Ingeniería Matemática
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

Modelos Generativos

MA5606: Tópicos Matemáticos en Aprendizaje de Máquinas, Redes Neuronales y Aprendizaje Profundo

Joaquín Fontbona, Claudio Muñoz, Diego Olguín, Álvaro Márquez y Javier Maass

Departamento de Ingeniería Matemática
Universidad de Chile

12 de junio

Contenidos

1. Modelos Generativos

2. Generative Adversarial Networks (GANs)

3. Denoising Diffusion Models (DDM)

Modelos Generativos

DIM Universidad de Chile

Generemos motivación

Digamos que tenemos datos que vienen dados por una distribución π (e.g. fotos de perros y gatos, todos los textos de literatura latinoamericana existentes, todas las películas de StarWars, una gaussiana, etc.).

vs

Generemos motivación

Digamos que tenemos datos que vienen dados por una distribución π (e.g. fotos de perros y gatos, todos los textos de literatura latinoamericana existentes, todas las películas de StarWars, una gaussiana, etc.).

vs

¿Y si quisiéramos **generar un nuevo dato de forma natural**? i.e. ¿si quisiéramos *samplear* de nuestra distribución π?

Generemos motivación

Digamos que tenemos datos que vienen dados por una distribución π (e.g. fotos de perros y gatos, todos los textos de literatura latinoamericana existentes, todas las películas de StarWars, una gaussiana, etc.).

vs

¿Y si quisiéramos **generar un nuevo dato de forma natural**? i.e. ¿si quisiéramos *samplear* de nuestra distribución π?

Aparecen los **Modelos Generativos** al rescate!

Generemos motivación

Pero, **¿por qué querríamos *samplear* datos desde π?**

- La tecnología nos ayuda a automatizar tareas que nunca imaginamos! ¿Y si quisiéramos automatizar el *arte*?

Generemos motivación

Pero, **¿por qué querríamos *samplear* datos desde π?**

- La tecnología nos ayuda a automatizar tareas que nunca imaginamos! ¿Y si quisiéramos automatizar el *arte*?
- ¿Y si quisiéramos generar fotos de perros?

Generemos motivación

Pero, **¿por qué querríamos *samplear* datos desde π?**

- La tecnología nos ayuda a automatizar tareas que nunca imaginamos! ¿Y si quisiéramos automatizar el *arte*?
- ¿Y si quisiéramos generar fotos de perros?

Generemos motivación

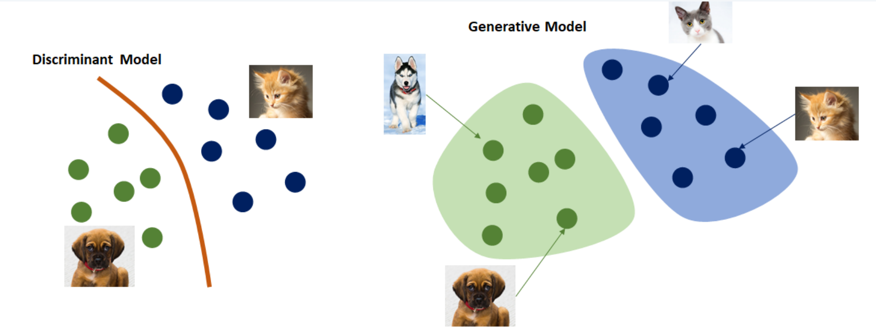
¿Y si pudiésemos hacer los sueños realidad?

Generemos motivación

¿Y si pudiésemos hacer los sueños realidad?

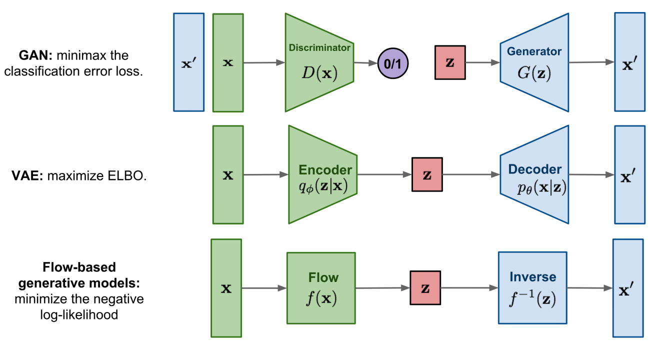
Generemos motivación

Más allá de los ejemplos ““graciosos””, la idea principal de los **Modelos Generativos** es intentar tener modelos que **entiendan muy bien una distribución de datos**, al punto que nos permitan incluso **samplear desde ella**.



Generemos motivación

Existen **varios tipos de Modelos Generativos**:



En este tutorial exploraremos principalmente las **GANs** y los **Modelos de Difusión**.

Generative Adversarial Networks (GANs)

DIM Universidad de Chile

Redes Generativas Adversarias (GANs)

Las slides de esta sección están fuertemente basadas en la sección de GANs del trabajo de tesis de Francisco Muñoz (Muñoz López (2024)).
Si bien me tocó adaptar el material, los créditos de la autoría y las fuentes recabadas son para él.

Introducción a GANs

- Las Redes Generativas Adversarias (**GANs**), proponen un marco de entrenamiento donde dos redes neuronales compiten entre sí.

Introducción a GANs

- Las Redes Generativas Adversarias (**GANs**), proponen un marco de entrenamiento donde dos redes neuronales compiten entre sí.
- Se basan en un juego min-max entre dos redes:

Introducción a GANs

- Las Redes Generativas Adversarias (**GANs**), proponen un marco de entrenamiento donde dos redes neuronales compiten entre sí.
- Se basan en un juego min-max entre dos redes:
  - **Generador (G)**: Aprende a generar datos similares a los reales.

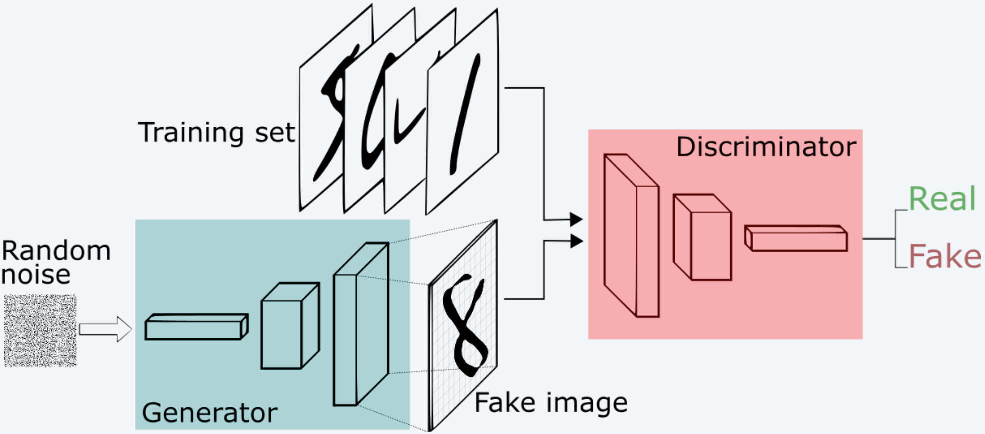
Introducción a GANs

- Las Redes Generativas Adversarias (**GANs**), proponen un marco de entrenamiento donde dos redes neuronales compiten entre sí.
- Se basan en un juego min-max entre dos redes:
  - **Generador (G)**: Aprende a generar datos similares a los reales.
  - **Discriminador (D)**: Aprende a distinguir entre datos reales y generados.

Introducción a GANs

- Las Redes Generativas Adversarias (**GANs**), proponen un marco de entrenamiento donde dos redes neuronales compiten entre sí.
- Se basan en un juego min-max entre dos redes:
  - **Generador (G)**: Aprende a generar datos similares a los reales.
  - **Discriminador (D)**: Aprende a distinguir entre datos reales y generados.
- Esta arquitectura fue propuesta inicialmente por Goodfellow et al. (2014); y ha sido objeto de estudio extensivo desde entonces (Arjovsky et al. (2017); Gulrajani et al. (2017); Heusel et al. (2017); Salimans et al. (2016); Zhu et al. (2017); Wang et al. (2018); Xiao et al. (2021); entre muchos otros).

Esquema de las GANs



Definición Formal de una GAN

Plantearemos la idea de las GANs en un contexto **no paramétrico**, basado en medidas de probabilidad generales, sin ahondar todavía en las NNs como tal.

Definición Formal de una GAN

Plantearemos la idea de las GANs en un contexto **no paramétrico**, basado en medidas de probabilidad generales, sin ahondar todavía en las NNs como tal.

Planteamiento de Teoría de Juegos

Dado $(\mathcal{X}, \pi)$ e.d.p., una GAN es un juego de **suma cero** con dos jugadores:

- Un **G**enerador μ_G, con un conjunto de estrategias $\mathcal{P}(\mathcal{X})$.
- Un **D**iscriminador $\mu_D(\cdot|x)$, con kernels Markovianos a $[0, 1]$ como estrategias.

La función valor objetivo del juego es:

$$V(\mu_G, \mu_D) = \mathbb{E}_{X\sim\pi}\mathbb{E}_{Y\sim\mu_D(X)}[\ln\ Y] + \mathbb{E}_{\tilde{X}\sim\mu_G}\mathbb{E}_{Y\sim\mu_D(\tilde{X})}[\ln(1-Y)]$$

La cual es **minimizada por G** y **maximizada por D**.

Definición Formal de una GAN

Teorema (Discriminador Óptimo)

Para un generador fijo μ_G, el discriminador óptimo (que ***existe y es único****) es:*

$$\mu_D^*(\cdot|x) = \delta_{D^*(x)}(\cdot), \quad donde \quad D^*(x) = \frac{d\pi}{d(\mu_G + \pi)}(x)$$

Es decir, es una función determinista $D^* : \mathcal{X} \to [0,1]$. En este caso, se tiene:

$$V(\mu_G, \mu_D^*) = JS(\pi, \mu_G) - 2\ln 2$$

con JS la ***divergencia de Jensen-Shannon*** *entre 2 medidas.*

Definición Formal de una GAN

Teorema (Discriminador Óptimo)

*Para un generador fijo μ_G, el discriminador óptimo (que **existe y es único**) es:*

$$\mu_D^*(\cdot|x) = \delta_{D^*(x)}(\cdot), \quad \textit{donde} \quad D^*(x) = \frac{d\pi}{d(\mu_G + \pi)}(x)$$

Es decir, es una función determinista $D^* : \mathcal{X} \to [0,1]$. En este caso, se tiene:

$$V(\mu_G, \mu_D^*) = JS(\pi, \mu_G) - 2\ln 2$$

*con JS la **divergencia de Jensen-Shannon** entre 2 medidas.*

Es decir, **el discriminador busca distinguir muestras verdaderas de falsas de la mejor forma posible**.

Definición Formal de una GAN

Esquema de demostración del Teorema del Discriminador Óptimo

Usando desigualdad de Jensen y lema de Doob:

1. Maximizar $V(\mu_G, \mu_D)$ sobre μ_D equivale a maximizar (**s.p.g**):

$$\max_{D:\mathcal{X}\to[0,1]} \mathbb{E}_{X\sim\pi}[\ln D(X)] + \mathbb{E}_{\tilde{X}\sim\mu_G}[\ln(1 - D(\tilde{X}))]$$

2. Como la función $y \mapsto a \ln y + b \ln(1-y)$ alcanza su máximo en $y^* = \frac{a}{a+b}$; la solución óptima se obtiene cuando:

$$D^*(x) = \frac{d\pi}{d(\pi + \mu_G)}(x)$$

3. Sustituyendo en V se obtiene la relación con $JS(\pi, \mu_G)$

Definición Formal de una GAN

Dado el teorema, podemos definir: $C(\mu_G) := V(\mu_G, \mu_D^*) = \mathrm{JS}(\mu, \mu_G) - 2\ln 2$

Definición Formal de una GAN

Dado el teorema, podemos definir: $C(\mu_G) := V(\mu_G, \mu_D^*) = \mathrm{JS}(\mu, \mu_G) - 2\ln 2$

Teorema (Generador Óptimo)

El mínimo global de la función $C(\mu_G)$ se alcanza en $\mu_G^* = \pi$, y el valor del mínimo es $C(\pi) = -\ln 4$. i.e. ***El mejor generador posible es uno que imita perfectamente π.***

Definición Formal de una GAN

Dado el teorema, podemos definir: $C(\mu_G) := V(\mu_G, \mu_D^*) = \text{JS}(\mu, \mu_G) - 2\ln 2$

Teorema (Generador Óptimo)

El mínimo global de la función $C(\mu_G)$ se alcanza en $\mu_G^* = \pi$, y el valor del mínimo es $C(\pi) = -\ln 4$. i.e. ***El mejor generador posible es uno que imita perfectamente*** π.

Demostración

- Partiendo del Teorema anterior:

$$\min_{\mu_G}\max_{\mu_D} V(\mu_G, \mu_D) = \min_{\mu_G} V(\mu_G, \mu_D^*) = \min_{\mu_G} C(\mu_G) = \min_{\mu_G} JS(\pi, \mu_G) - 2\ln 2$$

- Como la divergencia de Jensen-Shannon es estrictamente positiva si $\mu_G \neq \pi$, y nula si $\mu_G = \pi$, entonces el mínimo global de $C(\mu_G)$ se alcanza en $\mu_G^* = \pi$.
- El valor mínimo es $C(\pi) = -\ln 4$.

Implementación Práctica de las GANs

Planteamiento Funcional del Problema

- **El discriminador se puede buscar como una función** $D : \mathcal{X} \to [0, 1]$.
- Análogamente, en la práctica el **generador óptimo se plantea como una pushforward de una distribución conocida**: $\mu_G = G\#\mu_Z$, con $\mathcal{Z}$ un *espacio latente*, μ_Z una distribución conocida (e.g. *Gaussiana*), y $G : \mathcal{Z} \to \mathcal{X}$ medible.

Implementación Práctica de las GANs

Planteamiento Funcional del Problema

- **El discriminador se puede buscar como una función** $D : \mathcal{X} \to [0, 1]$.
- Análogamente, en la práctica el **generador óptimo se plantea como una pushforward de una distribución conocida**: $\mu_G = G\#\mu_Z$, con $\mathcal{Z}$ un *espacio latente*, μ_Z una distribución conocida (e.g. *Gaussiana*), y $G : \mathcal{Z} \to \mathcal{X}$ medible.

En la práctica, aproximamos esto con dos redes neuronales: una con pesos θ_g que llamaremos **Generador** y otra con pesos θ_d que llamaremos **Discriminador**.

Implementación Práctica de las GANs

Planteamiento Funcional del Problema

- **El discriminador se puede buscar como una función** $D : \mathcal{X} \to [0,1]$.
- Análogamente, en la práctica el **generador óptimo se plantea como una pushforward de una distribución conocida**: $\mu_G = G\#\mu_Z$, con $\mathcal{Z}$ un *espacio latente*, μ_Z una distribución conocida (e.g. *Gaussiana*), y $G : \mathcal{Z} \to \mathcal{X}$ medible.

En la práctica, aproximamos esto con dos redes neuronales: una con pesos θ_g que llamaremos **Generador** y otra con pesos θ_d que llamaremos **Discriminador**.

- **Generador (G)**: Define una función $G(z; \theta_g)$ que mapea ruido $z \sim \mu_Z$ a $\mathcal{X}$.

Implementación Práctica de las GANs

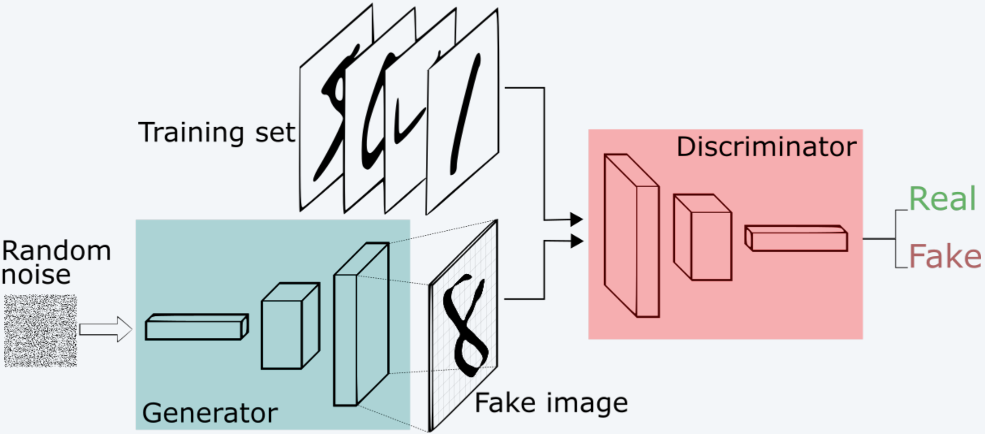
Planteamiento Funcional del Problema

- **El discriminador se puede buscar como una función** $D : \mathcal{X} \to [0, 1]$.
- Análogamente, en la práctica el **generador óptimo se plantea como una pushforward de una distribución conocida**: $\mu_G = G\#\mu_Z$, con $\mathcal{Z}$ un *espacio latente*, μ_Z una distribución conocida (e.g. *Gaussiana*), y $G : \mathcal{Z} \to \mathcal{X}$ medible.

En la práctica, aproximamos esto con dos redes neuronales: una con pesos θ_g que llamaremos **Generador** y otra con pesos θ_d que llamaremos **Discriminador**.

- **Generador (G)**: Define una función $G(z; \theta_g)$ que mapea ruido $z \sim \mu_Z$ a $\mathcal{X}$.
- **Discriminador (D)**: Define $D(x; \theta_d)$, que da la probabilidad de que x provenga de π.

Esquema de las GANs



Entrenando a tu GAN

El entrenamiento de las GANs se realiza usualmente repitiendo K veces el ciclo de:
Entrenar el Discriminador $\rightarrow$ **Entrenar el Generador**, hasta converger.

Entrenando a tu GAN

El entrenamiento de las GANs se realiza usualmente repitiendo K veces el ciclo de:
Entrenar el Discriminador $\to$ **Entrenar el Generador**, hasta converger.

i.e. En cada iteración:

- Muestrear $\{z^{(i)}\}_{i=1}^m$ de la distribución de ruido μ_Z, $\{x^{(i)}\}_{i=1}^m$ de la distribución de datos π y actualizar el discriminador según SGD:

$$\nabla_{\theta_d}\left[\frac{1}{m}\sum_{i=1}^{m}\left[\log D_{\theta_d}\left(x^{(i)}\right)+\log\left(1-D_{\theta_d}\left(G_{\theta_g}\left(z^{(i)}\right)\right)\right)\right]\right].$$

Entrenando a tu GAN

El entrenamiento de las GANs se realiza usualmente repitiendo K veces el ciclo de: **Entrenar el Discriminador** $\rightarrow$ **Entrenar el Generador**, hasta converger.

i.e. En cada iteración:

- Muestrear $\{z^{(i)}\}_{i=1}^{m}$ de la distribución de ruido μ_Z, $\{x^{(i)}\}_{i=1}^{m}$ de la distribución de datos π y actualizar el discriminador según SGD:

$$\nabla_{\theta_d}\left[\frac{1}{m}\sum_{i=1}^{m}\left[\log D_{\theta_d}\left(x^{(i)}\right)+\log\left(1-D_{\theta_d}\left(G_{\theta_g}\left(z^{(i)}\right)\right)\right)\right]\right].$$

- Muestrear $\{z^{(i)}\}_{i=1}^{m}$ de la distribución de ruido $p_g(z)$ y actualizar según SGD:

$$\nabla_{\theta_g}\left[\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D_{\theta_d}\left(G_{\theta_g}\left(z^{(i)}\right)\right)\right)\right].$$

Denoising Diffusion Models (DDM)

DIM Universidad de Chile

Modelos de Difusión

Utilizaremos las slides del curso de Generative Modelling de Valentin DeBortoli (link)...

Modelos de Difusión

Utilizaremos las slides del curso de Generative Modelling de Valentin DeBortoli (link)... TODAS las slides que vienen a partir de este punto son simplemente una traducción casi verbatim de las slides hechas por Valentin, por lo que no me hago creer el autor de ellas bajo ninguna circunstancia.

Modelos de Difusión

Denoising Diffusion Models (DDM) como alternativa para **generar datos de** π.
También llamados **Score-Based Generative Models**(ver aquí para más referencias).

Modelos de Difusión

Denoising Diffusion Models (DDM) como alternativa para **generar datos de** π.
También llamados **Score-Based Generative Models**(ver aquí para más referencias).

¿Por qué elegir este tipo de métodos?:

- Resultados **SOTA** Dhariwal & Nichol (2021); Karras et al. (2022).
- **Altamente flexibles** Poole et al. (2022); Rombach et al. (2022); Balaji et al. (2022); Saharia et al. (2022) (aplicaciones en Text2Image, CLIP, entre muchas otras).
- **Garantías Teóricas** De Bortoli et al. (2021b); Chen et al. (2022); Pidstrigach (2022); Lee et al. (2022).

** Eso sí, la **comprensión estadística** de estos modelos todavía es limitada.

Modelos de Difusión

Denoising Diffusion Models (DDM) como alternativa para **generar datos de** π.
También llamados **Score-Based Generative Models**(ver aquí para más referencias).

¿Por qué elegir este tipo de métodos?:

- Resultados **SOTA** Dhariwal & Nichol (2021); Karras et al. (2022).
- **Altamente flexibles** Poole et al. (2022); Rombach et al. (2022); Balaji et al. (2022); Saharia et al. (2022) (aplicaciones en Text2Image, CLIP, entre muchas otras).
- **Garantías Teóricas** De Bortoli et al. (2021b); Chen et al. (2022); Pidstrigach (2022); Lee et al. (2022).

** Eso sí, la **comprensión estadística** de estos modelos todavía es limitada.

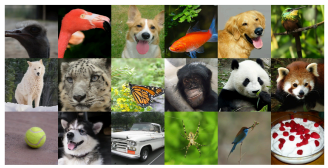
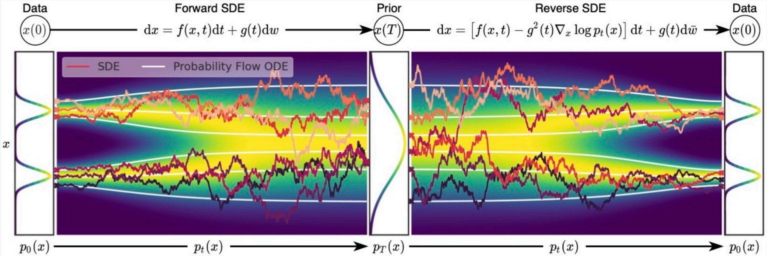


Figura: Imagen Generada por DDM Dhariwal & Nichol (2021).

- Primer artículo (enfoque variacional) Sohl-Dickstein et al. (2015).
- Primera aplicación exitosa Song & Ermon (2019).
- Concurrentemente (enfoque variacional) Ho et al. (2020).

Principios de DDM



Procesos de ruido y generativos en DDM. Imagen extraída de Song et al. (2020b).

- Buscamos **interpolar** entre dos distribuciones:
  - La distribución real de los datos, $\pi \in \mathcal{P}(\mathcal{X})$, y
  - La distribución *fácil de samplear*, $\mu_Z \in \mathcal{P}(\mathcal{X})$ (e.g. Gaussiana estándar).
- Es "**fácil**" ir de π a μ_Z, con un **proceso de *noising* (forward)**.
- Queremos **invertir** este proceso, para ir de μ_Z a π, y lograr así un **proceso generativo**.

Ancestral Sampling

- Sea $\mathcal{X} = \mathbb{R}^d$, $N \in \mathbb{N}$, $N > 0$ y p una densidad en $(\mathbb{R}^d)^{N+1}$ tal que para cualquier $x_{0:N} = \{x_k\}_{k=0}^N$, p admite una **descomposición forward** de la forma:

$$p(x_{0:N}) = p_0(x_0) \prod_{k=0}^{N-1} p_{k+1|k}(x_{k+1}|x_k)$$

Ancestral Sampling

- Sea $\mathcal{X} = \mathbb{R}^d$, $N \in \mathbb{N}$, $N > 0$ y p una densidad en $(\mathbb{R}^d)^{N+1}$ tal que para cualquier $x_{0:N} = \{x_k\}_{k=0}^{N}$, p admite una **descomposición forward** de la forma:

$$p(x_{0:N}) = p_0(x_0) \prod_{k=0}^{N-1} p_{k+1|k}(x_{k+1}|x_k)$$

- $\forall k \in \{0, \ldots, N-1\}$ definimos la **marginal** p_{k+1} para $x_{k+1} \in \mathbb{R}^d$

$$p_{k+1}(x_{k+1}) = \int_{\mathbb{R}^d} p_k(x_k) p_{k+1|k}(x_{k+1}|x_k) dx_k$$

Ancestral Sampling

- Sea $\mathcal{X} = \mathbb{R}^d$, $N \in \mathbb{N}$, $N > 0$ y p una densidad en $(\mathbb{R}^d)^{N+1}$ tal que para cualquier $x_{0:N} = \{x_k\}_{k=0}^N$, p admite una **descomposición forward** de la forma:

$$p(x_{0:N}) = p_0(x_0) \prod_{k=0}^{N-1} p_{k+1|k}(x_{k+1}|x_k)$$

- $\forall k \in \{0, ..., N-1\}$ definimos la **marginal** p_{k+1} para $x_{k+1} \in \mathbb{R}^d$

$$p_{k+1}(x_{k+1}) = \int_{\mathbb{R}^d} p_k(x_k) p_{k+1|k}(x_{k+1}|x_k) dx_k$$

- Asumimos que $\forall k \in \{0, ...N\}$, $p_k > 0$ y definimos $\forall x_k, x_{k+1} \in \mathbb{R}^d$:

$$p_{k|k+1}(x_k|x_{k+1}) = \frac{p_{k+1|k}(x_{k+1}|x_k) p_k(x_k)}{p_{k+1}(x_{k+1})}$$

Ancestral Sampling

- Sea $\mathcal{X} = \mathbb{R}^d$, $N \in \mathbb{N}$, $N > 0$ y p una densidad en $(\mathbb{R}^d)^{N+1}$ tal que para cualquier $x_{0:N} = \{x_k\}_{k=0}^N$, p admite una **descomposición forward** de la forma:

$$p(x_{0:N}) = p_0(x_0) \prod_{k=0}^{N-1} p_{k+1|k}(x_{k+1}|x_k)$$

- $\forall k \in \{0, ..., N-1\}$ definimos la **marginal** p_{k+1} para $x_{k+1} \in \mathbb{R}^d$

$$p_{k+1}(x_{k+1}) = \int_{\mathbb{R}^d} p_k(x_k) p_{k+1|k}(x_{k+1}|x_k) dx_k$$

- Asumimos que $\forall k \in \{0, ...N\}$, $p_k > 0$ y definimos $\forall x_k, x_{k+1} \in \mathbb{R}^d$:

$$p_{k|k+1}(x_k|x_{k+1}) = \frac{p_{k+1|k}(x_{k+1}|x_k) p_k(x_k)}{p_{k+1}(x_{k+1})}$$

- Podemos obtener la **descomposición backward**:

$$p(x_{0:N}) = p_N(x_N) \prod_{k=0}^{N-1} p_{k|k+1}(x_k|x_{k+1})$$

El Proceso Forward

- En la práctica consideramos:
  - π admite una densidad p_0 con respecto a la medida de Lebesgue.
  - La **descomposición forward** es un **proceso de ruido**.

$$p(x_{0:N}) = p_0(x_0) \prod_{k=0}^{N-1} p_{k+1|k}(x_{k+1}|x_k)$$

El Proceso Forward

- En la práctica consideramos:
  - π admite una densidad p_0 con respecto a la medida de Lebesgue.
  - La **descomposición forward** es un **proceso de ruido**.

$$p(x_{0:N}) = p_0(x_0) \prod_{k=0}^{N-1} p_{k+1|k}(x_{k+1}|x_k)$$

- ¿Cómo pasamos de la distribución de datos a la distribución fácil de muestrear?
  - **Proceso Autorregresivo**: Para $\{Z_k\}_{k\in\mathbb{N}} \sim N(0, Id)$ i.i.d. y $\alpha < 1$, hacemos:

$$X_{k+1} = \alpha X_k + \sqrt{1-\alpha^2} Z_{k+1}$$

  - $Law(X_k) \to \mathcal{N}(0, Id)$ exponencialmente rápido (en Wasserstein, TV) cuando $k \to \infty$.

El Proceso Forward

- En la práctica consideramos:
  - π admite una densidad p_0 con respecto a la medida de Lebesgue.
  - La **descomposición forward** es un **proceso de ruido**.

$$p(x_{0:N}) = p_0(x_0) \prod_{k=0}^{N-1} p_{k+1|k}(x_{k+1}|x_k)$$

- ¿Cómo pasamos de la distribución de datos a la distribución fácil de muestrear?
  - **Proceso Autorregresivo**: Para $\{Z_k\}_{k \in \mathbb{N}} \sim N(0, Id)$ i.i.d. y $\alpha < 1$, hacemos:

$$X_{k+1} = \alpha X_k + \sqrt{1-\alpha^2} Z_{k+1}$$

  - $Law(X_k) \to \mathcal{N}(0, Id)$ exponencialmente rápido (en Wasserstein, TV) cuando $k \to \infty$.
- Otra forma de verlo:
  - Proceso de **Ornstein-Uhlenbeck**: $dX_t = -X_t dt + \sqrt{2} dB_t$, y su discretización de **Euler-Maruyama**: $X_{k+1} = (1-\gamma)X_k + \sqrt{2\gamma} Z_{k+1}$; la cual converge **exponencialmente rápido** a $N(0, Id/(1-\gamma/2))$.

¿Cómo invertimos el proceso forward?

Podemos hacer unos cálculos medios densos, para obtener lo siguiente:

$$\begin{aligned}
p_{k|k+1}(x_k|x_{k+1}) &= p_{k+1|k}(x_{k+1}|x_k)p_k(x_k)/p_{k+1}(x_{k+1}) \\
&= C_0 \exp[-||x_{k+1} - (1-\gamma)x_k||^2/(4\gamma)] \exp[\log(p_k(x_k)) - \log(p_{k+1}(x_{k+1}))] \\
&= C_1 \exp[-||x_{k+1} - (1-\gamma)x_k||^2/(4\gamma)] \exp[\log(p_k(x_k)) - \log(p_k(x_{k+1}))] \\
&= C_1 \exp[-(||x_{k+1} - (1-\gamma)x_k||^2 + 4\gamma\{\log(p_k(x_k)) - \log(p_k(x_{k+1}))\})/(4\gamma)].
\end{aligned}$$

¿Cómo invertimos el proceso forward?

Podemos hacer unos cálculos medios densos, para obtener lo siguiente:

$$\begin{aligned} p_{k|k+1}(x_k|x_{k+1}) &= p_{k+1|k}(x_{k+1}|x_k)p_k(x_k)/p_{k+1}(x_{k+1}) \\ &= C_0 \exp[-||x_{k+1} - (1-\gamma)x_k||^2/(4\gamma)] \exp[\log(p_k(x_k)) - \log(p_{k+1}(x_{k+1}))] \\ &= C_1 \exp[-||x_{k+1} - (1-\gamma)x_k||^2/(4\gamma)] \exp[\log(p_k(x_k)) - \log(p_k(x_{k+1}))] \\ &= C_1 \exp[-(||x_{k+1} - (1-\gamma)x_k||^2 + 4\gamma\{\log(p_k(x_k)) - \log(p_k(x_{k+1}))\})/(4\gamma)]. \end{aligned}$$

Donde $C_0, C_1 > 0$ son constantes que dependen solo de x_{k+1}. Por otro lado:

- $||x_{k+1} - (1-\gamma)x_k||^2 = ||x_k - (1+\gamma)x_{k+1}||^2 - 2\gamma||x_{k+1} - x_k||^2 + \gamma^2\{||x_k||^2 - ||x_{k+1}||^2\}$.
- Y, por Taylor: $\log(p_k(x_k)) = \log(p_k(x_{k+1})) + \langle \nabla \log p_k(x_{k+1}), x_k - x_{k+1}\rangle + \int_0^1 \nabla^2 \log p_k((1-t)x_{k+1} + tx_k)(x_k - x_{k+1})^{\otimes 2} dt$.

¿Cómo invertimos el proceso forward?

- Si **asumimos** que: $||x_{k+1} - x_k||^2 \leq C\gamma$ y $\max(||x_k||, ||x_{k+1}||) \leq C$; entonces:
  - $\left| ||x_{k+1} - (1-\gamma)x_k||^2 - ||x_k - (1+\gamma)x_{k+1}||^2 \right| \leq 4C\gamma^2$.
  - $|\log(p_k(x_k)) - \log(p_{k+1}(x_{k+1})) - \langle \nabla \log p_k(x_{k+1}), x_k - x_{k+1} \rangle| \leq D\gamma$.

¿Cómo invertimos el proceso forward?

- Si **asumimos** que: $||x_{k+1} - x_k||^2 \leq C\gamma$ y $\text{máx}(||x_k||, ||x_{k+1}||) \leq C$; entonces:
  - $\left| ||x_{k+1} - (1-\gamma)x_k||^2 - ||x_k - (1+\gamma)x_{k+1}||^2 \right| \leq 4C\gamma^2$.
  - $|\log(p_k(x_k)) - \log(p_{k+1}(x_{k+1})) - \langle \nabla \log p_k(x_{k+1}), x_k - x_{k+1} \rangle| \leq D\gamma$.
- Con eso, **aproximando** hasta un término de orden γ en la exponencial, se tiene:

$$
\begin{aligned}
p_{k|k+1}(x_k|x_{k+1}) &\approx C_2 \exp\left[-||x_k - (1+\gamma)x_{k+1}||^2/(4\gamma) + \langle \nabla \log p_k(x_{k+1}), x_k - x_{k+1} \rangle \right] \\
&\approx N(x_k; x_{k+1} + \gamma\{x_{k+1} + 2\nabla \log p_k(x_{k+1})\}, 2\gamma Id)
\end{aligned}
$$

¿Cómo invertimos el proceso forward?

- Si **asumimos** que: $||x_{k+1} - x_k||^2 \leq C\gamma$ y $\text{máx}(||x_k||, ||x_{k+1}||) \leq C$; entonces:
  - $\left| ||x_{k+1} - (1-\gamma)x_k||^2 - ||x_k - (1+\gamma)x_{k+1}||^2 \right| \leq 4C\gamma^2$.
  - $|\log(p_k(x_k)) - \log(p_{k+1}(x_{k+1})) - \langle \nabla \log p_k(x_{k+1}), x_k - x_{k+1} \rangle| \leq D\gamma$.
- Con eso, **aproximando** hasta un término de orden γ en la exponencial, se tiene:

$$\begin{aligned} p_{k|k+1}(x_k|x_{k+1}) &\approx C_2 \exp\left[-||x_k - (1+\gamma)x_{k+1}||^2/(4\gamma) + \langle \nabla \log p_k(x_{k+1}), x_k - x_{k+1} \rangle \right] \\ &\approx N(x_k; x_{k+1} + \gamma\{x_{k+1} + 2\nabla \log p_k(x_{k+1})\}, 2\gamma Id) \end{aligned}$$

- Si bien la aproximación es media rancia, esto nos permite definir de forma simple el **proceso backward**, desde $X_N \sim \mu_Z$:

$$X_k = X_{k+1} + \gamma\{X_{k+1} + 2\nabla \log p_k(X_{k+1})\} + \sqrt{2\gamma} Z_{k+1}.$$

- Aquí el término $\nabla \log p_k$ es **intractable**, por lo que tendremos que aproximarlo...

Score-matching

Al término $\nabla \log p_k$ se le conoce como el **score (de Stein)**; y por eso hablamos de un problema de **Score-Matching** (ver Hyvärinen (2005); Vincent (2011)).

Pero, **¿cómo aprendemos a aproximar algo que es intractable?**

Score-matching

Al término $\nabla \log p_k$ se le conoce como el **score (de Stein)**; y por eso hablamos de un problema de **Score-Matching** (ver Hyvärinen (2005); Vincent (2011)).
Pero, **¿cómo aprendemos a aproximar algo que es intractable?**

- Usemos la siguiente identidad (ver Efron (2011)):

$$\nabla \log p_k(x_k) = \nabla p_k(x_k)/p_k(x_k) = \int_{\mathbb{R}^d} \nabla \log p_{k|0}(x_k|x_0) p_{0,k}(x_0, x_k) dx_0 / p_k(x_k)$$

$$= \int_{\mathbb{R}^d} \nabla \log p_{k|0}(x_k|x_0) p_{0|k}(x_0|x_k) dx_0 = \mathbb{E}_{p_{0|k}(\cdot|x_k)}[\nabla \log p_{k|0}(x_k|X_0)].$$

Score-matching

Al término $\nabla \log p_k$ se le conoce como el **score (de Stein)**; y por eso hablamos de un problema de **Score-Matching** (ver Hyvärinen (2005); Vincent (2011)).
Pero, **¿cómo aprendemos a aproximar algo que es intractable?**

- Usemos la siguiente identidad (ver Efron (2011)):

$$
\begin{aligned}
\nabla \log p_k(x_k) = \nabla p_k(x_k)/p_k(x_k) &= \int_{\mathbb{R}^d} \nabla \log p_{k|0}(x_k|x_0) p_{0,k}(x_0, x_k) dx_0 / p_k(x_k) \\
&= \int_{\mathbb{R}^d} \nabla \log p_{k|0}(x_k|x_0) p_{0|k}(x_0|x_k) dx_0 = \mathbb{E}_{p_{0|k}(\cdot|x_k)}[\nabla \log p_{k|0}(x_k|X_0)].
\end{aligned}
$$

- De este modo, tenemos una **expresión intermedia** más razonable:
  - $\nabla \log p_{k|0}(x_k|x_0)$ **sí es manejable** (es una transición *hacia adelante*).
  - Pero la **esperanza condicional** NO (es una transición *hacia atrás*).
- Pese a todo, esto nos permitirá plantear una **función de pérdida razonable**.

Score matching

- Recordando la siguiente propiedad de la **esperanza condicional**:
  - $Y = \mathbb{E}[X|U]$ si $Y = f(U)$, con $f = \arg\min\{\mathbb{E}[||X - f(U)||^2] : f \in L^2(U)\}$.
- Y dado que: $\nabla \log p_k(X_k) = \mathbb{E}[\nabla \log p_{k|0}(X_k|X_0)|X_k]$.
- Podemos decir que:

$$\nabla \log p_k = \arg\min\{\mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2] : f \in L^2(p_k)\}.$$

Score matching

- Recordando la siguiente propiedad de la **esperanza condicional**:
  - $Y = \mathbb{E}[X|U]$ si $Y = f(U)$, con $f = \arg\min\{\mathbb{E}[||X - f(U)||^2] : f \in L^2(U)\}$.
- Y dado que: $\nabla \log p_k(X_k) = \mathbb{E}[\nabla \log p_{k|0}(X_k|X_0)|X_k]$.
- Podemos decir que:

$$\nabla \log p_k = \arg\min\{\mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2] : f \in L^2(p_k)\}.$$

- Es decir, el **score minimiza una función de pérdida que sí podemos calcular**:
  - $\nabla \log p_{k|0}(x_k|x_0)$ es una transición forward!!
  - La esperanza se puede aproximar con **Monte Carlo** (distribución conjunta).
- Notar que esto es válido $\forall k \in \{0, ..., N-1\}$.

Denoising Score matching

- A la función de pérdida anterior se le conoce como **Denoising Score Matching**:

$$DSM(f) = \mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2].$$

Denoising Score matching

- A la función de pérdida anterior se le conoce como **Denoising Score Matching**:

$$DSM(f) = \mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2].$$

- Donde, como conocemos las **transiciones forward**, sabemos que:

$$X_k = m_k X_0 + \sqrt{2\gamma}\sum_{j=1}^{k}(1-\gamma)^{k-j} Z_k = m_k X_0 + \sigma_k \hat{Z}_{j+1}, \ \hat{Z}_k \sim N(0, Id)$$

De modo que:

  - $\log p_{k|0}(x_k|x_0) = -||x_k - m_k x_0||^2/(2\sigma_k^2) + C_k$, con $m_k = (1-\gamma)^k, \sigma_k^2 = \{1-(1-\gamma)^{2k}\}/(1-\gamma/2), C_k$ independiente de x_k; y, por ende: $\nabla \log p_{k|0}(x_k|x_0) = -(x_k - m_k x_0)/\sigma_k^2$.
  - Y, aplicado en la dinámica misma: $\nabla \log p_{k|0}(X_k|X_0) = -\hat{Z}_k/\sigma_k^2$, por lo que la **DSM** nos dice **cuánto es f capaz de predecir el *ruido residual***.

Implicit Score matching

Formulación alternativa: **Implicit Score Matching** (ISM), dado que:

$$\mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2]$$
$$= \mathbb{E}[||f(X_k)||^2] - 2\mathbb{E}[\langle f(X_k), \nabla \log p_{k|0}(X_k|X_0)\rangle] + \mathbb{E}[||\nabla \log p_{k|0}(X_k|X_0)||^2],$$ y:

$$\mathbb{E}[\langle f(X_k), \nabla \log p_{k|0}(X_k|X_0)\rangle|X_0] = \int_{\mathbb{R}^d} \langle f(x_k), \nabla \log p_{k|0}(x_k|X_0)\rangle p_{k|0}(x_k|X_0)dx_k$$
$$= \int_{\mathbb{R}^d} \langle f(x_k), \nabla p_{k|0}(x_k|X_0)\rangle dx_k = -\int_{\mathbb{R}^d} \mathrm{div}(f(x_k))p_{k|0}(x_k|X_0)dx_k = -\mathbb{E}[\mathrm{div}(f(X_k))|X_0].$$

Por lo que:
$\mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2] = \mathbb{E}[||f(X_k)||^2 + 2\mathrm{div}(f(X_k))] + \mathbb{E}[||\nabla \log p_{k|0}(X_k|X_0)||^2]$. Y podemos acceder al **score** sin necesitar la **densidad de transición**:

$$\nabla \log p_k = \arg\min\{\mathbb{E}[\frac{1}{2}||f(X_k)||^2 + \mathrm{div}(f(X_k))] : f \in L^2(p_k)\}.$$

¿Cómo hacemos este Score-Matching?

- Elegimos la pérdida **DSM** o **ISM** para todo tiempo $k \in \{1, ..., N\}$
  - $DSM_k(f) = \mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2]$.
  - $ISM_k(f) = \mathbb{E}[\frac{1}{2}||f(X_k)||^2 + \mathrm{div}(f(X_k))]$.

¿Cómo hacemos este Score-Matching?

- Elegimos la pérdida **DSM** o **ISM** para todo tiempo $k \in \{1, ..., N\}$
  - $DSM_k(f) = \mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2]$.
  - $ISM_k(f) = \mathbb{E}[\frac{1}{2}||f(X_k)||^2 + \mathrm{div}(f(X_k))]$.
- Definiendo la **pérdida integrada** en toda la trayectoria:
  - $\ell^{DSM}(f) = \sum_{k=1}^{N} \lambda_k DSM_k(f(k, \cdot))$,
  - $\ell^{ISM}(f) = \sum_{k=1}^{N} \lambda_k ISM_k(f(k, \cdot))$.
  - Donde tenemos una función de **ponderación** de cada tiempo $\lambda_k \geq 0$.

¿Cómo hacemos este Score-Matching?

- Elegimos la pérdida **DSM** o **ISM** para todo tiempo $k \in \{1, ..., N\}$
  - $DSM_k(f) = \mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2]$.
  - $ISM_k(f) = \mathbb{E}[\frac{1}{2}||f(X_k)||^2 + \mathrm{div}(f(X_k))]$.
- Definiendo la **pérdida integrada** en toda la trayectoria:
  - $\ell^{DSM}(f) = \sum_{k=1}^{N} \lambda_k DSM_k(f(k, \cdot))$,
  - $\ell^{ISM}(f) = \sum_{k=1}^{N} \lambda_k ISM_k(f(k, \cdot))$.
  - Donde tenemos una función de **ponderación** de cada tiempo $\lambda_k \geq 0$.
- Consideramos $\{s_\theta\}_{\theta \in \Theta}$ una **familia paramétrica de funciones** $s_\theta : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}^d$ (la primera variable es la *temporal*).
  - Usualmente, $\{s_\theta\}_{\theta \in \Theta}$ es una familia de **redes neuronales**.
  - Optimizamos $\ell^{DSM}(\theta) = \ell^{DSM}(s_\theta)$ o $\ell^{ISM}(\theta) = \ell^{ISM}(s_\theta)$ sobre Θ.

¿Cómo hacemos este Score-Matching?

- Elegimos la pérdida **DSM** o **ISM** para todo tiempo $k \in \{1, ..., N\}$
  - $DSM_k(f) = \mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2]$.
  - $ISM_k(f) = \mathbb{E}[\frac{1}{2}||f(X_k)||^2 + \text{div}(f(X_k))]$.
- Definiendo la **pérdida integrada** en toda la trayectoria:
  - $\ell^{DSM}(f) = \sum_{k=1}^{N} \lambda_k DSM_k(f(k, \cdot))$,
  - $\ell^{ISM}(f) = \sum_{k=1}^{N} \lambda_k ISM_k(f(k, \cdot))$.
  - Donde tenemos una función de **ponderación** de cada tiempo $\lambda_k \geq 0$.
- Consideramos $\{s_\theta\}_{\theta \in \Theta}$ una **familia paramétrica de funciones** $s_\theta : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}^d$ (la primera variable es la *temporal*).
  - Usualmente, $\{s_\theta\}_{\theta \in \Theta}$ es una familia de **redes neuronales**.
  - Optimizamos $\ell^{DSM}(\theta) = \ell^{DSM}(s_\theta)$ o $\ell^{ISM}(\theta) = \ell^{ISM}(s_\theta)$ sobre Θ.

En el contexto de NNs, **optimizamos la NN** con la función de pérdida escogida, hasta obtener un modelo final s_{θ^*} que, idealmente, es tal que $s_{\theta^*}(k, \cdot) \approx \nabla \log p_k$

¿Y ahora qué?

- Recordemos que nos interesa **samplear la distribución backward**.
- Ya **optimizamos un modelo** s_{θ^*} tal que $s_{\theta^*}(k,\cdot) \approx \nabla \log p_k$.
- Para samplear una **trayectoria backward**, usamos Ancestral Sampling:
    - Partimos de $X_N \sim \mu_Z = N(0, Id)$ (muestreo aproximado de p_N).
    - Iterativamente construimos:

$$X_k = X_{k+1} + \gamma\{X_{k+1} + 2s_{\theta^*}(k\gamma, X_{k+1})\} + \sqrt{2\gamma}Z_{k+1}.$$

    - De modo que X_1 se distribuirá aproximadamente según π.

¿Y ahora qué?

- Recordemos que nos interesa **samplear la distribución backward**.
- Ya **optimizamos un modelo** s_{θ^*} tal que $s_{\theta^*}(k,\cdot) \approx \nabla \log p_k$.
- Para samplear una **trayectoria backward**, usamos Ancestral Sampling:
  - Partimos de $X_N \sim \mu_Z = N(0, Id)$ (muestreo aproximado de p_N).
  - Iterativamente construimos:

$$X_k = X_{k+1} + \gamma\{X_{k+1} + 2s_{\theta^*}(k\gamma, X_{k+1})\} + \sqrt{2\gamma}Z_{k+1}.$$

  - De modo que X_1 se distribuirá aproximadamente según π.

Hemos hablado en tiempo discreto, pero en la tarea veremos que **esto es posible hacerlo en tiempo continuo también**

Modelos en el caso Continuo

- Recordemos que antes mencionamos que la dinámica: $X_{k+1} = X_k - \gamma X_k + \sqrt{2\gamma} Z_{k+1}$ corresponde a la discretización de **Euler-Maruyama** del proceso de **Ornstein-Ulhenbeck (OU)**: $dX_t = -X_t dt + \sqrt{2} dB_t$.

Modelos en el caso Continuo

- Recordemos que antes mencionamos que la dinámica: $X_{k+1} = X_k - \gamma X_k + \sqrt{2\gamma} Z_{k+1}$ corresponde a la discretización de **Euler-Maruyama** del proceso de **Ornstein-Ulhenbeck (OU)**: $dX_t = -X_t dt + \sqrt{2} dB_t$.
- Punto técnico (Ikeda & Watanabe (2014)): dos definiciones de *solución*.
  - Una **solución fuerte**: dado un espacio de probabilidad $(\Omega, \mathcal{F}, \mathbb{P})$ y un movimiento Browniano $(B_t)_{t\geq 0}$ con filtración $(\mathcal{F}_t)_{t\geq 0}$ queremos encontrar $(X_t)_{t\geq 0}$ que es $(\mathcal{F}_t)_{t\geq 0}$ adaptado y para cualquier $t \geq 0$

$$X_t = X_0 + \int_0^t b(s, X_s) ds + \int_0^t \sigma(s, X_s) dB_s.$$

  - Una **solución débil**: solo necesitamos que *exista* un espacio de probabilidad tal que exista tal proceso.

Punto Técnico sobre soluciones a SDE

- La formulación débil es equivalente a un problema de **martingala**, ver Stroock & Varadhan (1997, Capítulo 6).
- Introducimos el **generador infinitesimal** $\mathcal{A} : \mathbb{R}_+ \times C^2(\mathbb{R}^d) \to \mathcal{F}(\mathbb{R}^d)$ tal que para cualquier $f \in C^2(\mathbb{R}_+ \times \mathbb{R}^d)$ y $t \geq 0$

$$\mathcal{A}(t,f)(x) = \langle b(t,x), \nabla f(t,x)\rangle + (1/2)\langle \Sigma(t,x), \nabla^2 f(t,x)\rangle.$$

  Donde $\Sigma = \sigma^\top \sigma$. Moralmente, $\mathbb{E}[\mathcal{A}(t,f)(X_t)] = \text{lím}_{h\to 0}(\mathbb{E}[f(X_{t+h})] - \mathbb{E}[f(X_t)])/h$.
- La existencia de una solución débil es equivalente a la existencia de un proceso $(X_t)_{t\geq 0}$ tal que para cualquier $f \in C^2(\mathbb{R}_+ \times \mathbb{R}^d)$, el proceso $(M_t^f)_{t\geq 0}$ es una **martingala** $(\mathcal{F}_t)_{t\geq 0}$ donde

$$M_t^f = f(t,X_t) - f(0,X_0) - \int_0^t (\partial_s f(s,X_s) + \mathcal{A}(s,f)(X_s))ds.$$

Análisis del proceso Ornstein-Ulhenbeck

- Por su parte, el proceso de **OU** es muy *buena onda*, y admite **soluciones fuertes**:
- $(X_t)_{t\geq 0}$ es un proceso Gausiano y su solución tiene una forma cerrada:

$$X_t = e^{-t}X_0 + B_{1-e^{-2t}}.$$

Análisis del proceso Ornstein-Ulhenbeck

- Por su parte, el proceso de **OU** es muy *buena onda*, y admite **soluciones fuertes**:
- $(X_t)_{t\geq 0}$ es un proceso Gausiano y su solución tiene una forma cerrada:

$$X_t = e^{-t}X_0 + B_{1-e^{-2t}}.$$

- En particular: $W_2(\mathcal{L}(X_t), \mu_Z) \leq e^{-t}\{\mathbb{E}^{1/2}[||X_0||^2] + d^{1/2}\}$.
- También, como μ_Z satisface una desigualdad de log-Sobolev (Bakry et al. (2014)), podemos controlar la divergencia de **Kullback-Leibler** entre $\mathcal{L}(X_t)$ y μ_Z.
- Tasas de convergencia geométricas, **independientes de la dimensión**.

‘Time Reversal’ (reversión temporal)

- Tenemos nuestro proceso **forward** $(X_t)_{t\in[0,T]}$ (de OU en este caso).
- En vez de un **muestreo ancestral**, en tiempo continuo necesitamos calcular la **reversión temporal** del proceso de OU, i.e. el proceso: $(Y_t)_{t\in[0,T]} = (X_{T-t})_{t\in[0,T]}$.

‘Time Reversal’ (reversión temporal)

- Tenemos nuestro proceso **forward** $(X_t)_{t\in[0,T]}$ (de OU en este caso).
- En vez de un **muestreo ancestral**, en tiempo continuo necesitamos calcular la **reversión temporal** del proceso de OU, i.e. el proceso: $(Y_t)_{t\in[0,T]} = (X_{T-t})_{t\in[0,T]}$.
- Lo bueno es que bajo ciertas condiciones, $(Y_t)_{t\in[0,T]}$ es una solución (débil) de la siguiente SDE:

$$dY_t = \{Y_t + 2\nabla \log p_{T-t}(Y_t)\}dt + \sqrt{2}dB_t.$$

  donde $\forall t \in [0, T]$, p_t es la densidad de $\mathcal{L}(X_t)$ (c/r a Lebesgue).

‘Time Reversal’ (reversión temporal)

- Tenemos nuestro proceso **forward** $(X_t)_{t\in[0,T]}$ (de OU en este caso).
- En vez de un **muestreo ancestral**, en tiempo continuo necesitamos calcular la **reversión temporal** del proceso de OU, i.e. el proceso: $(Y_t)_{t\in[0,T]} = (X_{T-t})_{t\in[0,T]}$.
- Lo bueno es que bajo ciertas condiciones, $(Y_t)_{t\in[0,T]}$ es una solución (débil) de la siguiente SDE:

$$dY_t = \{Y_t + 2\nabla \log p_{T-t}(Y_t)\}dt + \sqrt{2}dB_t.$$

  donde $\forall t \in [0, T]$, p_t es la densidad de $\mathcal{L}(X_t)$ (c/r a Lebesgue).
- Es muy similar a la formulación discreta! (de hecho, la versión discreta es la discretización EM de esta SDE! Son cercanas si el paso γ es pequeño).
- Más contexto:
  - Encontrado por primera vez en Anderson (1982); Haussmann & Pardoux (1986).
  - La fórmula de reversión temporal es válida para **difusiones más complicadas**.
  - La fórmula anterior es válida en un **marco abstracto** Cattiaux et al. (2021).

En la práctica

- Al igual que en el caso discreto, el **score de Stein** $\nabla \log p_t$ se aproxima mediante **score-matching** (con versiones adaptadas de DSM/ISM).

En la práctica

- Al igual que en el caso discreto, el **score de Stein** $\nabla \log p_t$ se aproxima mediante **score-matching** (con versiones adaptadas de DSM/ISM).
- Obtendremos una red $s_{\theta*}$ que aproxime bien el score, y samplearemos:

$$Y_{k+1} = Y_k + \gamma\{Y_k + 2s_{\theta*}((N-k)\gamma, Y_k)\} + \sqrt{2\gamma}Z_{k+1}, \quad Y_0 \sim \mu_Z.$$

(de la discretización EM del proceso en que reemplazamos $s_{\theta*}(t, \cdot) \approx \nabla \log p_t$).

En la práctica

Ingeniería Matemática
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

- Al igual que en el caso discreto, el **score de Stein** $\nabla \log p_t$ se aproxima mediante **score-matching** (con versiones adaptadas de DSM/ISM).
- Obtendremos una red s_{θ^*} que aproxime bien el score, y samplearemos:

$$Y_{k+1} = Y_k + \gamma\{Y_k + 2s_{\theta^*}((N-k)\gamma, Y_k)\} + \sqrt{2\gamma}Z_{k+1}, \quad Y_0 \sim \mu_Z.$$

(de la discretización EM del proceso en que reemplazamos $s_{\theta^*}(t,\cdot) \approx \nabla \log p_t$).

¿Qué tan cerca está el modelo generativo de la **distribución real de datos** π?
Tenemos un resultado teórico al respecto!

Resultado teórico

Convergencia de modelos de difusión De Bortoli et al. (2021a)

- Asumir que existe $M \geq 0$ tal que para cualquier $t \in [0, T]$ y $x \in \mathbb{R}^d$
$$||s_{\theta^*}(t, x) - \nabla \log p_t(x)|| \leq M,$$
con $s_{\theta^*} \in C([0, T] \times \mathbb{R}^d, \mathbb{R}^d)$ y condiciones de regularidad en la densidad de π con respecto a la medida de Lebesgue y sus gradientes.
- Entonces existen B, C, $D \geq 0$ tal que para cualquier $N \in \mathbb{N}$ y $\{\gamma_k\}_{k=1}^N$ se cumple lo siguiente:
$$||\mathcal{L}(Y_N) - \pi||_{TV} \leq B \exp[-T] + C(M + \gamma^{1/2}) \exp[DT].$$
donde $T = N\gamma$.

Resultado teórico

Convergencia de modelos de difusión De Bortoli et al. (2021a)

- Asumir que existe $M \geq 0$ tal que para cualquier $t \in [0, T]$ y $x \in \mathbb{R}^d$
$$||s_{\theta^*}(t, x) - \nabla \log p_t(x)|| \leq M,$$
con $s_{\theta^*} \in C([0, T] \times \mathbb{R}^d, \mathbb{R}^d)$ y condiciones de regularidad en la densidad de π con respecto a la medida de Lebesgue y sus gradientes.
- Entonces existen B, C, $D \geq 0$ tal que para cualquier $N \in \mathbb{N}$ y $\{\gamma_k\}_{k=1}^N$ se cumple lo siguiente:
$$||\mathcal{L}(Y_N) - \pi||_{TV} \leq B \exp[-T] + C(M + \gamma^{1/2}) \exp[DT].$$
donde $T = N\gamma$.

- La hipótesis sobre π no se cumple si π se define en una **variedad (manifold)** de $\mathbb{R}^d$ con dimensión $p < d$, ver De Bortoli (2022).
- La suposición de aproximación es fuerte y podría ser **relajada**. También, el término $\exp[DT]$ puede mejorarse a una **dependencia polinómica**.
- Extensión y mejoras Lee et al. (2022); Chen et al. (2022, 2023).

Esquema de la Demostración

- La descomposición central

$$||\mathcal{L}(Y_N) - \pi||_{TV} = ||\pi_0 \hat{R}_N - \pi||_{TV} = ||\pi_0 \hat{R}_N - p_T Q_T||_{TV}$$
$$\leq ||\pi_0 \hat{R}_N - \pi_0 Q_T||_{TV} + ||p_T Q_T - \pi_0 Q_T||_{TV} \leq ||\pi_0 \hat{R}_N - \pi_0 Q_T||_{TV} + ||\pi P_T - \pi_0||_{TV},$$

  donde
  - $\pi_0 = N(0, Id)$, π es la distribución de datos.
  - $(P_t)_{t\in[0,T]}$ es el semi-grupo de Ornstein-Ulhenbeck **hacia adelante**.
  - $(Q_t)_{t\in[0,T]}$ es el semi-grupo de Ornstein-Ulhenbeck **hacia atrás**.
  - $(\hat{R}_n)_{n\in\{1,\dots,N\}}$ es el kernel iterado asociado con la cadena de Markov hacia atrás.
- $||\pi_0 \hat{R}_N - \pi_0 Q_T||_{TV}$ es el **error de aproximación**.
  - Se usa el **teorema de Pinsker** para controlar $KL(\pi_0 \hat{R}_N | \pi_0 Q_T)$.
  - Luego se usa el **teorema de transferencia** y la **teoría de Girsanov**, siguiendo las directrices de Durmus & Moulines (2017); Dalalyan (2017).
- $||\pi P_T - \pi_0||_{TV}$ es el **tiempo de mezcla**.
  - La cota se obtiene simplemente de la **ergodicidad geométrica** del proceso de **OU**.

Trucos para implementar DDM en la práctica

Originalmente, estos modelos eran **difíciles** de entrenar Song & Ermon (2019), ver también este blogpost.

Trucos para implementar DDM en la práctica

Originalmente, estos modelos eran **difíciles** de entrenar Song & Ermon (2019), ver también este blogpost.

En estas slides describimos una **serie de trucos** que facilitan enormemente el entrenamiento de estos modelos. Estos trucos se pueden encontrar en Song et al. (2020b); Song & Ermon (2020); Nichol & Dhariwal (2021); Ho & Salimans (2021); De Bortoli et al. (2021a); Karras et al. (2022).

Trucos para implementar DDM en la práctica

Originalmente, estos modelos eran **difíciles** de entrenar Song & Ermon (2019), ver también este blogpost.

En estas slides describimos una **serie de trucos** que facilitan enormemente el entrenamiento de estos modelos. Estos trucos se pueden encontrar en Song et al. (2020b); Song & Ermon (2020); Nichol & Dhariwal (2021); Ho & Salimans (2021); De Bortoli et al. (2021a); Karras et al. (2022).

Mostraremos sólo algunas de las técnicas principales: **Scheduling en EM de OU, Ponderar la loss y EMA**. Sin embargo, existen muchas otras variantes (e.g. **otros esquemas de discretización** de las SDE (Durham & Gallant (2002); De Bortoli et al. (2021a)), o **scores condicionales** en base a una estructura de clases (Song et al. (2020b); Ho & Salimans (2021); Dhariwal & Nichol (2021)))

Ornstein-Ulhenbeck y discretización

- La discretización del proceso de OU con **paso constante** NO es lo que se suele hacer en la práctica.
- En la práctica, se usa un **schedule** que va reduciendo progresivamente el paso:

$$X_k = X_{k+1} + \gamma_k \{X_{k+1} + 2s_\theta(\sum_{j=0}^{k} \gamma_j, X_{k+1})\} + \sqrt{2\gamma_k} Z_{k+1}$$

Ornstein-Ulhenbeck y discretización

- La discretización del proceso de OU con **paso constante** NO es lo que se suele hacer en la práctica.
- En la práctica, se usa un **schedule** que va reduciendo progresivamente el paso:

$$X_k = X_{k+1} + \gamma_k\{X_{k+1} + 2s_\theta(\sum_{j=0}^{k}\gamma_j, X_{k+1})\} + \sqrt{2\gamma_k}Z_{k+1}$$

  - **Intuición**: necesitamos pasos más grandes cerca de la distribución de datos.
  - e.g. Linear Scheduling $\gamma_k = \gamma_{min} + (\gamma_{max} - \gamma_{min})(N-k)/N$ Song et al. (2020b).
  - Hay muchos tipos distintos de schedule (coseno, tangente hiperbólica) Song & Ermon (2019); Ho et al. (2020); Nichol & Dhariwal (2021); Karras et al. (2022).
  - Cambiar el **schedule de discretización** equivale a hacer un **cambio de tiempo** en el proceso OU original y luego una **discretización fija**.

Ponderación de la función de pérdida

- En la práctica, se utiliza una versión **ponderada** de la pérdida DSM.
  - Recordemos que la **pérdida DSM** viene dada por:

$$DSM_k(f) = \mathbb{E}[||f(X_k) - \nabla \log p_{k|0}(X_k|X_0)||^2].$$

$$\ell^{DSM}(f) = \sum_{k=1}^{N} \lambda_k DSM_k(f(k, \cdot)).$$

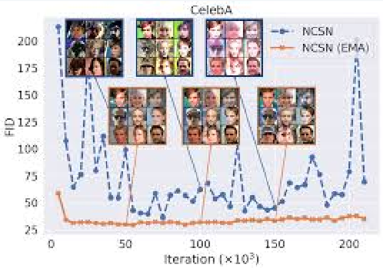
  con $\nabla \log p_{k|0}(X_k|X_0) = -\hat{Z}_k/\sigma_k^2$
- **Intuición**: Si tomamos λ_k como una función de σ_k, se debiese **estabilizar** la pérdida Song et al. (2020b); Ho et al. (2020); Nichol & Dhariwal (2021).
- Esto se justifica con teoría de **Girsanov** en Song et al. (2021); Huang et al. (2021).

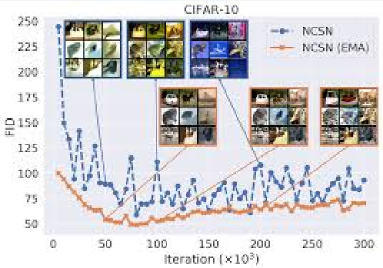
Exponential Moving Average (EMA)

- El entrenamiento de la red es **altamente inestable**.

Exponential Moving Average (EMA)

- El entrenamiento de la red es **altamente inestable**.
- Para regularizar esto, consideramos una **Media Móvil Exponencial** (Exponential Moving Average; EMA) de los **pesos de nuestras NNs**. Es decir, actualizamos:
$$\overline{\theta}_{n+1} = (1-m)\overline{\theta}_n + m\theta_n.$$
  - El parámetro m corresponde al **olvido** de las condiciones iniciales.
  - Los parámetros $\overline{\theta}_n$ se actualizan en cada paso de entrenamiento.





Inestabilidades en el entrenamiento. Imagen extraída de Song & Ermon (2020).

Ingeniería Matemática
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

¡Gracias por su atención!

Joaquín Fontbona, Claudio Muñoz, Diego Olguín, Álvaro Márquez y Javier Maass

Departamento de Ingeniería Matemática
Universidad de Chile

12 de junio


10
GROK

Referencias I

Anderson, B. D. O. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. *Proceedings of the 34th International Conference on Machine Learning*, PMLR, 70, 214-223.

Bakry, D., Gentil, I., Ledoux, M., et al. (2014). Analysis and geometry of Markov diffusion operators. *Springer*, volume 103.

Balaji, Y., et al. (2022). ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*.

Brock, A., Donahue, J., & Simonyan, K. (2019). Large Scale GAN Training for High Fidelity Natural Image Synthesis. *International Conference on Learning Representations (ICLR) 2019*.

Cattiaux, P., Conforti, G., Gentil, I., & Léonard, C. (2021). Time reversal of diffusion processes under a finite entropy condition. *arXiv preprint arXiv:2104.07708*.

Referencias II

Chen, S., Chewi, S., Li, J., Li, Y., Salim, A., & Zhang, A. R. (2022). Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*.

Chen, M., Huang, K., Zhao, T., & Wang, M. (2023). Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data. *arXiv preprint arXiv:2302.07194*.

Dalalyan, A. S. (2017). Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):651–676.

De Bortoli, V. (2022). Convergence of denoising diffusion models under the manifold hypothesis. *arXiv preprint arXiv:2208.05314*.

De Bortoli, V., Doucet, A., Heng, J., & Thornton, J. (2021a). Simulating diffusion bridges with score matching. *arXiv preprint arXiv:2111.07243*.

De Bortoli, V., Thornton, J., Heng, J., & Doucet, A. (2021b). Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34.

Referencias III

Dhariwal, P., & Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34.

Durham, G. B., & Gallant, A. R. (2022). Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business & Economic Statistics*, 20(3):297–338.

Durmus, A., & Moulines, É. (2017). Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *Annals of Applied Probability*, 27(3):1551–1587.

Efron, B. (2011). Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614.

Gao, R., Song, Y., Poole, B., Wu, Y. N., & Kingma, D. P. (2020). Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*.

Goodfellow, I., et al. (2014). Generative Adversarial Networks. *Advances in Neural Information Processing Systems 27 (NIPS 2014)*.

Referencias IV

Google. (2024a). Overview of GAN Structure. *Machine Learning - Google for Developers*. Accedido 10-05-2024. Dirección: `https://developers.google.com/machine-learning/gan/gan_structure`

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved Training of Wasserstein GANs. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*.

Haussmann, U. G., & Pardoux, E. (1986). Time reversal of diffusions. *The Annals of Probability*, pages 1188–1205.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Neural Information Processing Systems (NIPS)*.

Ho, J., & Salimans, T. (2021). Classifier-free diffusion guidance. *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Referencias V

Huang, C.-W., Lim, J. H., & Courville, A. C. (2021). A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4).

Ikeda, N., & Watanabe, S. (2014). Stochastic differential equations and diffusion processes. *Elsevier*.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*.

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive Growing of GANs for Improved Quality, Stability, and Variation. *International Conference on Learning Representations (ICLR) 2018*.

Karras, T., Laine, S., & Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2019*.

Karras, T., Aittala, M., Aila, T., & Laine, S. (2022). Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*.

Referencias VI

Lee, H., Lu, J., & Tan, Y. (2022). Convergence for score-based generative modeling with polynomial complexity. *arXiv preprint arXiv:2206.06227*.

Luhman, E. & Luhman, T. (2021). Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*.

Meng, C., Gao, R., Kingma, D. P., Ermon, S., Ho, J. & Salimans, T. (2022). On distillation of guided diffusion models. *arXiv preprint arXiv:2210.03142*.

Muñoz López, F. I. (2024). *Cálculo del baricentro de Wasserstein bayesiano en conjuntos de imágenes mediante el descenso del gradiente estocástico*. Memoria para optar al título de Ingeniero Civil Matemático, Universidad de Chile.

Nichol, A. Q., & Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. *International Conference on Machine Learning*, pages 8162–8171. PMLR.

Pidstrigach, J. (2022). Score-based generative models detect manifolds. *arXiv preprint arXiv:2206.01018*.

Referencias VII

Poole, B., Jain, A., Barron, J. T., & Mildenhall, B. (2022). Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *International Conference on Learning Representations (ICLR) 2016*.

Radford, A., et al. (2021). Learning transferable visual models from natural language supervision. *International conference on machine learning*, pages 8748–8763. PMLR.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Referencias VIII

Saharia, C., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*.

Salimans, T., et al. (2016). Improved Techniques for Training GANs. *Advances in Neural Information Processing Systems 29 (NIPS 2016)*.

Salimans, T. & Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*.

San Martín-Arastegui, J. (2018). *Teoría de la Medida*, 1.ª ed. Editorial Universitaria. ISBN: 978-956-11-2577-3.

Santana, C. (2017). Creando Caras Artificiales Con Gans mejoradas: Data Coffee #4. Accedido 10-01-2024. Dirección: `https://youtu.be/bMDpMRB7CEs?si=JtMHA9kZ3uKrK14T&t=272`

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Referencias IX

Song, Y., & Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32.

Song, Y., & Ermon, S. (2020). Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448.

Song, J., Meng, C., & Ermon, S. (2020a). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Song, Y., Durkan, C., Murray, I., & Ermon, S. (2021). Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34.

Stroock, D. W. & Varadhan, S. R. S. (1997). Multidimensional diffusion processes. *Springer Science & Business Media*, volume 233.

Referencias X

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674.

Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., & Loy, C. C. (2018). ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. *European Conference on Computer Vision (ECCV)*.

Watson, D., Ho, J., Norouzi, M., & Chan, W. (2021). Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*.

Wikipedia. (2024b). Generative adversarial network - Wikipedia. Accedido 09-05-2024. Dirección: `https://en.wikipedia.org/wiki/Generative_adversarial_network`

Xiao, Z., Kreis, K., & Vahdat, A. (2021). Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*.

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *International Conference on Computer Vision (ICCV) 2017*.