracle!

Figure: $X_0 \sim \pi$.

Figure: $X_\sigma = X_0 + \sigma Z$. [4]1

- D_{θ_0} untrained CNN,
- $L_\sigma(\theta) = \|D_\theta(X_\sigma) - X_\sigma\|^2$, $L_0(\theta) = \|D_\theta(X_\sigma) - X_0\|^2$,
- $\theta_t = \theta_{t-1} - \nabla L_\sigma(\theta_{t-1})$

What happens to $\{L_\sigma(\theta_t)\}_{t=1}^N$ and $\{L_0(\theta_t)\}_{t=1}^N$?

[1] $Z \sim \mathcal{N}(0, \mathrm{I})$, $X_0 \perp Z$.

The original miracle!

Figure: $X_0 \sim \pi$.

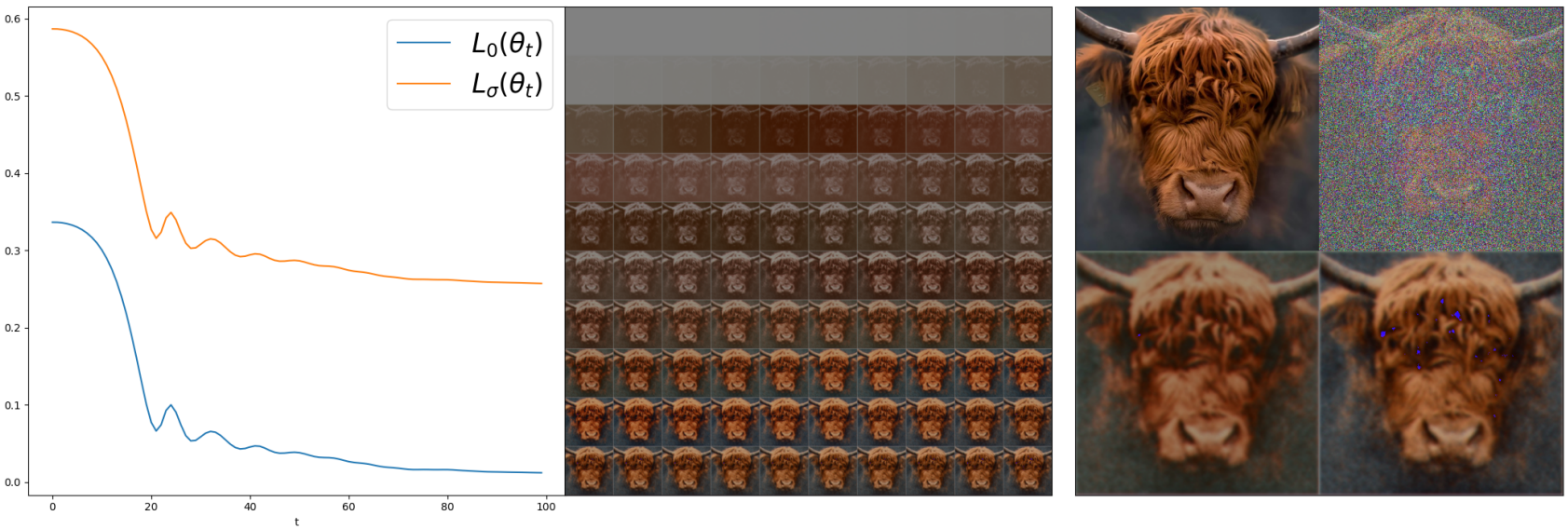
Figure: $X_\sigma = X_0 + \sigma Z$. [4]1

- D_{θ_0} untrained CNN,
- $L_\sigma(\theta) = \|D_\theta(X_\sigma) - X_\sigma\|^2$, $L_0(\theta) = \|D_\theta(X_\sigma) - X_0\|^2$,
- $\theta_t = \theta_{t-1} - \nabla L_\sigma(\theta_{t-1})$

What happens to $\{L_\sigma(\theta_t)\}_{t=1}^N$ and $\{L_0(\theta_t)\}_{t=1}^N$?

[1] $Z \sim \mathcal{N}(0, \mathrm{I})$, $X_0 \perp Z$.

The original miracle [5].

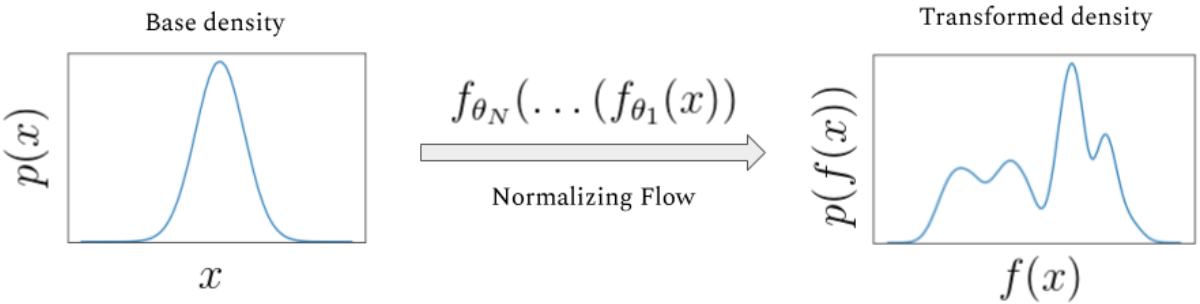


[5] This experiment is inspired by Ulyanov, Vedaldi, and Lempitsky, 2018

Outline

General strategy

1. Define a family of distributions $\mathcal{F} = \{p_\theta \in \mathcal{P}(\mathbb{R}^{d_x}) | \theta \in \Theta\}$. Typically this is done by
   1. Defining a neural network architecture f_θ parameterized by θ
   2. Define a reference distribution λ[6]
   3. $p_\theta := \text{Law}(f_\theta(Z))$ where $Z \sim \lambda$.



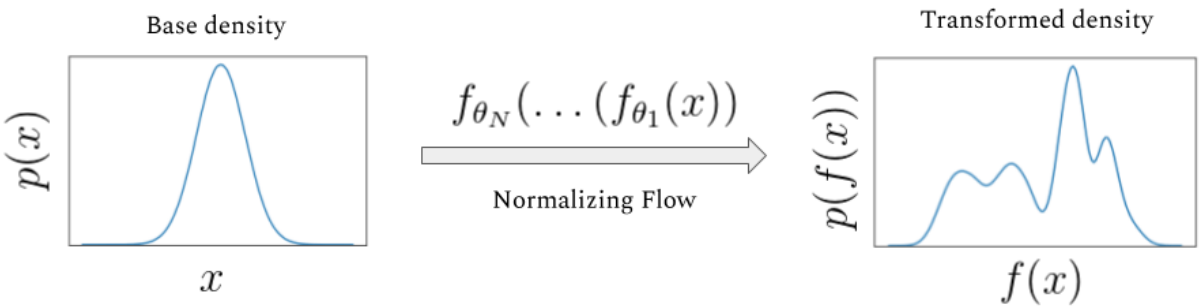
2. Choose a divergence[7] over distributions $D : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_*$
3. Solve

$$\theta^* \in \arg\min_{\theta \in \Theta} D(\pi, p_\theta)$$

[6] Typically, a Gaussian or Uniform.
[7] Only property we need is $D(p, q) = 0 \iff p = q$.

General strategy

1. Define a family of distributions $\mathcal{F} = \{p_\theta \in \mathcal{P}(\mathbb{R}^{d_x}) | \theta \in \Theta\}$. Typically this is done by
   1. Defining a neural network architecture f_θ parameterized by θ
   2. Define a reference distribution λ[6]
   3. $p_\theta := \text{Law}(f_\theta(Z))$ where $Z \sim \lambda$.



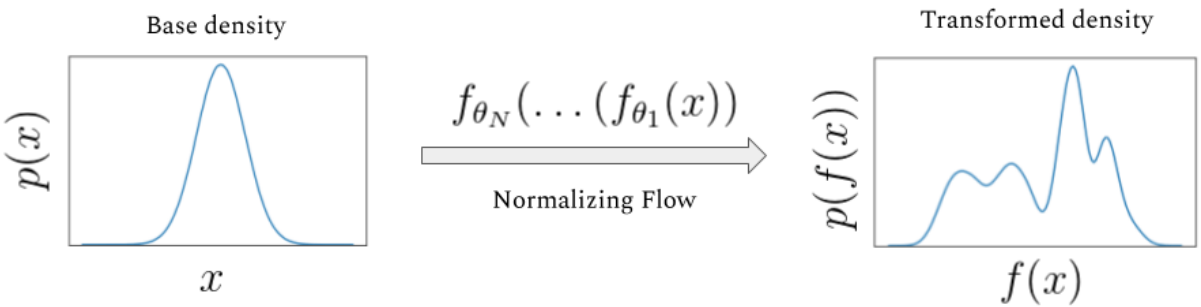
2. Choose a divergence[7] over distributions $D : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_*$
3. Solve

$$\theta^* \in \arg\min_{\theta \in \Theta} D(\pi, p_\theta)$$

[6] Typically, a Gaussian or Uniform.
[7] Only property we need is $D(p, q) = 0 \iff p = q$.

General strategy

1. Define a family of distributions $\mathcal{F} = \{p_\theta \in \mathcal{P}(\mathbb{R}^{d_x}) | \theta \in \Theta\}$. Typically this is done by
   1. Defining a neural network architecture f_θ parameterized by θ
   2. Define a reference distribution λ[6]
   3. $p_\theta := \text{Law}(f_\theta(Z))$ where $Z \sim \lambda$.



2. Choose a divergence[7] over distributions $D : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_*$
3. Solve

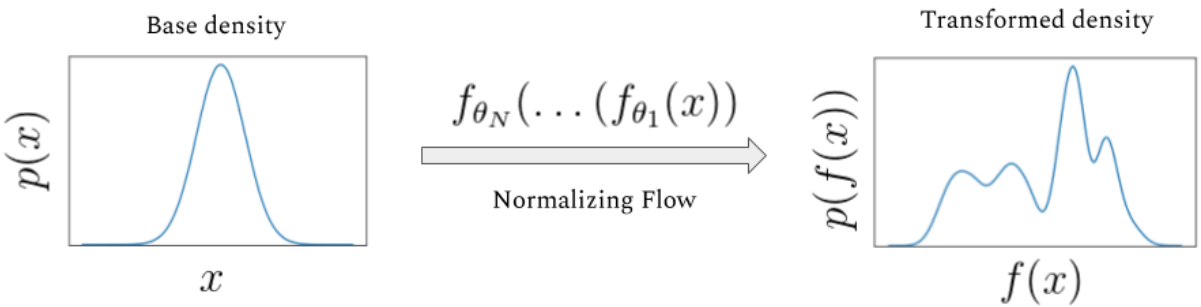
$$\theta^* \in \arg\min_{\theta \in \Theta} D(\pi, p_\theta)$$

[6] Typically, a Gaussian or Uniform.

[7] Only property we need is $D(p, q) = 0 \iff p = q$.

General strategy

1. Define a family of distributions $\mathcal{F} = \{p_\theta \in \mathcal{P}(\mathbb{R}^{d_x}) | \theta \in \Theta\}$. Typically this is done by
   1. Defining a neural network architecture f_θ parameterized by θ
   2. Define a reference distribution λ[6]
   3. $p_\theta := \text{Law}(f_\theta(Z))$ where $Z \sim \lambda$.



2. Choose a divergence[7] over distributions $D : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_*$
3. Solve

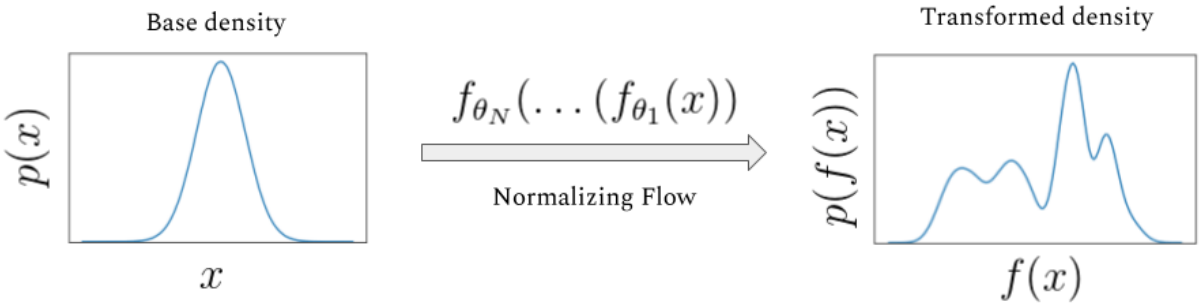
$$\theta^* \in \arg\min_{\theta \in \Theta} D(\pi, p_\theta)$$

[6] Typically, a Gaussian or Uniform.

[7] Only property we need is $D(p, q) = 0 \iff p = q$.

General strategy

1. Define a family of distributions $\mathcal{F} = \{p_\theta \in \mathcal{P}(\mathbb{R}^{d_x}) | \theta \in \Theta\}$. Typically this is done by
   1. Defining a neural network architecture f_θ parameterized by θ
   2. Define a reference distribution λ[6]
   3. $p_\theta := \text{Law}(f_\theta(Z))$ where $Z \sim \lambda$.



2. Choose a divergence[7] over distributions $D : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_*$
3. Solve

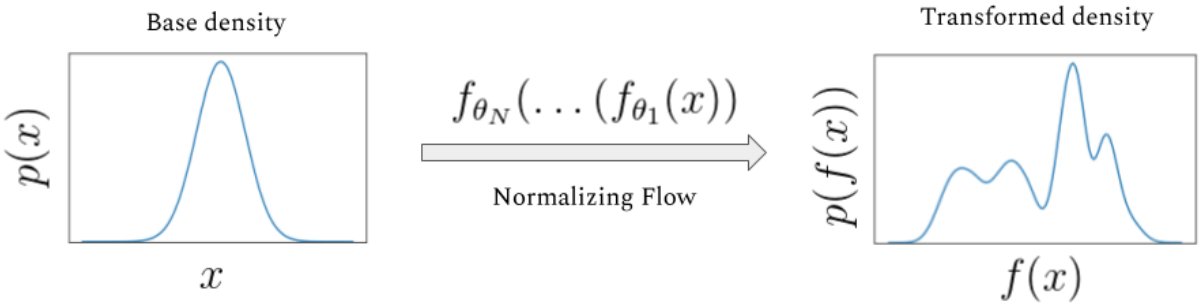
$$\theta^* \in \arg\min_{\theta \in \Theta} D(\pi, p_\theta)$$

[6] Typically, a Gaussian or Uniform.

[7] Only property we need is $D(p, q) = 0 \iff p = q$.

General strategy

1. Define a family of distributions $\mathcal{F} = \{p_\theta \in \mathcal{P}(\mathbb{R}^{d_x}) | \theta \in \Theta\}$. Typically this is done by
   1. Defining a neural network architecture f_θ parameterized by θ
   2. Define a reference distribution λ[6]
   3. $p_\theta := \text{Law}(f_\theta(Z))$ where $Z \sim \lambda$.



2. Choose a divergence[7] over distributions $D : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_*$
3. Solve

$$\theta^* \in \arg\min_{\theta \in \Theta} D(\pi, p_\theta)$$

[6] Typically, a Gaussian or Uniform.

[7] Only property we need is $D(p, q) = 0 \iff p = q$.

Kullback Leibler

Let $(p, q) \in \mathcal{P}^2$ such that $p \ll q$

$$\mathrm{D_{KL}}\left(p||q\right) := \mathbb{E}_p\left[\log \frac{\partial p(X)}{\partial q(X)}\right] . \tag{1}$$

If they both admit a density with respect to a given measure μ then

$$\mathrm{D_{KL}}\left(p||q\right) = \int \log \frac{p(x)}{q(x)} p(x)\mu(\mathrm{d}x) . \tag{2}$$

Some properties

1. $\mathrm{D_{KL}}\left(p||q\right) = 0 \iff p = q$
2. $\mathrm{D_{KL}}\left(p||q\right) \geq 0$
3. $\mathrm{D_{KL}}\left(\mathcal{N}(\mu_0, \Sigma_0)||\mathcal{N}(\mu_1, \Sigma_1)\right) =$
   $\frac{1}{2}\left[\log \frac{|\Sigma_0|}{|\Sigma_1|} - d + \|\mu_1 - \mu_0\|^2_{\Sigma_1^{-1}} + \mathrm{tr}(\Sigma_1^{-1}\Sigma_0)\right]$.[8]

[8] $\|v\|^2_A = v^t A v$.

Kullback Leibler

Let $(p, q) \in \mathcal{P}^2$ such that $p \ll q$

$$\mathrm{D_{KL}}\left(p||q\right) := \mathbb{E}_p\left[\log \frac{\partial p(X)}{\partial q(X)}\right] . \tag{1}$$

If they both admit a density with respect to a given measure μ then

$$\mathrm{D_{KL}}\left(p||q\right) = \int \log \frac{p(x)}{q(x)} p(x)\mu(\mathrm{d}x) . \tag{2}$$

Some properties

1. $\mathrm{D_{KL}}\left(p||q\right) = 0 \iff p = q$
2. $\mathrm{D_{KL}}\left(p||q\right) \geq 0$
3. $\mathrm{D_{KL}}\left(\mathcal{N}(\mu_0, \Sigma_0)||\mathcal{N}(\mu_1, \Sigma_1)\right) =$
   $\frac{1}{2}\left[\log \frac{|\Sigma_0|}{|\Sigma_1|} - d + \|\mu_1 - \mu_0\|^2_{\Sigma_1^{-1}} + \mathrm{tr}(\Sigma_1^{-1}\Sigma_0)\right]$.[8]

[8] $\|v\|^2_A = v^t A v$.

Kullback Leibler and Maximum Likelihood

$$\mathrm{D}_{\mathrm{KL}}\left(p||q\right) = \int \log \frac{p(x)}{q(x)} p(x)\mu(\mathrm{d}x) = \mathbb{E}_p\left[\log p(X)\right] - \mathbb{E}_p\left[\log q(X)\right] \ . \quad (3)$$

Thus,

$$\theta^* \in \arg\min_{\theta\in\Theta} \mathrm{D}_{\mathrm{KL}}\left(\pi||p_\theta\right) \iff \theta^* \in \arg\max_{\theta\in\Theta} \mathbb{E}_p\left[\log p_\theta(X)\right] \quad (4)$$

Minimizing KL $\iff$ doing Maximum likelihood.

So, maybe we can train our generative model by a Monte Carlo approximation of $\mathbb{E}_p\left[\log p_\theta(X)\right]$:

$$\arg\max_{\theta\in\Theta} n^{-1} \sum_{i=1}^{n} \log p_\theta\left(X_i\right) \quad (5)$$

Kullback Leibler and Maximum Likelihood

$$\mathrm{D}_{\mathrm{KL}}\left(p||q\right) = \int \log \frac{p(x)}{q(x)} p(x)\mu(\mathrm{d}x) = \mathbb{E}_p\left[\log p(X)\right] - \mathbb{E}_p\left[\log q(X)\right] \ . \quad (3)$$

Thus,

$$\theta^* \in \arg\min_{\theta\in\Theta} \mathrm{D}_{\mathrm{KL}}\left(\pi||p_\theta\right) \iff \theta^* \in \arg\max_{\theta\in\Theta} \mathbb{E}_p\left[\log p_\theta(X)\right] \quad (4)$$

Minimizing KL $\iff$ doing Maximum likelihood.

So, maybe we can train our generative model by a Monte Carlo approximation of $\mathbb{E}_p\left[\log p_\theta(X)\right]$:

$$\arg\max_{\theta\in\Theta} n^{-1} \sum_{i=1}^{n} \log p_\theta\left(X_i\right) \quad (5)$$

Kullback Leibler and Maximum Likelihood

$$\mathrm{D}_{\mathrm{KL}}\left(p||q\right) = \int \log \frac{p(x)}{q(x)} p(x)\mu(\mathrm{d}x) = \mathbb{E}_p\left[\log p(X)\right] - \mathbb{E}_p\left[\log q(X)\right] \ . \quad (3)$$

Thus,

$$\theta^* \in \arg\min_{\theta\in\Theta} \mathrm{D}_{\mathrm{KL}}\left(\pi||p_\theta\right) \iff \theta^* \in \arg\max_{\theta\in\Theta} \mathbb{E}_p\left[\log p_\theta(X)\right] \quad (4)$$

Minimizing KL $\iff$ doing Maximum likelihood.

So, maybe we can train our generative model by a Monte Carlo approximation of $\mathbb{E}_p\left[\log p_\theta(X)\right]$:

$$\arg\max_{\theta\in\Theta} n^{-1} \sum_{i=1}^{n} \log p_\theta\left(X_i\right) \quad (5)$$

$$\mathrm{D}_{\mathrm{KL}}\left(p||q\right) = \int \log \frac{p(x)}{q(x)} p(x)\mu(\mathrm{d}x) = \mathbb{E}_p\left[\log p(X)\right] - \mathbb{E}_p\left[\log q(X)\right] \ . \quad (3)$$

Thus,

$$\theta^* \in \arg\min_{\theta\in\Theta} \mathrm{D}_{\mathrm{KL}}\left(\pi||p_\theta\right) \iff \theta^* \in \arg\max_{\theta\in\Theta} \mathbb{E}_p\left[\log p_\theta(X)\right] \quad (4)$$

Minimizing KL $\iff$ doing Maximum likelihood.

So, maybe we can train our generative model by a Monte Carlo approximation of $\mathbb{E}_p\left[\log p_\theta(X)\right]$:

$$\arg\max_{\theta\in\Theta} n^{-1} \sum_{i=1}^{n} \log p_\theta\left(X_i\right) \quad (5)$$

Can you calculate $\log p_\theta$?

Outline

Change of variable formula

Suppose $f_\theta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ is invertible. Then,

$$p_\theta(x) = \lambda(f_\theta^{-1}(x))|\mathrm{J}_{f_\theta}(f_\theta^{-1}(x))|^{-1} . \tag{6}$$

Normalizing flow (Rezende and Mohamed, 2015)

A Normalizing flow is a Neural net

$$f_\theta(x) = \mathcal{T}_{\theta_k} \circ \cdots \circ \mathcal{T}_{\theta_0} , \tag{7}$$

where for all $\ell = 0, \cdots, k$, $\mathcal{T}_{\theta_\ell}$ is invertible and has an easy to calculate $|\mathrm{J}_{\mathcal{T}_{\theta_\ell}}|$!

Change of variable formula

Suppose $f_\theta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ is invertible. Then,

$$p_\theta(x) = \lambda(f_\theta^{-1}(x))|\mathrm{J}_{f_\theta}(f_\theta^{-1}(x))|^{-1} . \tag{6}$$

Normalizing flow (Rezende and Mohamed, 2015)

A Normalizing flow is a Neural net

$$f_\theta(x) = \mathcal{T}_{\theta_k} \circ \cdots \circ \mathcal{T}_{\theta_0} , \tag{7}$$

where for all $\ell = 0, \cdots, k$, $\mathcal{T}_{\theta_\ell}$ is invertible and has an easy to calculate $|\mathrm{J}_{\mathcal{T}_{\theta_\ell}}|$!

Algorithm 1 Log likelihood with NF

procedure LOGLIK
Input: x, θ.
Output: $p_\theta(x)$.
    $L = 0$, $z = x$.
    for $\ell = k, \cdots, 0$ **do**
        $z = \mathcal{T}_{\theta_\ell}^{-1}(z)$.
        $L = L - \log |\mathcal{T}_{\theta_\ell}(z)|$.
    end for
    $L = L + \log \lambda(z)$.
    Return: L
end procedure

It is possible to

$$\arg\max_{\theta \in \Theta} n^{-1} \sum_{i=1}^{n} \log p_\theta (X_i) \tag{8}$$

Algorithm 2 Log likelihood with NF

procedure LOGLIK
Input: x, θ.
Output: $p_\theta(x)$.
    $L = 0$, $z = x$.
    for $\ell = k, \cdots, 0$ **do**
        $z = \mathcal{T}_{\theta_\ell}^{-1}(z)$.
        $L = L - \log |\mathcal{T}_{\theta_\ell}(z)|$.
    end for
    $L = L + \log \lambda(z)$.
    Return: L
end procedure

It is possible to

$$\arg\max_{\theta \in \Theta} n^{-1} \sum_{i=1}^{n} \log p_\theta (X_i) \tag{8}$$

Illustration

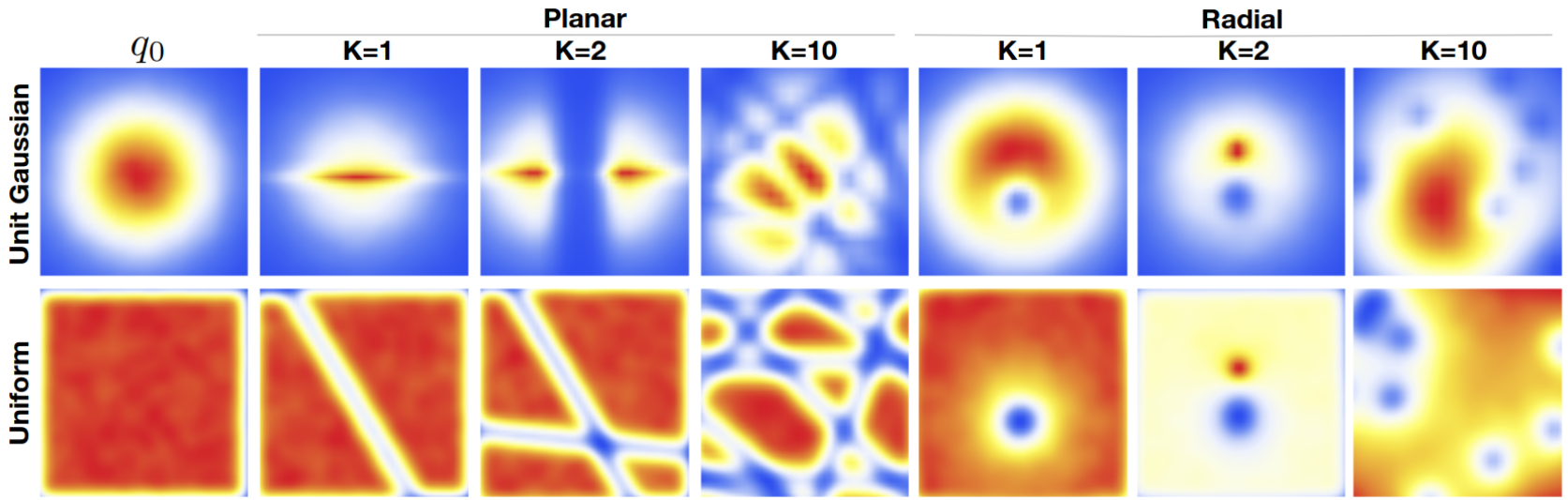


Figure: Illustration from Rezende and Mohamed, 2015 show the transformations obtained during a flow.

RealNVP (Dinh, Sohl-Dickstein, and Bengio, 2017)

$$\mathcal{T}_\theta(x[1{:}d], x[d{:}d_x]) = (x[1{:}d], x[d{:}d_x] \cdot e^{s(x[1:d];\theta)} + t(x[1{:}d];\theta)) \,, \quad (9)$$

where $s(\cdot;\theta), t(\cdot;\theta) : \mathbb{R}^d \to \mathbb{R}^{d_x - d}$ are typically neural networks.
The Jacobian of such flow is

$$\mathrm{J}_{\mathcal{T}_\theta}(x) = \begin{bmatrix} \mathrm{I}d & 0 \\ \dots & \mathrm{diag}(e^{s(x[1:d];\theta)}) \end{bmatrix} \,, \quad (10)$$

and thus $\log |\mathrm{J}_{\mathcal{T}_\theta}(x)| = \sum_{j=1}^{d_x - d} s(x[1{:}d];\theta)[j]$.

Illustration RNVP

Figure: Figure taken from Dinh, Sohl-Dickstein, and Bengio, 2017 illustrating generated samples from RealNVP on the Celeb dataset.

Problems with Normalizing flow

1. Considerable restriction on network architectures.
2. Manifold hypothesis: p_θ admits a density with respect to the ambient Lebesgue measure!

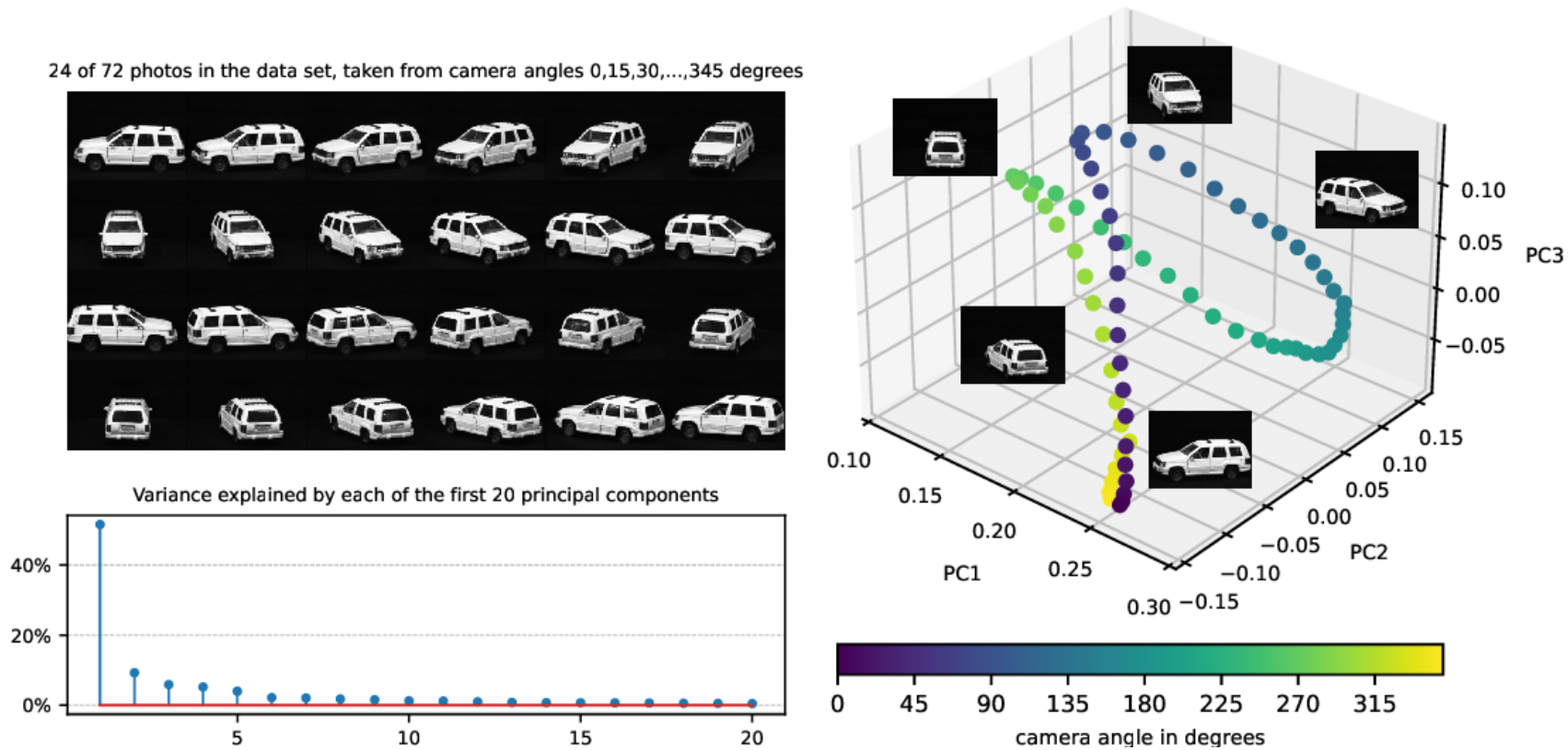


Figure: Taken from (Whiteley, Gray, and Rubin-Delanchy, 2024)

Outline

Latent model

Context: We think our data is lower dimensional.

Latent model

Context: We think our data is lower dimensional.

1. Define a latent distribution $\lambda \in \mathcal{P}(\mathbb{R}^{d_z})$, $d_z \ll d_x$.

Latent model

Context: We think our data is lower dimensional.

1. Define a latent distribution $\lambda \in \mathcal{P}(\mathbb{R}^{d_z})$, $d_z \ll d_x$.
2. Define a conditional distribution over the data space:

$$z \in \mathbb{R}^{d_z} \to p_\theta(\cdot|z) \in \mathcal{P}(\mathbb{R}^{d_x}) .$$

1

[1] Taken from Diederik P Kingma, Welling, et al., 2019.

Latent model

Context: We think our data is lower dimensional.

1. Define a latent distribution $\lambda \in \mathcal{P}(\mathbb{R}^{d_z})$, $d_z \ll d_x$.
2. Define a conditional distribution over the data space:

$$z \in \mathbb{R}^{d_z} \to p_\theta(\cdot|z) \in \mathcal{P}(\mathbb{R}^{d_x}) .$$

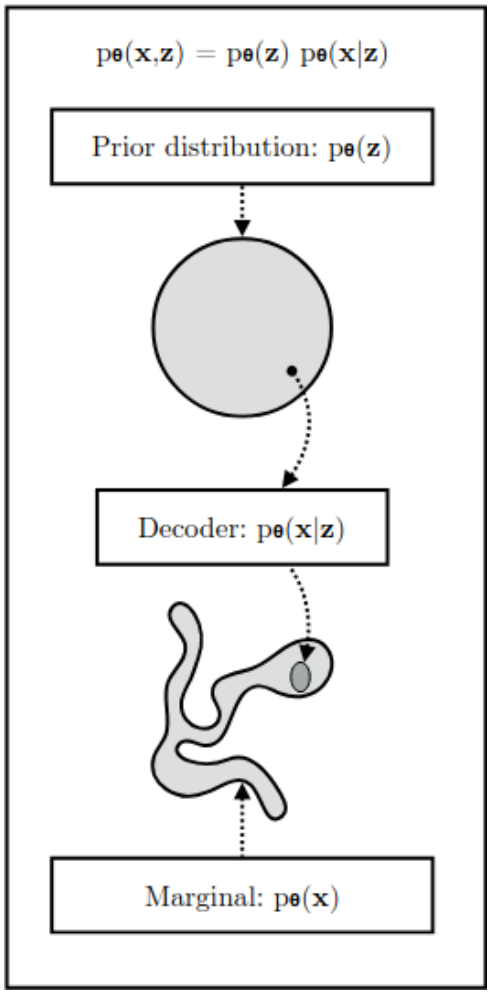


Figure: Illustration of the latent model [1].

[1] Taken from Diederik P Kingma, Welling, et al., 2019.

Latent model

Context: We think our data is lower dimensional.

1. Define a latent distribution $\lambda \in \mathcal{P}(\mathbb{R}^{d_z})$, $d_z \ll d_x$.
2. Define a conditional distribution over the data space:

$$z \in \mathbb{R}^{d_z} \to p_\theta(\cdot|z) \in \mathcal{P}(\mathbb{R}^{d_x}).$$

Gaussian Gaussian Latent model

We can consider

1. $\lambda = \mathcal{N}(0, \mathrm{I})$,
2. $p_\theta(\cdot|z) = \mathcal{N}(\mu_\theta(z), \sigma^2 \mathrm{I})$.

For $\mu_\theta(z) = Az$ we recover the (Probabilistic) PCA.
In general, μ_θ will be a Neural network.

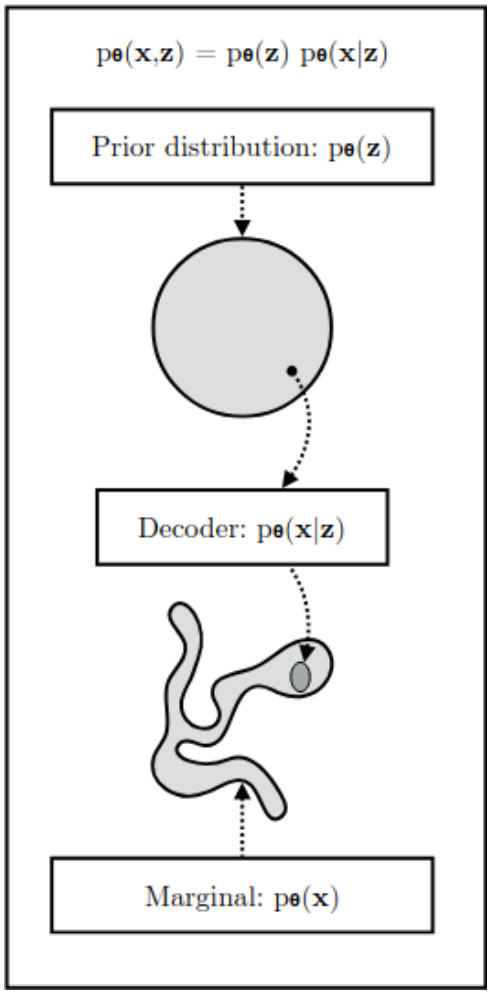


Figure: Illustration of the latent model [1].

[1] Taken from Diederik P Kingma, Welling, et al., 2019.

Can you calculate $\log p_\theta$?

For the VAE,

$$\log p_\theta(x) = \log \int p_\theta(x|z)\,\lambda(\mathrm{d}z)\,. \tag{11}$$

The answer is: Yes, but only for linear μ_θ!

Can you calculate $\log p_\theta$?

For the VAE,

$$\log p_\theta(x) = \log \int p_\theta(x|z)\, \lambda(\mathrm{d}z)\,. \tag{11}$$

The answer is: Yes, but only for linear μ_θ!

Can you estimate $\log p_\theta(x) = \log \int p_\theta(x|z)\,\lambda(\mathrm{d}z)$?

Naive MC:

1. Sample $Z_{1:K} \sim \lambda$
2. Approximate $p_\theta(x)$ by $K^{-1}\sum_{i=1}^{K} p_\theta(x|Z_i)$.

Problem: This estimation has high variance for realistic cases!

Can you estimate $\log p_\theta(x) = \log \int p_\theta(x|z)\,\lambda(\mathrm{d}z)$?

Naive MC:

1. Sample $Z_{:1:K} \sim \lambda$
2. Approximate $p_\theta(x)$ by $K^{-1}\sum_{i=1}^{K} p_\theta(x|Z_i)$.

Problem: This estimation has high variance for realistic cases!

Can you estimate $\log p_\theta(x) = \log \int p_\theta(x|z)\,\lambda(\mathrm{d}z)$?

Naive MC:

1. Sample $Z_{:1:K} \sim \lambda$
2. Approximate $p_\theta(x)$ by $K^{-1}\sum_{i=1}^{K} p_\theta(x|Z_i)$.

Problem: This estimation has high variance for realistic cases!

The VAE solution (Diederik P. Kingma and Welling, 2013).

Introduce a parametric family of instrumental distribution $q_\phi(\cdot|x) \in \mathcal{P}(\mathbb{R}^{d_z})$ to reduce MC variance.

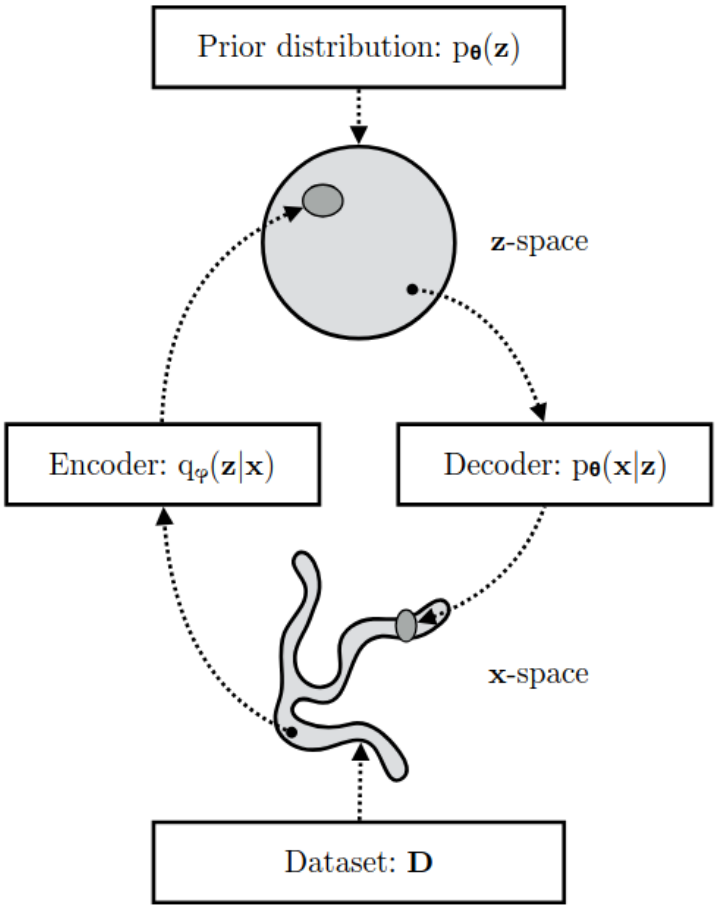


Figure: Overall view of the VAE.[1]

[1] Taken from Diederik P Kingma, Welling, et al., 2019.

The VAE solution (Diederik P. Kingma and Welling, 2013).

Introduce a parametric family of instrumental distribution $q_\phi(\cdot|x) \in \mathcal{P}(\mathbb{R}^{d_z})$ to reduce MC variance.

Make it match the true (intractable) posterior:

$$p_\theta(z|x) = \frac{p_\theta(x|z)\,\lambda(z)}{p_\theta(x)}\,. \qquad (12)$$

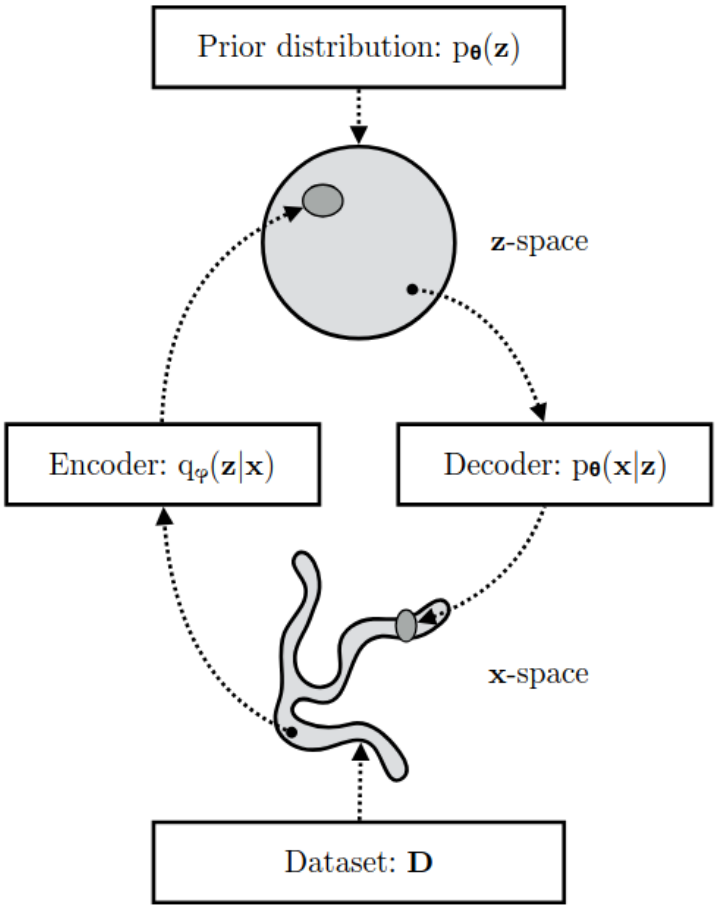


Figure: Overall view of the VAE.[1]

[1] Taken from Diederik P Kingma, Welling, et al., 2019.

Matching the posterior with Kullback Leibler

$$\mathsf{D}_{\mathsf{KL}}\left(q_\phi\left(\cdot|x\right)||p_\theta\left(\cdot|x\right)\right) = \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(z|x\right)} q_\phi\left(z|x\right) \mathrm{d}z$$

Matching the posterior with Kullback Leibler

$$\begin{aligned}
\mathsf{D}_{\mathsf{KL}}\left(q_\phi\left(\cdot|x\right)||p_\theta\left(\cdot|x\right)\right) &= \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(z|x\right)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \int \log \frac{q_\phi\left(z|x\right) p_\theta\left(x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z
\end{aligned}$$

Matching the posterior with Kullback Leibler

$$
\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||p_\theta\left(\cdot|x\right)\right) &= \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(z|x\right)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \int \log \frac{q_\phi\left(z|x\right) p_\theta\left(x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z
\end{aligned}
$$

Matching the posterior with Kullback Leibler

$$
\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||p_\theta\left(\cdot|x\right)\right) &= \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(z|x\right)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \int \log \frac{q_\phi\left(z|x\right) p_\theta\left(x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \int \log \frac{q_\phi\left(z|x\right)}{\lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z - \int \log p_\theta\left(x|z\right) q_\phi\left(z|x\right) \mathrm{d}z
\end{aligned}
$$

Matching the posterior with Kullback Leibler

$$
\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||p_\theta\left(\cdot|x\right)\right) &= \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(z|x\right)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \int \log \frac{q_\phi\left(z|x\right) p_\theta\left(x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \int \log \frac{q_\phi\left(z|x\right)}{\lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z - \int \log p_\theta\left(x|z\right) q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right) - \mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] .
\end{aligned}
$$

Matching the posterior with Kullback Leibler

$$
\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||p_\theta\left(\cdot|x\right)\right) &= \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(z|x\right)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \int \log \frac{q_\phi\left(z|x\right) p_\theta\left(x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \int \log \frac{q_\phi\left(z|x\right)}{p_\theta\left(x|z\right) \lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \int \log \frac{q_\phi\left(z|x\right)}{\lambda(z)} q_\phi\left(z|x\right) \mathrm{d}z - \int \log p_\theta\left(x|z\right) q_\phi\left(z|x\right) \mathrm{d}z \\
&= \log p_\theta\left(x\right) + \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right) - \mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] .
\end{aligned}
$$

Evidence lower bound (ELBO)

$$
\begin{aligned}
m(\theta, \phi; x) &:= \log p_\theta\left(x\right) - \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||p_\theta\left(\cdot|x\right)\right) \\
&= \mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right) .
\end{aligned}
$$

Evidence lower bound (ELBO)

$$
\begin{aligned}
m(\theta, \phi; x) &:= \log p_\theta(x) - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, p_\theta(\cdot|x)\right) \\
&= \mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta(x|Z)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, \lambda\right) \ .
\end{aligned}
$$

Evidence lower bound (ELBO)

$$
\begin{aligned}
m(\theta, \phi; x) &:= \log p_\theta(x) - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, p_\theta(\cdot|x)\right) \\
&= \mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta(x|Z)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, \lambda\right) \ .
\end{aligned}
$$

1. Maximize $m(\theta, \phi; x)$ in $\phi \Rightarrow$ Minimize $\mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, p_\theta(\cdot|x)\right)$.

Evidence lower bound (ELBO)

$$
\begin{aligned}
m(\theta, \phi; x) &:= \log p_\theta(x) - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, p_\theta(\cdot|x)\right) \\
&= \mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta(x|Z)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, \lambda\right) \ .
\end{aligned}
$$

1. Maximize $m(\theta, \phi; x)$ in $\phi \Rightarrow$ Minimize $\mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, p_\theta(\cdot|x)\right)$.
2. Maximize $m(\theta, \phi; x)$ in $\theta \Rightarrow$ Maximize $\log p_\theta(x)$.

Evidence lower bound (ELBO)

$$
\begin{aligned}
m(\theta, \phi; x) &:= \log p_\theta(x) - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, p_\theta(\cdot|x)\right) \\
&= \mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta(x|Z)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, \lambda\right) \ .
\end{aligned}
$$

1. Maximize $m(\theta, \phi; x)$ in $\phi \Rightarrow$ Minimize $\mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, p_\theta(\cdot|x)\right)$.
2. Maximize $m(\theta, \phi; x)$ in $\theta \Rightarrow$ Maximize $\log p_\theta(x)$.

We know $\log p_\theta(x) - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, p_\theta(\cdot|x)\right)$ is intractable. Is $\mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta(x|Z)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi(\cdot|x) \,||\, \lambda\right)$ tractable?

Calculating the ELBO

$$\mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right)$$

Calculating the ELBO

$$\mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right)$$

Gaussian Gaussian Latent model

Consider

1. $\lambda = \mathcal{N}(0, \mathrm{I})$,
2. $p_\theta\left(\cdot|z\right) = \mathcal{N}(\mu_\theta(z), \sigma^2\mathrm{I})$,
3. $q_\phi\left(\cdot|x\right) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$.

Here, we consider $\Sigma_\phi(x) = \mathrm{diag}(\sigma^2_\phi(\mathrm{x}))$ where $\sigma^2_\phi(x) \in \mathbb{R}^{d_z}$.

Calculating the ELBO

$$\mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right)$$

Gaussian Gaussian Latent model

Consider

1. $\lambda = \mathcal{N}(0, \mathrm{I})$,
2. $p_\theta\left(\cdot|z\right) = \mathcal{N}(\mu_\theta(z), \sigma^2\mathrm{I})$,
3. $q_\phi\left(\cdot|x\right) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$.

Here, we consider $\Sigma_\phi(x) = \mathrm{diag}(\sigma_\phi^2(\mathrm{x}))$ where $\sigma_\phi^2(x) \in \mathbb{R}^{d_z}$.

$$\mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right) = \frac{1}{2}\left[\|\mu_\phi(x)\|^2 + \mathrm{tr}(\Sigma_\phi(x)) + \log|\Sigma_\phi(x)| - d_z\right] \quad (13)$$

Calculating the ELBO

$$\mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right)$$

Gaussian Gaussian Latent model

Consider

1. $\lambda = \mathcal{N}(0, \mathrm{I})$,
2. $p_\theta\left(\cdot|z\right) = \mathcal{N}(\mu_\theta(z), \sigma^2\mathrm{I})$,
3. $q_\phi\left(\cdot|x\right) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$.

Here, we consider $\Sigma_\phi(x) = \mathrm{diag}(\sigma_\phi^2(\mathrm{x}))$ where $\sigma_\phi^2(x) \in \mathbb{R}^{d_z}$.

$$\mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right) = \frac{1}{2}\left[\|\mu_\phi(x)\|^2 + \mathrm{tr}(\Sigma_\phi(x)) + \log|\Sigma_\phi(x)| - d_z\right] \quad (13)$$

$$\mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] = \mathbb{E}_{q_\phi(\cdot|x)}\left[-\frac{1}{2\sigma^2}\|x - \mu_\theta(Z)\|^2\right] + cte. \quad (14)$$

The reparametrization trick for $\mathbb{E}_{q_\phi(\cdot|x)}\left[-\frac{1}{2\sigma^2}\|x-\mu_\theta(Z)\|^2\right]$

$$q_\phi(\cdot|x) = \mathcal{N}(\mu_\phi(x), diag(\sigma_\phi^2(x))).$$

One way of sampling:

1. Sample $\epsilon \sim \mathcal{N}(0, \mathrm{I})$
2. Set $Z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$.

Thus,

$$\mathbb{E}_{q_\phi(\cdot|x)}\left[-\frac{1}{2\sigma^2}\|x-\mu_\theta(Z)\|^2\right] = \mathbb{E}_{\mathcal{N}(0,\mathrm{I})}\left[-\frac{1}{2\sigma^2}\|x-\mu_\theta(\mu_\phi(x)+\sigma_\phi(x)\odot\epsilon)\|^2\right]$$

We can approximate with MC

$$\begin{aligned}&\mathbb{E}_{\mathcal{N}(0,\mathrm{I})}\left[-\frac{1}{2\sigma^2}\|x-\mu_\theta(\mu_\phi(x)+\sigma_\phi(x)\odot\epsilon)\|^2\right]\\ &\approx K^{-1}\sum_{k=1}^{K}-\frac{1}{2\sigma^2}\|x-\mu_\theta(\mu_\phi(x)+\sigma_\phi(x)\odot\epsilon_k)\|^2\end{aligned}$$

where $\epsilon_k \sim \mathcal{N}(0, \mathrm{I})$.

Calculating the ELBO: Recap

$$\mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] - \mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right)$$

Gaussian Gaussian Latent model

Consider

1. $\lambda = \mathcal{N}(0, \mathrm{I})$,
2. $p_\theta\left(\cdot|z\right) = \mathcal{N}(\mu_\theta(z), \sigma^2\mathrm{I})$,
3. $q_\phi\left(\cdot|x\right) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$.

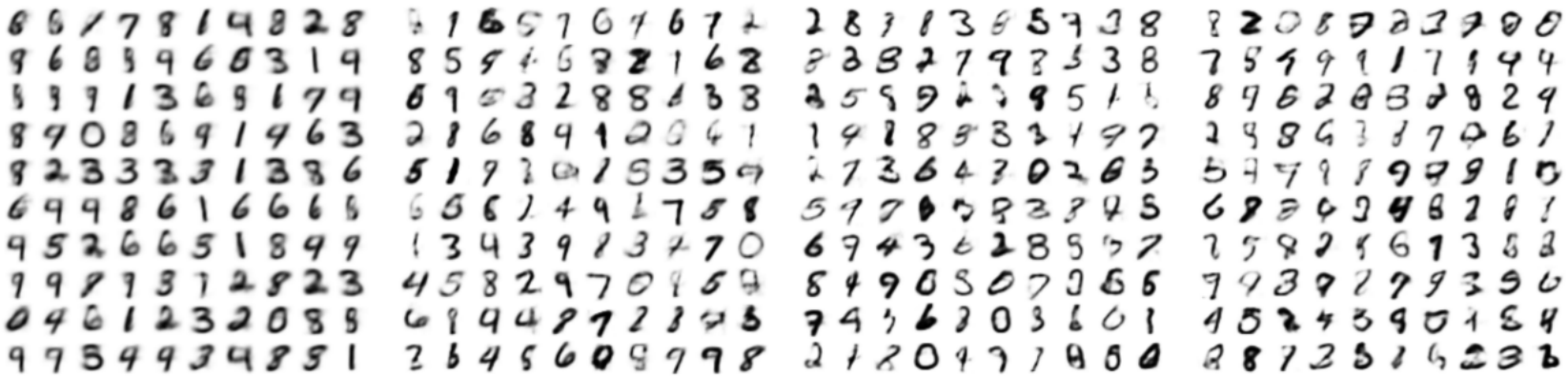
Here, we consider $\Sigma_\phi(x) = \mathrm{diag}(\sigma^2_\phi(\mathrm{x}))$ where $\sigma^2_\phi(x) \in \mathbb{R}^{d_z}$.

$$\mathrm{D}_{\mathrm{KL}}\left(q_\phi\left(\cdot|x\right)||\lambda\right) = \frac{1}{2}\left[\|\mu_\phi(x)\|^2 + \mathrm{tr}(\Sigma_\phi(x)) + \log|\Sigma_\phi(x)| - d_z\right] \quad (15)$$

$$\mathbb{E}_{q_\phi(\cdot|x)}\left[\log p_\theta\left(x|Z\right)\right] \approx K^{-1}\sum_{k=1}^{K} -\frac{1}{2\sigma^2}\|x - \mu_\theta(\mu_\phi(x) + \sigma_\phi(x)\odot\epsilon_k)\|^2 + cte. \quad (16)$$

We can calculate gradients!

VAE on MNIST



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure: MNIST with different latent dimension.

1. VAE are good encoders! (Encode + Decode generalizes well!) ✓
2. Sampling from the prior fails for high dimensional latent models! x

$$Z \sim \lambda\,, \quad X \sim p_\theta\left(\cdot|Z\right)$$

[8] Taken from Diederik P. Kingma and Welling, 2013.

Outline

Bypassing log density completely: Duality

Riesz representation theorem

Every positive linear functional L in $C_c(\mathbb{R}^{d_x})$ can be represented by $L(f) = \mathbb{E}_\mu [f(X)]$ for some $\mu \in \mathcal{P}$.

Maybe, for some divergences, we can bypass calculating log densities by some duality theorem!

For us a duality theorem for a given divergence D allows to evaluate the divergence D between p_0 and p_1 by just considering their effect on functions in a given class!

Bypassing log density completely: Duality

Riesz representation theorem

Every positive linear functional L in $C_c(\mathbb{R}^{d_x})$ can be represented by $L(f) = \mathbb{E}_\mu [f(X)]$ for some $\mu \in \mathcal{P}$.

Maybe, for some divergences, we can bypass calculating log densities by some duality theorem!

For us a duality theorem for a given divergence D allows to evaluate the divergence D between p_0 and p_1 by just considering their effect on functions in a given class!

Bypassing log density completely: Duality

Riesz representation theorem

Every positive linear functional L in $C_c(\mathbb{R}^{d_x})$ can be represented by $L(f) = \mathbb{E}_\mu [f(X)]$ for some $\mu \in \mathcal{P}$.

Maybe, for some divergences, we can bypass calculating log densities by some duality theorem!

For us a duality theorem for a given divergence D allows to evaluate the divergence D between p_0 and p_1 by just considering their effect on functions in a given class!

Duality for the KL

Theorem (Donsker and Varadhan, 1983)

Let $(q, p) \in \mathcal{P}$, such that $p \ll q$ and $q \ll p$. Then,

$$\mathrm{D}_{\mathrm{KL}}\left(q||p\right) = \sup_{f \in \mathcal{G}_q} \mathbb{E}_q\left[f\right] - \log \mathbb{E}_p\left[\exp(f)\right] \ , \tag{17}$$

where $\mathcal{G}_q$ is the set of measurable g such that $\mathbb{E}_q\left[\exp(g)\right] < \infty$.
Furthermore, equality hold if and only if $f = \log \frac{\mathrm{d}q}{\mathrm{d}p}$.

Duality for Jensen-Shannon

Theorem (Jensen Shannon duality Goodfellow et al., 2014)

Let $(q, p) \in \mathcal{P}$, such that $p \ll q$ and $q \ll p$. Define the Jensen-Shannon entropy as,

$$\mathrm{D}_{\mathrm{JS}}\left(q||p\right) = \frac{1}{2}\left(\mathrm{D}_{\mathrm{KL}}\left(q||p\right) + \mathrm{D}_{\mathrm{KL}}\left(p||q\right)\right) \ . \tag{18}$$

Then,

$$2\,\mathrm{D}_{\mathrm{JS}}\left(q||p\right) = \sup_{D \in \mathcal{C}\ell} \mathbb{E}_q\left[\log D(X)\right] + \mathbb{E}_p\left[\log(1 - D(X))\right] + \log(4)\ , \tag{19}$$

where $\mathcal{C}\ell$ is the set of measurable functions from $\mathbb{R}^{d_x} \to [0, 1]$.

Minimize $\sup_{D \in \mathcal{C}\ell} \mathbb{E}_q [\log D(X)] + \mathbb{E}_p [\log(1 - D(X))]$ $\Leftrightarrow$ Minimize $\mathrm{D}_{\mathrm{JS}}(q||p)$.

The GAN framework

1. Approximate $\mathcal{C}\ell$ by the set of classifiers defined by a Neural network architecture $\mathcal{D}_\phi = \{D_\phi : \mathbb{R}^{d_x} \to [0,1]\}_\phi$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$, and p_θ the law of X.
3. Train it in two steps:

Minimize $\sup_{D \in \mathcal{C}\ell} \mathbb{E}_q [\log D(X)] + \mathbb{E}_p [\log(1 - D(X))] \Leftrightarrow$ Minimize $\mathrm{D}_{\mathrm{JS}}(q||p)$.

The GAN framework

1. Approximate $\mathcal{C}\ell$ by the set of classifiers defined by a Neural network architecture $\mathrm{D}_\phi = \{D_\phi : \mathbb{R}^{d_x} \to [0,1]\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   - For fixed θ, set $\phi^* = \arg\max_\phi \mathbb{E}_q [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$
   - For fixed ϕ, set $\theta^* = \arg\min_\theta \mathbb{E}_q [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$

Minimize $\sup_{D \in \mathcal{C}\ell} \mathbb{E}_q [\log D(X)] + \mathbb{E}_p [\log(1 - D(X))] \Leftrightarrow$ Minimize $\mathrm{D}_{\mathrm{JS}}(q||p)$.

The GAN framework

1. Approximate $\mathcal{C}\ell$ by the set of classifiers defined by a Neural network architecture $\mathrm{D}_\phi = \{D_\phi : \mathbb{R}^{d_x} \to [0,1]\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   a. For fixed θ, set
   $$\phi = \arg\max_\phi \mathbb{E}_{p_\theta} [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$$
   b. For fixed ϕ, set
   $$\theta = \arg\min_\theta \mathbb{E}_{p_\theta} [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$$

Minimize $\sup_{D \in \mathcal{C}\ell} \mathbb{E}_q [\log D(X)] + \mathbb{E}_p [\log(1 - D(X))] \Leftrightarrow$ Minimize $\mathrm{D}_{\mathrm{JS}}(q||p)$.

The GAN framework

1. Approximate $\mathcal{C}\ell$ by the set of classifiers defined by a Neural network architecture $\mathrm{D}_\phi = \{D_\phi : \mathbb{R}^{d_x} \to [0,1]\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   1. For fixed θ, set
   $$\phi = \arg\max_\phi \mathbb{E}_\pi [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$$
   2. For fixed ϕ, set
   $$\theta = \arg\min_\theta \mathbb{E}_\pi [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$$

Minimize $\sup_{D \in \mathcal{C}\ell} \mathbb{E}_q [\log D(X)] + \mathbb{E}_p [\log(1 - D(X))] \Leftrightarrow$ Minimize $\mathrm{D}_{\mathrm{JS}}(q||p)$.

The GAN framework

1. Approximate $\mathcal{C}\ell$ by the set of classifiers defined by a Neural network architecture $\mathrm{D}_\phi = \{D_\phi : \mathbb{R}^{d_x} \to [0,1]\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   1. For fixed θ, set

$$\phi = \arg\max_\phi \mathbb{E}_\pi [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$$

   2. For fixed ϕ, set

$$\theta = \arg\min_\theta \mathbb{E}_\pi [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$$

Minimize $\sup_{D \in \mathcal{C}\ell} \mathbb{E}_q [\log D(X)] + \mathbb{E}_p [\log(1 - D(X))] \Leftrightarrow$ Minimize $\mathrm{D}_{\mathrm{JS}}(q||p)$.

The GAN framework

1. Approximate $\mathcal{C}\ell$ by the set of classifiers defined by a Neural network architecture $\mathrm{D}_\phi = \{D_\phi : \mathbb{R}^{d_x} \to [0,1]\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   1. For fixed θ, set
   $$\phi = \arg\max_\phi \mathbb{E}_\pi [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$$
   2. For fixed ϕ, set
   $$\theta = \arg\min_\theta \mathbb{E}_\pi [\log D_\phi(X)] + \mathbb{E}_\lambda [\log(1 - D_\phi(f_\theta(Z)))]$$

A closer look at GAN training

1. $\phi = \arg\max_\phi \overbrace{\mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]}^{-Cross-entropy}$
   This is equivalent to training a classifier to distinguish between real data $X_r \sim \pi$ from fake data $X_f \sim p_\theta$.
2. $\theta = \arg\min_\theta \mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]$ We are now training the generator to "fool" the classifier by minimizing the cross-entropy.

[8] $p_\theta = \text{Law}(f_\theta(Z))$ with $Z \sim \lambda$.

A closer look at GAN training

1. $\phi = \arg\max_\phi \overbrace{\mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]}^{-Cross-entropy}$
   This is equivalent to training a classifier to distinguish between real data $X_r \sim \pi$ from fake data $X_f \sim p_\theta$.
2. $\theta = \arg\min_\theta \mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]$ We are now training the generator to "fool" the classifier by minimizing the cross-entropy.

[8] $p_\theta = \text{Law}(f_\theta(Z))$ with $Z \sim \lambda$.

A closer look at GAN training

1. $\phi = \arg\max_\phi \overbrace{\mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]}^{-Cross-entropy}$
   This is equivalent to training a classifier to distinguish between real data $X_r \sim \pi$ from fake data $X_f \sim p_\theta$.
2. $\theta = \arg\min_\theta \mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]$ We are now training the generator to "fool" the classifier by augmenting the cross entropy.

[8] $p_\theta = \text{Law}(f_\theta(Z))$ with $Z \sim \lambda$.

A closer look at GAN training

1. $\phi = \arg\max_\phi \overbrace{\mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]}^{-Cross-entropy}$
   This is equivalent to training a classifier to distinguish between real data $X_r \sim \pi$ from fake data $X_f \sim p_\theta$.
2. $\theta = \arg\min_\theta \mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]$ We are now training the generator to "fool" the classifier by augmenting the cross entropy.

[8] $p_\theta = \text{Law}(f_\theta(Z))$ with $Z \sim \lambda$.

A closer look at GAN training

1. $\phi = \arg\max_\phi \overbrace{\mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]}^{-Cross-entropy}$
   This is equivalent to training a classifier to distinguish between real data $X_r \sim \pi$ from fake data $X_f \sim p_\theta$.
2. $\theta = \arg\min_\theta \mathbb{E}_\pi \left[\log D_\phi(X)\right] + \mathbb{E}_\lambda \left[\log(1 - D_\phi(f_\theta(Z)))\right]$ We are now training the generator to "fool" the classifier by augmenting the cross entropy.

[8] $p_\theta = \text{Law}(f_\theta(Z))$ with $Z \sim \lambda$.

A closer look at GAN training

$$\overbrace{\mathbb{E}_{\pi}\left[\log D_{\phi}(X)\right] + \mathbb{E}_{\lambda}\left[\log(1 - D_{\phi}(f_{\theta}(Z)))\right]}^{-Cross-entropy}$$

1. $\phi = \arg\max_{\phi}$
2. $\theta = \arg\min_{\theta} \mathbb{E}_{\pi}\left[\log D_{\phi}(X)\right] + \mathbb{E}_{\lambda}\left[\log(1 - D_{\phi}(f_{\theta}(Z)))\right]$

"In the proposed adversarial nets framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles."[9]

[9] from Goodfellow et al., 2014

Original GANs

1. No restriction on architecture ✓
2. No restriction on dimension of the latent space ✓
3. Training requires two networks x
4. Adversarial loss $\rightarrow$ unstable training x
5. Mode collapse x

Original GANs

1. No restriction on architecture ✓
2. No restriction on dimension of the latent space ✓
3. Training requires two networks x
4. Adversarial loss $\rightarrow$ unstable training x
5. Mode collapse x

Original GANs

1. No restriction on architecture ✓
2. No restriction on dimension of the latent space ✓
3. Training requires two networks x
4. Adversarial loss $\rightarrow$ unstable training x
5. Mode collapse x

Original GANs

1. No restriction on architecture ✓
2. No restriction on dimension of the latent space ✓
3. Training requires two networks x
4. Adversarial loss $\rightarrow$ unstable training x
5. Mode collapse x

Original GANs

1. No restriction on architecture ✓
2. No restriction on dimension of the latent space ✓
3. Training requires two networks x
4. Adversarial loss $\rightarrow$ unstable training x
5. Mode collapse x

An evolution: Wasserstein GANs

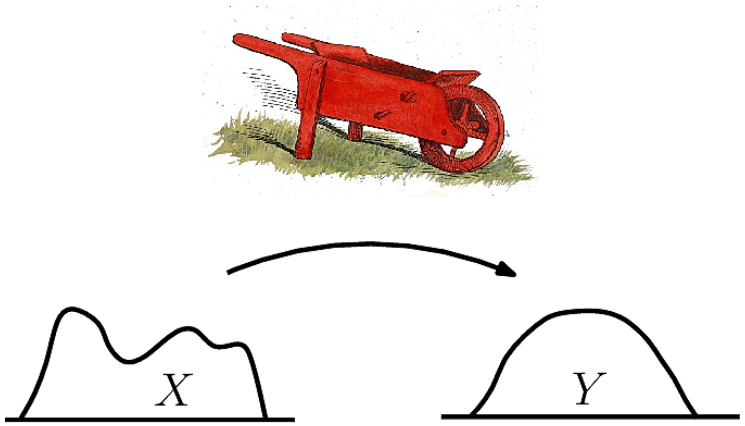


Figure: The earth mover problem

Wasserstein distance

Let $(q, p) \in (\mathcal{P}(\mathbb{R}^{d_x}))^2$, define the set of coupling distributions

$$C(q,p) = \{v \in \mathcal{P}(\mathbb{R}^{d_x} \times \mathbb{R}^{d_x}) | q(A) = v(A \times \mathbb{R}^{d_x}), p(A) = v(\mathbb{R}^{d_x} \times A)\} .$$

The Wasserstein distance is defined as

$$\mathcal{W}(q,p) = \inf_{v \in C(q,p)} \int \|x-y\| v(\mathrm{d}x, \mathrm{d}y) = \inf_{v \in C(q,p)} \mathbb{E}_{(X_0,X_1)\sim v} \left[\|X_0 - X_1\|\right] . \tag{20}$$

[9] Taken from `https://sbl.inria.fr/`.

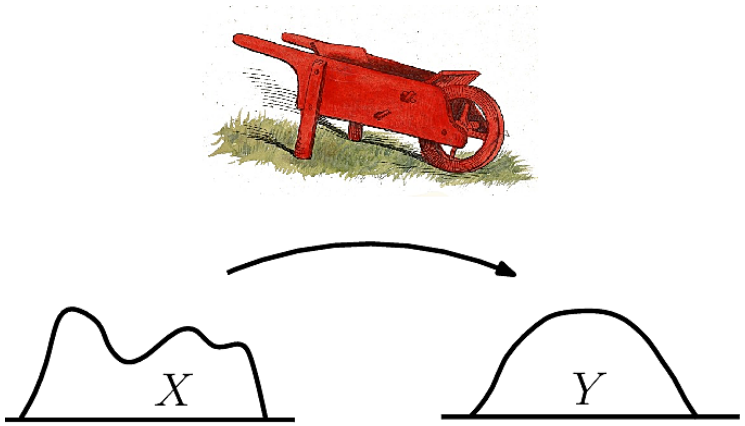


Figure: The earth mover problem

Wasserstein distance

Let $(q, p) \in (\mathcal{P}(\mathbb{R}^{d_x}))^2$, define the set of coupling distributions

$$C(q,p) = \{\upsilon \in \mathcal{P}(\mathbb{R}^{d_x} \times \mathbb{R}^{d_x}) | q(A) = \upsilon(A \times \mathbb{R}^{d_x}), p(A) = \upsilon(\mathbb{R}^{d_x} \times A)\} .$$

The Wasserstein distance is defined as

$$\mathcal{W}(q,p) = \inf_{\upsilon \in C(q,p)} \int \|x-y\| \upsilon(\mathrm{d}x, \mathrm{d}y) = \inf_{\upsilon \in C(q,p)} \mathbb{E}_{(X_0,X_1)\sim\upsilon} \left[\|X_0 - X_1\|\right] . \quad (20)$$

[9] Taken from `https://sbl.inria.fr/`.

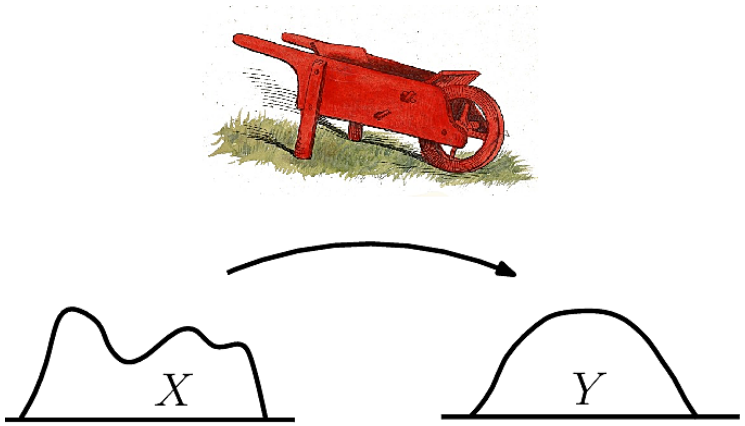


Figure: The earth mover problem

Wasserstein distance

Let $(q, p) \in (\mathcal{P}(\mathbb{R}^{d_x}))^2$, define the set of coupling distributions

$$C(q,p) = \{\upsilon \in \mathcal{P}(\mathbb{R}^{d_x} \times \mathbb{R}^{d_x}) | q(A) = \upsilon(A \times \mathbb{R}^{d_x}), p(A) = \upsilon(\mathbb{R}^{d_x} \times A)\} .$$

The Wasserstein distance is defined as

$$\mathcal{W}(q,p) = \inf_{\upsilon \in C(q,p)} \int \|x-y\| \upsilon(\mathrm{d}x, \mathrm{d}y) = \inf_{\upsilon \in C(q,p)} \mathbb{E}_{(X_0,X_1)\sim\upsilon} \left[\|X_0 - X_1\|\right] . \tag{20}$$

[9] Taken from `https://sbl.inria.fr/`.

Theorem (Kantorovich-Rubinstein theorem (see (Villani, 2021)))

$$\mathcal{W}(q,p) = \sup_{f \in \mathrm{Lip}_1} \mathbb{E}_q[f] - \mathbb{E}_p[f] \ . \tag{21}$$

where $\mathrm{Lip}_1 := \{f : \mathbb{R}^{d_x} \to \mathbb{R} || f(x) - f(y)| \leq \|x - y\|\}$.

In Arjovsky, Chintala, and Bottou, 2017 proposes changing $\mathrm{D}_{\mathrm{JS}}(\pi||p_\theta)$ by $\mathcal{W}(\pi, p_\theta)$ in the training of GANs.

Theorem (Kantorovich-Rubinstein theorem (see (Villani, 2021)))

$$\mathcal{W}(q,p) = \sup_{f \in \mathrm{Lip}_1} \mathbb{E}_q[f] - \mathbb{E}_p[f] \ . \tag{21}$$

where $\mathrm{Lip}_1 := \{f : \mathbb{R}^{d_x} \to \mathbb{R} || f(x) - f(y)| \leq \|x - y\|\}$.

In Arjovsky, Chintala, and Bottou, 2017 proposes changing $\mathrm{D}_{\mathrm{JS}}(\pi||p_\theta)$ by $\mathcal{W}(\pi, p_\theta)$ in the training of GANs.

Wasserstein GANs (Arjovsky, Chintala, and Bottou, 2017)

$$\mathcal{W}(q,p) = \sup_{f \in \mathrm{Lip}_1} \mathbb{E}_q[f] - \mathbb{E}_p[f] \ . \tag{22}$$

1. Approximate Lip_1 by a Lipschitz Neural network architecture $\mathrm{D}_\phi = \{g_\phi : \mathbb{R}^{d_x} \to \mathbb{R}\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   - For fixed θ, set $\phi \leftarrow \arg\max_\phi \mathbb{E}_p[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$
   - For fixed ϕ, set $\theta \leftarrow \arg\min_\theta \mathbb{E}_p[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$

Wasserstein GANs (Arjovsky, Chintala, and Bottou, 2017)

$$\mathcal{W}(q,p) = \sup_{f \in \text{Lip}_1} \mathbb{E}_q[f] - \mathbb{E}_p[f] \ . \tag{22}$$

1. Approximate Lip_1 by a Lipschitz Neural network architecture $\mathrm{D}_\phi = \{g_\phi : \mathbb{R}^{d_x} \to \mathbb{R}\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   1. For fixed θ, set
   $$\phi = \arg\max_\phi \mathbb{E}_\pi[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$$
   2. For fixed ϕ, set
   $$\theta = \arg\min_\theta \mathbb{E}_\pi[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$$

Wasserstein GANs (Arjovsky, Chintala, and Bottou, 2017)

$$\mathcal{W}(q,p) = \sup_{f \in \text{Lip}_1} \mathbb{E}_q[f] - \mathbb{E}_p[f] \ . \tag{22}$$

1. Approximate Lip_1 by a Lipschitz Neural network architecture $\mathrm{D}_\phi = \{g_\phi : \mathbb{R}^{d_x} \to \mathbb{R}\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   1. For fixed θ, set
   $$\phi = \arg\max_\phi \mathbb{E}_\pi[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$$
   2. For fixed ϕ, set
   $$\theta = \arg\min_\theta \mathbb{E}_\pi[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$$

Wasserstein GANs (Arjovsky, Chintala, and Bottou, 2017)

$$\mathcal{W}(q,p) = \sup_{f \in \text{Lip}_1} \mathbb{E}_q[f] - \mathbb{E}_p[f] \ . \tag{22}$$

1. Approximate Lip_1 by a Lipschitz Neural network architecture $\mathrm{D}_\phi = \{g_\phi : \mathbb{R}^{d_x} \to \mathbb{R}\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   1. For fixed θ, set
   $$\phi = \arg\max_\phi \mathbb{E}_\pi[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$$
   2. For fixed ϕ, set
   $$\theta = \arg\min_\theta \mathbb{E}_\pi[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$$

Wasserstein GANs (Arjovsky, Chintala, and Bottou, 2017)

$$\mathcal{W}(q,p) = \sup_{f \in \text{Lip}_1} \mathbb{E}_q[f] - \mathbb{E}_p[f] \ . \tag{22}$$

1. Approximate Lip_1 by a Lipschitz Neural network architecture $\mathrm{D}_\phi = \{g_\phi : \mathbb{R}^{d_x} \to \mathbb{R}\}$.
2. Generative model: $Z \in \mathbb{R}^{d_z} \sim \lambda$, apply a Neural net $X = f_\theta(Z)$. and p_θ the law of X.
3. Train it in two steps:
   1. For fixed θ, set
   $$\phi = \arg\max_\phi \mathbb{E}_\pi[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$$
   2. For fixed ϕ, set
   $$\theta = \arg\min_\theta \mathbb{E}_\pi[g_\phi(X)] - \mathbb{E}_\lambda[g_\phi(f_\theta(Z))]$$

No more mode collapse

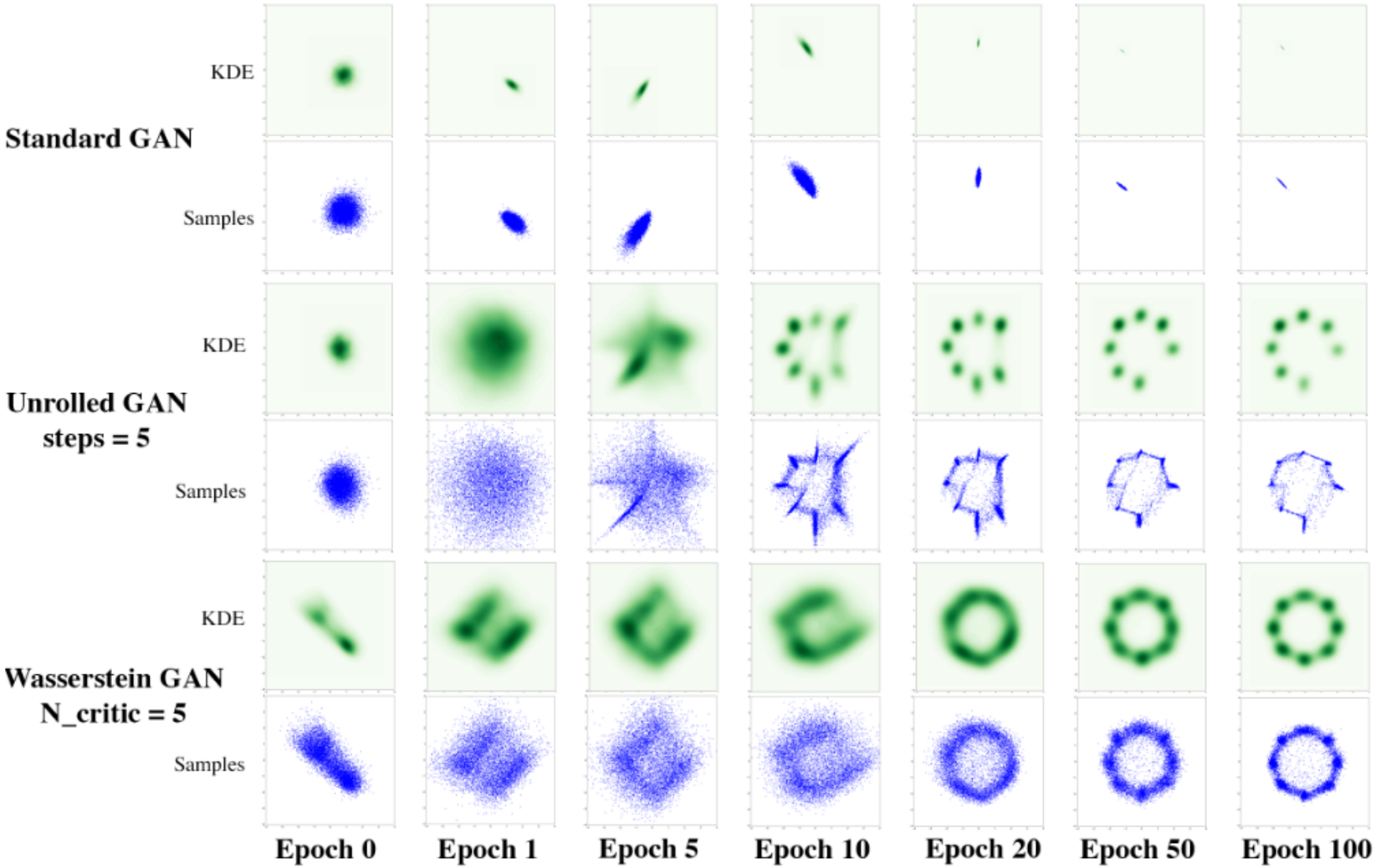


Figure: Comparison between GAN and Wasserstein GANs taken from Arjovsky, Chintala, and Bottou, 2017.

Wasserstein GANs

1. Same pros as the GANs ✓
2. No more mode collapse ✓
3. Growing literature on how to constraint Lipschitz norm of neural net ✓
4. still SOTA (state of the art) in some image tasks ✓
5. Training still requires some care !

Figure: Uncurated images from StyleGAN Karras, Laine, and Aila, 2019.

Outline

Noising and Denoising

Figure: $X_0 \sim \pi$.

Figure: $X_t = X_0 + \sigma_t Z_t$. [10]1

Suppose we have an increasing noise schedule

$$\sigma : [0, T] \to \sigma_t \in \mathbb{R}_* \ .$$

Can we reverse the procedure?

[1] $Z_t \perp X_0$, $Z_t \sim \mathcal{N}(0, \mathrm{I})$.

Noising and Denoising

Suppose we have an increasing noise schedule

$$\sigma : [0, T] \to \sigma_t \in \mathbb{R}_* \ .$$

1. Noising kernel:

$$p_{t|0}(x_t|x_0) = \mathcal{N}(x_t; x_0, \sigma_t^2 \mathrm{I}) \ ,$$

2. Marginal:

$$p_t(x_t) = \int p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0) \ .$$

Our goal is to go backwards:

$$p_T \to p_{T-\Delta t} \to \cdots \to p_0 \ .$$

Figure: $X_0 \sim \pi$.

Figure:
$X_t = X_0 + \sigma_t Z_t$. [12]1

Generative models by score matching Y. Song et al., 2021

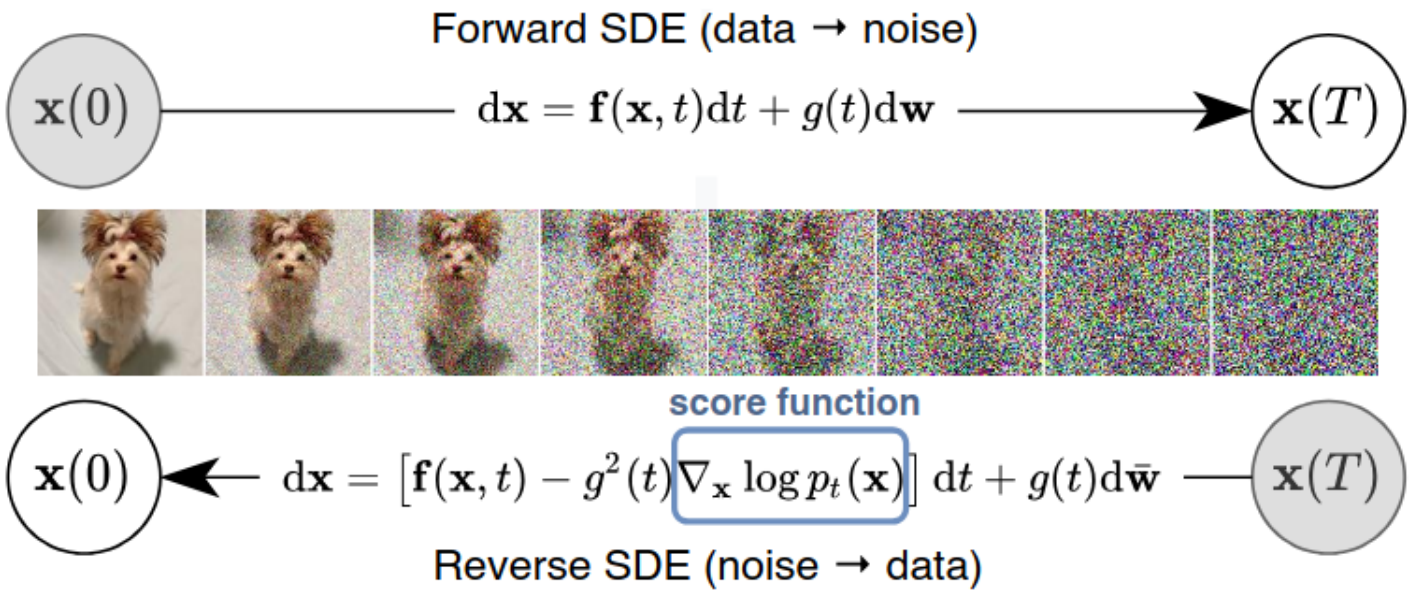


Figure: Illustration of noising and denoising with SDEs.

Our p_t is the time t marginal of the SDE

$$\mathrm{d}x = \sqrt{\frac{d\sigma_t^2}{dt}}\, dw \ .$$

It admits the backward decomposition

$$\mathrm{d}x = -\frac{d\sigma_t^2}{dt} \nabla \log p_t(x) d_t + \sqrt{\frac{d\sigma_t^2}{dt}}\, d\bar{w} \ .$$

[12] From Y. Song et al., 2021.

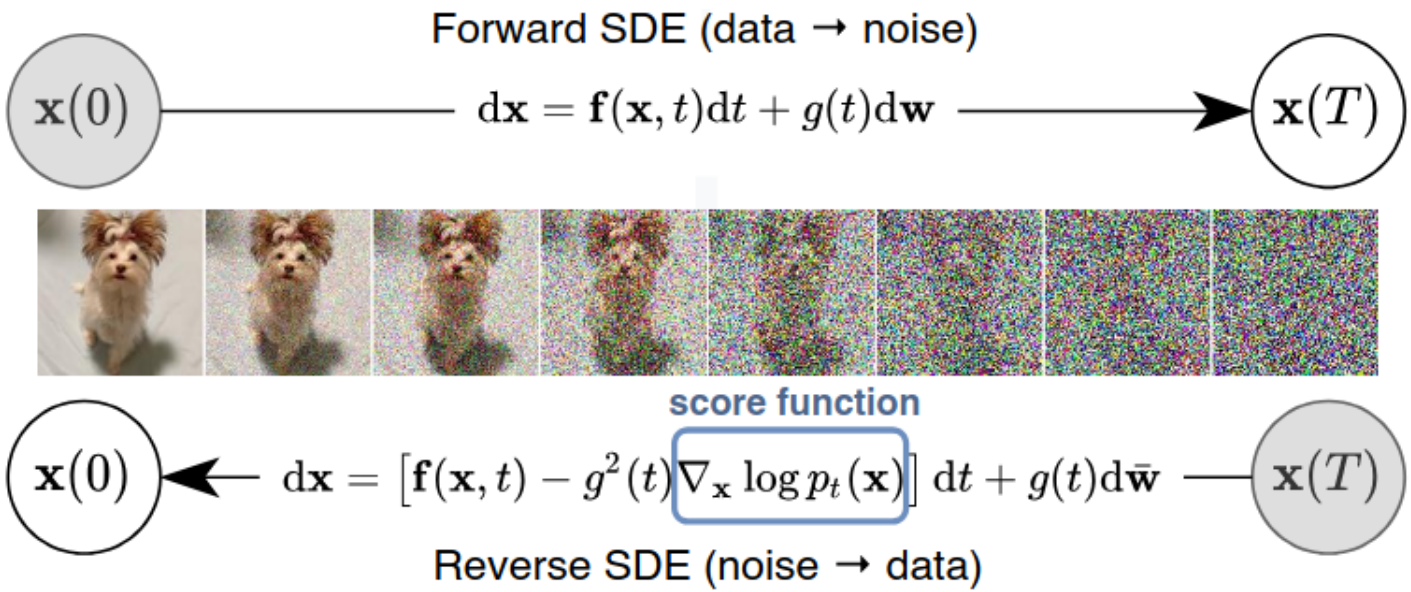


Figure: Illustration of noising and denoising with SDEs.

Our p_t is the time t marginal of the SDE

$$\mathrm{d}x = \sqrt{\frac{d\sigma_t^2}{dt}}dw \ .$$

It admits the backward decomposition

$$\mathrm{d}x = -\frac{d\sigma_t^2}{dt}\nabla \log p_t(x)d_t + \sqrt{\frac{d\sigma_t^2}{dt}}d\bar{w} \ .$$

[12] From Y. Song et al., 2021.

$\nabla \log p_t(x)$ and denoising

$$\begin{aligned}
\nabla_{x_t} \log p_t(x_t) &= \frac{\nabla_{x_t} p_t(x_t)}{p_t(x_t)} = \frac{\int \nabla_{x_t} p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\
&= \frac{\int \nabla \log p_{t|0}(x_t|x_0) p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\
&= \int \nabla \log p_{t|0}(x_t|x_0) \frac{p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\
&= \int \nabla \log p_{t|0}(x_t|x_0) p_{0|t}(\mathrm{d}x_0|x_t) = \mathbb{E}\left[\nabla \log p_{t|0}(x_t|X_0)|X_t = t\right] .
\end{aligned}$$

$\nabla \log p_{t|0}(x_t|x_0) = -\sigma_t^{-2}(x_t - x_0)$, thus

$$\nabla_{x_t} \log p_t(x_t) = \mathbb{E}\left[\nabla \log p_{t|0}(x_t|X_0)|X_t = t\right] = -\sigma_t^{-2}(x_t - \mathbb{E}\left[X_0|X_t = x_t\right])$$

or equivalently:

$$\mathbb{E}\left[X_0|X_t = x_t\right] = x_t + \sigma_t^2 \nabla \log p_t(x_t) .$$

$\nabla \log p_t(x)$ and denoising

$$\begin{aligned}\nabla_{x_t} \log p_t(x_t) &= \frac{\nabla_{x_t} p_t(x_t)}{p_t(x_t)} = \frac{\int \nabla_{x_t} p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\ &= \frac{\int \nabla \log p_{t|0}(x_t|x_0) p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\ &= \int \nabla \log p_{t|0}(x_t|x_0) \frac{p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\ &= \int \nabla \log p_{t|0}(x_t|x_0) p_{0|t}(\mathrm{d}x_0|x_t) = \mathbb{E}\left[\nabla \log p_{t|0}(x_t|X_0)|X_t = t\right] .\end{aligned}$$

$\nabla \log p_{t|0}(x_t|x_0) = -\sigma_t^{-2}(x_t - x_0)$, thus

$$\nabla_{x_t} \log p_t(x_t) = \mathbb{E}\left[\nabla \log p_{t|0}(x_t|X_0)|X_t = t\right] = -\sigma_t^{-2}(x_t - \mathbb{E}\left[X_0|X_t = x_t\right])$$

or equivalently:

$$\mathbb{E}\left[X_0|X_t = x_t\right] = x_t + \sigma_t^2 \nabla \log p_t(x_t) .$$

$\nabla \log p_t(x)$ and denoising

$$\nabla_{x_t} \log p_t(x_t) = \frac{\nabla_{x_t} p_t(x_t)}{p_t(x_t)} = \frac{\int \nabla_{x_t} p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)}$$
$$= \frac{\int \nabla \log p_{t|0}(x_t|x_0) p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)}$$
$$= \int \nabla \log p_{t|0}(x_t|x_0) \frac{p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)}$$
$$= \int \nabla \log p_{t|0}(x_t|x_0) p_{0|t}(\mathrm{d}x_0|x_t) = \mathbb{E}\left[\nabla \log p_{t|0}(x_t|X_0)|X_t = t\right] .$$

$\nabla \log p_{t|0}(x_t|x_0) = -\sigma_t^{-2}(x_t - x_0)$, thus

$$\nabla_{x_t} \log p_t(x_t) = \mathbb{E}\left[\nabla \log p_{t|0}(x_t|X_0)|X_t = t\right] = -\sigma_t^{-2}(x_t - \mathbb{E}\left[X_0|X_t = x_t\right])$$

or equivalently:

$$\mathbb{E}\left[X_0|X_t = x_t\right] = x_t + \sigma_t^2 \nabla \log p_t(x_t) .$$

$\nabla \log p_t(x)$ and denoising

$$\begin{aligned}
\nabla_{x_t} \log p_t(x_t) &= \frac{\nabla_{x_t} p_t(x_t)}{p_t(x_t)} = \frac{\int \nabla_{x_t} p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\
&= \frac{\int \nabla \log p_{t|0}(x_t|x_0) p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\
&= \int \nabla \log p_{t|0}(x_t|x_0) \frac{p_{t|0}(x_t|x_0)\pi(\mathrm{d}x_0)}{p_t(x_t)} \\
&= \int \nabla \log p_{t|0}(x_t|x_0) p_{0|t}(\mathrm{d}x_0|x_t) = \mathbb{E}\left[\nabla \log p_{t|0}(x_t|X_0)|X_t = t\right] .
\end{aligned}$$

$\nabla \log p_{t|0}(x_t|x_0) = -\sigma_t^{-2}(x_t - x_0)$, thus

$$\nabla_{x_t} \log p_t(x_t) = \mathbb{E}\left[\nabla \log p_{t|0}(x_t|X_0)|X_t = t\right] = -\sigma_t^{-2}(x_t - \mathbb{E}\left[X_0|X_t = x_t\right])$$

or equivalently:

$$\mathbb{E}\left[X_0|X_t = x_t\right] = x_t + \sigma_t^2 \nabla \log p_t(x_t) .$$

Denoising and score

$$\mathbb{E}\left[X_0 | X_t = x_t\right] = x_t + \sigma_t^2 \nabla \log p_t(x_t) \; .$$

But $\mathbb{E}\left[\|X_0 - \mathbb{E}\left[X_0|X_t\right]\|^2\right] = \min_{g \text{ measurable}} \mathbb{E}\left[\|X_0 - g(X_t)\|^2\right]$.

Learning the score from data $\Leftrightarrow$ Learn a denoiser from data.

$$L_{sm}(\theta) = \mathbb{E}_{t \sim \text{Unif}([0,T])}\left[\gamma_t^2 \mathbb{E}_{X_0 \sim \pi, Z \sim \mathcal{N}(0,\text{I})}\left[\|X_0 - D_\theta(X_0 + \sigma_t Z, \sigma_t)\|^2\right]\right] \; .$$

Denoising and score

$$\mathbb{E}\left[X_0 | X_t = x_t\right] = x_t + \sigma_t^2 \nabla \log p_t(x_t) \ .$$

But $\mathbb{E}\left[\|X_0 - \mathbb{E}\left[X_0|X_t\right]\|^2\right] = \min_{g \text{ measurable}} \mathbb{E}\left[\|X_0 - g(X_t)\|^2\right]$.

Learning the score from data $\Leftrightarrow$ Learn a denoiser from data.

$$L_{sm}(\theta) = \mathbb{E}_{t\sim \text{Unif}([0,T])}\left[\gamma_t^2 \mathbb{E}_{X_0\sim\pi, Z\sim\mathcal{N}(0,\text{I})}\left[\|X_0 - D_\theta(X_0 + \sigma_t Z, \sigma_t)\|^2\right]\right] \ .$$

Denoising and score

$$\mathbb{E}\left[X_0 | X_t = x_t\right] = x_t + \sigma_t^2 \nabla \log p_t(x_t) \ .$$

But $\mathbb{E}\left[\|X_0 - \mathbb{E}\left[X_0|X_t\right]\|^2\right] = \min_{g \text{ measurable}} \mathbb{E}\left[\|X_0 - g(X_t)\|^2\right]$.

Learning the score from data $\Leftrightarrow$ Learn a denoiser from data.

$$L_{sm}(\theta) = \mathbb{E}_{t\sim \text{Unif}([0,T])}\left[\gamma_t^2 \mathbb{E}_{X_0\sim\pi, Z\sim\mathcal{N}(0,\text{I})}\left[\|X_0 - D_\theta(X_0 + \sigma_t Z, \sigma_t)\|^2\right]\right] \ .$$

Denoising and score

$$\mathbb{E}\left[X_0 | X_t = x_t\right] = x_t + \sigma_t^2 \nabla \log p_t(x_t) \ .$$

But $\mathbb{E}\left[\|X_0 - \mathbb{E}\left[X_0|X_t\right]\|^2\right] = \min_{g \text{ measurable}} \mathbb{E}\left[\|X_0 - g(X_t)\|^2\right]$.

Learning the score from data $\Leftrightarrow$ Learn a denoiser from data.

$$L_{sm}(\theta) = \mathbb{E}_{t \sim \text{Unif}([0,T])}\left[\gamma_t^2 \mathbb{E}_{X_0 \sim \pi, Z \sim \mathcal{N}(0,\text{I})}\left[\|X_0 - D_\theta(X_0 + \sigma_t Z, \sigma_t)\|^2\right]\right] \ .$$

Backward simulation

Suppose we have a trained denoiser $D_\theta(x, \sigma_t)$.
We now have to simulate from

$$\mathrm{d}x = \frac{d\sigma_t^2}{dt}\sigma_t^{-2}(x - D_\theta(x, \sigma_t))d_t + \sqrt{\frac{d\sigma_t^2}{dt}}d\bar{w} \ .$$

We should in theory start from p_T.

But $\mathrm{D_{KL}}\left(p_T || \mathcal{N}(0, \sigma_T^2 \mathrm{I})\right) \leq \sigma_T^{-2}\mathbb{E}\left[\|X_0\|^2\right]$.

With an appropriate choice of σ_T, we can start from $\mathcal{N}(0, \sigma_T^2 \mathrm{I})$.

Backward simulation

Suppose we have a trained denoiser $D_\theta(x, \sigma_t)$.
We now have to simulate from

$$\mathrm{d}x = \frac{d\sigma_t^2}{dt}\sigma_t^{-2}(x - D_\theta(x, \sigma_t))d_t + \sqrt{\frac{d\sigma_t^2}{dt}}d\bar{w} \ .$$

We should in theory start from p_T.

But $\mathrm{D_{KL}}\left(p_T || \mathcal{N}(0, \sigma_T^2 \mathrm{I})\right) \leq \sigma_T^{-2}\mathbb{E}\left[\|X_0\|^2\right]$.

With an appropriate choice of σ_T, we can start from $\mathcal{N}(0, \sigma_T^2 \mathrm{I})$.

Backward simulation

Suppose we have a trained denoiser $D_\theta(x, \sigma_t)$.
We now have to simulate from

$$\mathrm{d}x = \frac{d\sigma_t^2}{dt}\sigma_t^{-2}(x - D_\theta(x, \sigma_t))d_t + \sqrt{\frac{d\sigma_t^2}{dt}}d\bar{w} \ .$$

We should in theory start from p_T.

But $\mathrm{D_{KL}}\left(p_T || \mathcal{N}(0, \sigma_T^2 \mathrm{I})\right) \leq \sigma_T^{-2}\mathbb{E}\left[\|X_0\|^2\right]$.

With an appropriate choice of σ_T, we can start from $\mathcal{N}(0, \sigma_T^2 \mathrm{I})$.

Backward simulation

Suppose we have a trained denoiser $D_\theta(x, \sigma_t)$.
We now have to simulate from

$$\mathrm{d}x = \frac{d\sigma_t^2}{dt}\sigma_t^{-2}(x - D_\theta(x, \sigma_t))d_t + \sqrt{\frac{d\sigma_t^2}{dt}}d\bar{w} \ .$$

We should in theory start from p_T.

But $\mathrm{D_{KL}}\left(p_T || \mathcal{N}(0, \sigma_T^2 \mathrm{I})\right) \leq \sigma_T^{-2}\mathbb{E}\left[\|X_0\|^2\right]$.

With an appropriate choice of σ_T, we can start from $\mathcal{N}(0, \sigma_T^2 \mathrm{I})$.

Backward simulation; Naive discretization

We have to simulate from

$$\mathrm{d}x = \frac{d\sigma_t^2}{dt}\sigma_t^{-2}(x - D_\theta(x, \sigma_t))d_t + \sqrt{\frac{d\sigma_t^2}{dt}}d\bar{w} \ .$$

We consider the rather crude approximation

$$X_{t-1} = X_t + \frac{(\sigma_{t-1}^2 - \sigma_t^2)}{\sigma_t^2}(X_t - D_\theta(X_t, \sigma_t)) + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}Z_t$$

Backward simulation; Naive discretization

We have to simulate from

$$\mathrm{d}x = \frac{d\sigma_t^2}{dt}\sigma_t^{-2}(x - D_\theta(x, \sigma_t))d_t + \sqrt{\frac{d\sigma_t^2}{dt}}d\bar{w} \ .$$

We consider the rather crude approximation

$$X_{t-1} = X_t + \frac{(\sigma_{t-1}^2 - \sigma_t^2)}{\sigma_t^2}(X_t - D_\theta(X_t, \sigma_t)) + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}Z_t$$

Backward simulation; Naive discretization

We have to simulate from

$$\mathrm{d}x = \frac{d\sigma_t^2}{dt}\sigma_t^{-2}(x - D_\theta(x, \sigma_t))d_t + \sqrt{\frac{d\sigma_t^2}{dt}}d\bar{w} \ .$$

We consider the rather crude approximation

$$X_{t-1} = X_t + \frac{(\sigma_{t-1}^2 - \sigma_t^2)}{\sigma_t^2}(X_t - D_\theta(X_t, \sigma_t)) + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}Z_t$$

Algorithm 3 Backward sampling

procedure SAMPLING
Input: N.
Output: X_0.
  $X_N \sim \mathcal{N}(0, \mathrm{I})$.
  for $i = N-1, \cdots, 0$ **do**
    $t_i = (i/N)T$
    $Z_i \sim \mathcal{N}(0, \mathrm{I})$.
    $X_i = X_{i+1} + \frac{(\sigma^2_{t_i} - \sigma^2_{t_{i+1}})}{\sigma^2_{t_{i+1}}}(X_{i+1} - D_\theta(X_{i+1}, \sigma_{t_{i+1}})) + \sqrt{\sigma^2_{t_{i+1}} - \sigma^2_{t_i}} Z_i$.
  end for
  Return: X_0
end procedure

This is a very naive sampler. There are much better samplers, see Karras, Aittala, et al., 2022 or J. Song, Meng, and Ermon, 2021.

Illustration

Figure: Illustration on the Celeb dataset from Y. Song et al., 2021.

1. SOTA in most image tasks ✓
2. No mode collapse ✓
3. No adversarial training ✓
4. Easier to condition on text (for example) ✓
5. Easy to conditiong à-posteriori ✓
6. Sampling requires more Neural function evaluations (NFE) x

If you want to learn more about Normalizing flows

- George Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". In: *Journal of Machine Learning Research* 22.57 (2021), pp. 1–64
- Durk P Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. (Visited on 03/03/2025)
- Aaron van den Oord et al. "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016. (Visited on 03/03/2025)
- Louis Grenioux et al. "On Sampling with Approximate Transport Maps". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, July 2023, pp. 11698–11733

If you want to learn more about Normalizing flows

- Yufeng Zhang et al. "Kullback-Leibler Divergence-Based out-of-Distribution Detection with Flow-Based Generative Models". In: *IEEE Transactions on Knowledge and Data Engineering* 36.4 (Apr. 2024), pp. 1683–1697. ISSN: 1558-2191. DOI: 10.1109/TKDE.2023.3309853. (Visited on 02/01/2025)
- Tim Salimans et al. "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications". In: *International Conference on Learning Representations*. Feb. 2017. (Visited on 03/03/2025)
- Ruizhi Deng et al. "Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 7805–7815. (Visited on 03/03/2025)
- Shuangfei Zhai et al. *Normalizing Flows Are Capable Generative Models*. Dec. 2024. DOI: 10.48550/arXiv.2412.06329. arXiv: 2412.06329 [cs]. (Visited on 03/03/2025)

If you want to learn more about VAE

- Diederik P Kingma, Max Welling, et al. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392
- Rong Tang and Yun Yang. "On Empirical Bayes Variational Autoencoder: An Excess Risk Bound". In: *Proceedings of Thirty Fourth Conference on Learning Theory*. PMLR, July 2021, pp. 4068–4125. (Visited on 10/23/2024)
- Aaron Van Den Oord and Oriol Vinyals. "Neural Discrete Representation Learning". In: *Advances in Neural Information Processing Systems* 30 (2017). (Visited on 03/03/2025)
- Irina Higgins et al. "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. Feb. 2017. (Visited on 03/03/2025)

If you want to learn more about VAE

- Jakub Tomczak and Max Welling. "VAE with a VampPrior". In: *Proceedings of the Twenty-first International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 1214–1223. (Visited on 03/03/2025)
- Arash Vahdat and Jan Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 19667–19679. (Visited on 03/03/2025)
- Ali Razavi, Aaron van den Oord, and Oriol Vinyals. "Generating Diverse High-Fidelity Images with VQ-VAE-2". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. (Visited on 03/03/2025)

If you want to know more about GANs

- Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 5769–5779. ISBN: 978-1-5108-6096-4
- Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410. (Visited on 03/02/2025)
- Tero Karras, Samuli Laine, Miika Aittala, et al. *Analyzing and Improving the Image Quality of StyleGAN*. Mar. 2020. DOI: 10.48550/arXiv.1912.04958. arXiv: 1912.04958 [cs]. (Visited on 12/20/2024)
- Cem Anil, James Lucas, and Roger Grosse. *Sorting out Lipschitz Function Approximation*. June 2019. DOI: 10.48550/arXiv.1811.05381. arXiv: 1811.05381 [cs]. (Visited on 12/20/2024)

- Gérard Biau, Maxime Sangnier, and Ugo Tanielian. "Some Theoretical Insights into Wasserstein GANs". In: *Journal of Machine Learning Research* 22.119 (2021), pp. 1–45. ISSN: 1533-7928. (Visited on 11/26/2024)
- Elen Vardanyan et al. "Statistically Optimal Generative Modeling with Maximum Deviation from the Empirical Distribution". In: *Proceedings of the 41st International Conference on Machine Learning*. PMLR, July 2024, pp. 49203–49225. (Visited on 03/03/2025)

If you want to know more about Score based generative models

- Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 6840–6851
- Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models". In: *International Conference on Learning Representations*. 2021
- Prafulla Dhariwal and Alexander Nichol. "Diffusion Models Beat Gans on Image Synthesis". In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794
- Tero Karras, Miika Aittala, et al. "Elucidating the Design Space of Diffusion-Based Generative Models". In: *Proc. NeurIPS*. 2022

If you want to know more about Score based generative models

- Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695. (Visited on 01/28/2025)
- Zahra Kadkhodaie et al. "Generalization in Diffusion Models Arises from Geometry-Adaptive Harmonic Representation". In: *The Twelfth International Conference on Learning Representations*. 2023
- Giovanni Conforti, Alain Durmus, and Marta Gentiloni Silveri. *KL Convergence Guarantees for Score Diffusion Models under Minimal Data Assumptions*. Sept. 2024. arXiv: 2308.12240 [math, stat]. (Visited on 09/17/2024)
- Xingchao Liu, Chengyue Gong, and Qiang Liu. "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow". In: *The Eleventh International Conference on Learning Representations*. Sept. 2022. (Visited on 03/03/2025)

References I

[1] Cem Anil, James Lucas, and Roger Grosse. *Sorting out Lipschitz Function Approximation*. June 2019. DOI: 10.48550/arXiv.1811.05381. arXiv: 1811.05381 [cs]. (Visited on 12/20/2024).

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *International Conference on Machine Learning*. PMLR, 2017, pp. 214–223.

[3] Gérard Biau, Maxime Sangnier, and Ugo Tanielian. "Some Theoretical Insights into Wasserstein GANs". In: *Journal of Machine Learning Research* 22.119 (2021), pp. 1–45. ISSN: 1533-7928. (Visited on 11/26/2024).

[4] Giovanni Conforti, Alain Durmus, and Marta Gentiloni Silveri. *KL Convergence Guarantees for Score Diffusion Models under Minimal Data Assumptions*. Sept. 2024. arXiv: 2308.12240 [math, stat]. (Visited on 09/17/2024).

References II

[5] Ruizhi Deng et al. "Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 7805–7815. (Visited on 03/03/2025).

[6] Prafulla Dhariwal and Alexander Nichol. "Diffusion Models Beat Gans on Image Synthesis". In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density Estimation Using Real NVP". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

[8] M. D. Donsker and S. R. S. Varadhan. "Asymptotic Evaluation of Certain Markov Process Expectations for Large Time. IV". In: *Communications on Pure and Applied Mathematics* 36.2 (1983), pp. 183–212. ISSN: 1097-0312. DOI: 10.1002/cpa.3160360204. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160360204 (visited on 01/03/2025).

References III

[9] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680.

[10] Louis Grenioux et al. "On Sampling with Approximate Transport Maps". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, July 2023, pp. 11698–11733.

[11] Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 5769–5779. ISBN: 978-1-5108-6096-4.

References IV

[12] Irina Higgins et al. "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. Feb. 2017. (Visited on 03/03/2025).

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 6840–6851.

[14] Zahra Kadkhodaie et al. "Generalization in Diffusion Models Arises from Geometry-Adaptive Harmonic Representation". In: *The Twelfth International Conference on Learning Representations*. 2023.

[15] Tero Karras, Miika Aittala, et al. "Elucidating the Design Space of Diffusion-Based Generative Models". In: *Proc. NeurIPS*. 2022.

References V

[16] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410. (Visited on 03/02/2025).

[17] Tero Karras, Samuli Laine, Miika Aittala, et al. *Analyzing and Improving the Image Quality of StyleGAN*. Mar. 2020. DOI: 10.48550/arXiv.1912.04958. arXiv: 1912.04958 [cs]. (Visited on 12/20/2024).

[18] Diederik P Kingma, Max Welling, et al. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[19] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: (Dec. 23, 2013). URL: https://openreview.net/forum?id=33X9fd2-9FyZd (visited on 01/03/2025).

References VI

[20] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. (Visited on 03/03/2025).

[21] Xingchao Liu, Chengyue Gong, and Qiang Liu. "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow". In: *The Eleventh International Conference on Learning Representations*. Sept. 2022. (Visited on 03/03/2025).

[22] George Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". In: *Journal of Machine Learning Research* 22.57 (2021), pp. 1–64.

[23] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. "Generating Diverse High-Fidelity Images with VQ-VAE-2". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. (Visited on 03/03/2025).

References VII

[24] Danilo Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1530–1538.

[25] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695. (Visited on 01/28/2025).

[26] Tim Salimans et al. "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications". In: *International Conference on Learning Representations*. Feb. 2017. (Visited on 03/03/2025).

[27] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models". In: *International Conference on Learning Representations*. 2021.

References VIII

[28] Yang Song et al. "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations*. 2021.

[29] Rong Tang and Yun Yang. "On Empirical Bayes Variational Autoencoder: An Excess Risk Bound". In: *Proceedings of Thirty Fourth Conference on Learning Theory*. PMLR, July 2021, pp. 4068–4125. (Visited on 10/23/2024).

[30] Jakub Tomczak and Max Welling. "VAE with a VampPrior". In: *Proceedings of the Twenty-first International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 1214–1223. (Visited on 03/03/2025).

[31] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep Image Prior". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

References IX

[32] Arash Vahdat and Jan Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 19667–19679. (Visited on 03/03/2025).

[33] Aaron Van Den Oord and Oriol Vinyals. "Neural Discrete Representation Learning". In: *Advances in Neural Information Processing Systems* 30 (2017). (Visited on 03/03/2025).

[34] Aaron van den Oord et al. "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016. (Visited on 03/03/2025).

[35] Elen Vardanyan et al. "Statistically Optimal Generative Modeling with Maximum Deviation from the Empirical Distribution". In: *Proceedings of the 41st International Conference on Machine Learning*. PMLR, July 2024, pp. 49203–49225. (Visited on 03/03/2025).

[36] Cédric Villani. *Topics in optimal transportation*. Vol. 58. American Mathematical Soc., 2021.

[37] Nick Whiteley, Annie Gray, and Patrick Rubin-Delanchy. *Statistical Exploration of the Manifold Hypothesis*. Feb. 2024. DOI: 10.48550/arXiv.2208.11665. arXiv: 2208.11665 [stat]. (Visited on 02/28/2025).

[38] Shuangfei Zhai et al. *Normalizing Flows Are Capable Generative Models*. Dec. 2024. DOI: 10.48550/arXiv.2412.06329. arXiv: 2412.06329 [cs]. (Visited on 03/03/2025).

[39] Yufeng Zhang et al. "Kullback-Leibler Divergence-Based out-of-Distribution Detection with Flow-Based Generative Models". In: *IEEE Transactions on Knowledge and Data Engineering* 36.4 (Apr. 2024), pp. 1683–1697. ISSN: 1558-2191. DOI: 10.1109/TKDE.2023.3309853. (Visited on 02/01/2025).

Algorithm 4 Backward sampling

procedure SAMPLING
Input: N.
Output: X_0.
  $X_N \sim \mathcal{N}(0, \mathrm{I})$.
  for $i = N-1, \cdots, 0$ **do**
    $t_i = (i/N)T$
    $Z_i \sim \mathcal{N}(0, \mathrm{I})$.
    $X_i = X_{i+1} + \frac{\sigma_{t_i}^2 - \sigma_{t_{i+1}}^2}{\sigma_{t_{i+1}}^2}(X_{i+1} - D_\theta(X_{i+1}, \sigma_{t_{i+1}})) + \sqrt{\frac{\sigma_{t_i}^2}{\sigma_{t_{i+1}}^2}\left(\sigma_{t_{i+1}}^2 - \sigma_{t_i}^2\right)} Z_i$.
  end for
  Return: X_0
end procedure