UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

# CÁLCULO DEL BARICENTRO DE WASSERSTEIN BAYESIANO EN CONJUNTOS DE IMÁGENES MEDIANTE DESCENSO DEL GRADIENTE ESTOCÁSTICO

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO

FRANCISCO EMMANUEL MUÑOZ GUAJARDO

PROFESOR GUÍA:
JOAQUÍN FONTBONA TORRES

MIEMBROS DE LA COMISIÓN:
JULIO BACKHOFF VERAGUAS
GONZALO RÍOS DÍAZ
FELIPE TOBAR HENRÍQUEZ

Este trabajo ha sido parcialmente financiado por
CMM ANID BASAL FB210005

SANTIAGO DE CHILE
2024

# Resumen

En el presente trabajo de tesis se expone el desarrollo e implementación del algoritmo de Descenso del Gradiente Estocástico sobre el Espacio de Wasserstein (SGDW) para el cálculo de baricentros de Wasserstein Bayesiano (BWB). La relevancia de este trabajo radica en que, hasta la fecha, no existe implementación del SGDW ni el BWB sobre conjuntos de imágenes.

Para ello, se comienza con una revisión de la teoría de transporte óptimo y la distancia de Wasserstein, seguida de una explicación de los baricentros de Wasserstein de población y la manera en que se pueden calcular utilizando el SGDW. Posteriormente, y dado que se utilizará como medida a priori de la distribución posterior, se presenta la teoría necesaria de las redes generativas.

En este contexto, se introduce un método para entrenar redes generativas adversarias basadas en la distancia de Wasserstein, donde también se entrena simultáneamente un codificador para el generador. El codificador será necesario en el futuro para obtener un proyector sobre la variedad de imágenes deseada.

Se explica la implementación del algoritmo SGDW, se presenta la variación proyectada del algoritmo y se concluye con el cálculo del baricentro de Wasserstein Bayesiano. Cada una de estas secciones está acompañada de experimentos que validan cualitativamente la calidad de las simulaciones.

Finalmente, se menciona que el resultado de este trabajo es una librería en *Python*, utilizada para desarrollar las simulaciones de este trabajo. Como parte del trabajo futuro, se señala que el SGDW proyectado y la red generativa con codificador requieren especial atención.

*"La teoría y la práctica deben interactuar constantemente.*
*La teoría sin la práctica está vacía, y la práctica sin la teoría, ciega."*
*– Cross, 1981, p. 110*

# Agradecimientos

Me gustaría agradecer a mi familia por el constante apoyo que me han dado a mi carrera y a mi persona, que me han permitido llegar hasta este punto. Sin todo el esfuerzo y cariño que me han entregado, no estaría escribiendo estas palabras.

Agradezco también a mis amigos, que me han acompañado desde el inicio de mi carrera y de especialidad. Especialmente, a *los matraqueadores*, que me han acompañado en los momentos de estudio y de relajo, y que me han permitido mantener la cordura en los momentos de mayor estrés, como también de tener la confianza de que siempre estarán ahí para apoyarme.

Continuando con la lista, agradezco a mis amigos de *les ~~otres~~ putres* que han sido como el curso de colegio que nunca tuve, y que me han permitido conocer a personas maravillosas. Agradezco a mis amigos que he hecho en otras carreras, como a los *Kviniones*, o como la Fran o la Kne, que también han sido un apoyo importante en mi vida. También me gustaría agradecer a la Ote, que ha sido mi amiga ya desde 1ero medio, y que aún así y con todo, siempre tiene un tiempo para juntarnos y conversar como si nunca hubiera pasado el tiempo. Quisiera agradecer a todas aquellas personas de la U en las que no nombro aquí, pero ha sido muy grato compartir un ratito con ellas, y que son también muy muy buenos amigos.

Agradezco a todos aquellos profesores que me han aportado en mi formación y que no solo me han enseñado matemáticas, sino que también me han enseñado a ser una mejor persona.

Por último, me gustaría agradecer a la Susi, mi polola, que me ha acompañado en este último tiempo y que ha sido un lugar de descanso y de apoyo en los momentos de mayor estrés. Gracias por estar siempre ahí, y por ser una persona tan maravillosa.

# Tabla de Contenido

# Índice de Ilustraciones

# Índice de Algoritmos

# Introducción

El tema principal de lo que trata este trabajo de tesis es acerca del *Baricentro de Wasserstein Bayesiano*, concepto introducido por Gonzalo Ríos en el 2020 [56]. Para poder explicar este concepto, se puede desglosar en tres partes: **baricentro**, **Wasserstein** y **Bayesiano**. A continuación se desarrolla cada uno de estos conceptos.

Se sabe que para cualquier triángulo formado por tres puntos $x_1, x_2, x_3 \in \mathbb{R}^d$, su *baricentro* corresponde al promedio de los puntos, es decir, el punto $x_b$ tal que $x_b = \frac{x_1+x_2+x_3}{3}$. Por tanto, el **baricentro** es sólo otra manera de llamar al **promedio**. Sin embargo, este baricentro también se puede calcular resolviendo el siguiente problema de optimización:

$$x_b = \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{3} \frac{1}{3} \|x - x_i\|_2^2. \tag{1}$$

En este contexto, la función definida por $\text{dist}(x, y) \stackrel{\text{def}}{=} \|x - y\|$ es una métrica en $\mathbb{R}^d$, y por tanto, el baricentro se puede interpretar como aquel punto que minimiza el promedio de las distancias al cuadrado con respecto a los puntos dados. En otras palabras, es aquel punto que, en promedio, intenta mantenerse lo más cerca posible de los demás puntos, siendo este un **representante** de todos ellos.

Si bien cambiar el cálculo del baricentro por un problema de optimización lo hace considerablemente más difícil, la ganancia de hacerlo radica en la generalidad del planteamiento. En efecto, el promedio utiliza nociones vectoriales (sumas y ponderación), mientras que el problema de optimización definido en (1) utiliza únicamente nociones métricas (distancias), lo cual es una propiedad mucho más general. Esto significa que el baricentro depende **exclusivamente de la métrica que se esté utilizando**, lo que hace posible que **este cambie según la métrica empleada**.

Ahora que se ha explicado el concepto de **baricentro**, se procede a desarrollar la segunda idea del nombre: la parte de **Wasserstein**. Consideremos la situación en la que se tiene una montaña de tierra, y se desea mover esta tierra a un pozo, el cuál posee un volumen igual a la de la montaña. La pregunta es: ¿Cuál es la manera de mover la tierra de la montaña al pozo de la manera más eficiente? La respuesta a esta pregunta la da la teoría de *transporte óptimo*, y a partir de esta rama es que se define la *distancia de Wasserstein*.

En síntesis, la *distancia de Wasserstein* es una métrica entre dos medidas de probabilidad: una medida $\mu \in \mathcal{P}(\mathcal{X})$ que representa la montaña, y otra medida $\nu \in \mathcal{P}(\mathcal{X})$ que representa

el pozo. En este sentido, la distancia de Wasserstein **determina el esfuerzo promedio mínimo** que se necesita para mover la tierra de la montaña al pozo.

Teniendo en cuenta esta analogía, si es que la montaña estuviera más lejos del pozo, entonces la distancia de Wasserstein sería mayor. La propiedad de que esta distancia sea sensible a traslaciones espaciales (que no la cumplen otras métricas y divergencias) le provee a esta distancia múltiples propiedades interesantes.

Son estas propiedades que convierten a la distancia de Wasserstein en un método práctico para calcular baricentros. En donde, si se considera un conjunto de medidas de probabilidad $\{\mu_1, \ldots, \mu_n\}$, y denotando a la $p$-distancia de Wasserstein[1] entre dos medidas $\mu$ y $\nu$ por $W_p(\mu, \nu)$, entonces el *baricentro de Wasserstein* se define por

$$\bar{\mu} \stackrel{\text{def}}{=} \operatorname*{arg\,min}_{\mu \in \mathcal{P}(\mathcal{X})} \sum_{i=1}^{n} \gamma_i W_p(\mu, \mu_i)^p. \tag{2}$$

donde $\{\gamma_1, \ldots, \gamma_n\}$ es una secuencia de pesos no negativos que suman 1.

Cabe destacar que, en la definición del baricentro de Wasserstein, se está considerando un conjunto **finito** de medidas $\mu_1, \ldots, \mu_n$. Sin embargo, esta definición se puede generalizar a un conjunto **infinito** de medidas. Para esto, se considera una medida de probabilidad sobre medidas de probabilidad: una medida $\Gamma \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$, que cumple el rol de los pesos $\gamma_i$ en la ecuación (2). Con este enfoque, la sumatoria es reemplazada por una integral, y la definición del baricentro de Wasserstein se generaliza a

$$\bar{\mu} = \operatorname*{arg\,min}_{\mu \in \mathcal{P}(\mathcal{X})} \int_{\mathcal{P}(\mathcal{X})} W_p(\mu, \nu)^p \, \Gamma(\mathrm{d}\nu). \tag{3}$$

Ahora que ya se conoce el concepto del **baricentro de Wasserstein**, se procede a explicar la componente **Bayesiana** [7]. En el enfoque Bayesiano, es usual llamar por *modelos* a las de medidas de probabilidad, donde al *conjunto de modelos* se le denota por $\mathcal{M}$. Supongamos entonces que tenemos una muestra $\{x_1, \ldots, x_n\}$ de $n$ datos que pertenecen a un espacio $\mathcal{X}$. A partir de esta muestra, se desea determinar un modelo $\bar{\mu}$ en $\mathcal{M} \subseteq \mathcal{P}(\mathcal{X})$ que mejor explique los datos, como si estos datos hubieran sido generados por $\bar{\mu}$.

Para esto, se considera la verosimilitud de un modelo $\mu$ y que lo denotaremos por $\mathcal{L}_n(\mu)$, y un prior sobre los modelos $\Pi(\mathrm{d}\mu) \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$. En virtud de la regla de Bayes, la *medida posterior* se define por

$$\Pi_n(\mathrm{d}\mu) \stackrel{\text{def}}{=} \frac{\mathcal{L}_n(\mu)}{\int_{\mathcal{P}(\mathcal{X})} \mathcal{L}_n(\nu) \, \Pi(\mathrm{d}\nu)} \Pi(\mathrm{d}\mu). \tag{4}$$

De esta manera, la medida posterior le dará mayor peso a aquellas medidas que sean más verosímiles con respecto a los datos.

Así, el *baricentro de Wasserstein Bayesiano* se define utilizando la medida posterior:

$$\bar{\mu} \stackrel{\text{def}}{=} \operatorname*{arg\,min}_{\mu \in \mathcal{M}} \int_{\mathcal{P}(\mathcal{X})} W_p(\mu, \nu)^p \, \Pi_n(\mathrm{d}\nu). \tag{5}$$

[1] La constante $p$ parametriza la distancia de forma similar a las $p$-normas, aunque para esta introducción, no es necesario entrar en detalles.

Cabe mencionar que, gracias a las grandes propiedades que posee la distancia de Wasserstein y el baricentro de Wasserstein, es que ha sido utilizada en una gran variedad de aplicaciones: para comparar los histogramas de colores [57], para comparar imágenes [54], para restaurar imágenes [40], para sintetizar texturas [63], en la clasificación de texto [38] y en particular, en la función de pérdida para las redes generativas adversarias, para aliviar los problemas del desvanecimiento del gradiente y del modo de colapso [4]. Este último tema se trata en mayor profundidad más adelante en la tesis.

Sin embargo, desde la concepción del baricentro de Wasserstein Bayesiano, tan sólo se han realizado algunos experimentos con conjuntos de datos "de juguete"[2] (utilizando distribuciones normales, por ejemplo), para estudiar sus propiedades matemáticas y estadísticas, pero no se han visto aplicaciones en conjuntos de datos más complejos (como podría ser en imágenes).

Una razón de esto, es porque obtener la distribución posterior $\Pi_n(\mathrm{d}\mu)$ puede ser infactible de calcular para conjuntos de datos finitos, incluso si estos son muy grandes. Otro motivo es que, en el caso en que se utiliza un conjunto infinito de modelos, es necesario utilizar el Descenso del Gradiente Estocástico sobre el Espacio de Wasserstein [6], el cuál es un algoritmo que, de la misma manera, es bastante reciente y no existen herramientas computacionales que lo implementen de forma general, y mucho menos para conjuntos de datos complejos.

Por este motivo, el objetivo general de este trabajo de tesis es la implementación eficiente de una librería en *Python* para el cálculo del baricentro de Wasserstein Bayesiano sobre conjuntos de imágenes, utilizando el Descenso del Gradiente Estocástico sobre el Espacio de Wasserstein.

Dicho esto, los objetivos específicos son:

- **OE1**: Obtener aproximaciones de las medidas de probabilidad $\Gamma \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$.
- **OE2**: Implementar el SGDW para una medida $\Gamma \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$.
- **OE3**: Aproximar la medida posterior $\Pi_n(\mathrm{d}\mu)$.
- **OE4**: Calcular el baricentro de Wasserstein Bayesiano.

[2]Se utiliza "conjunto de datos de juguete" para referirse al anglicismo *toy datasets*.

# Capítulo 1

# Transporte Óptimo de Masas

En este capítulo se aborda el problema de transporte óptimo, la distancia de Wasserstein, y el problema de los baricentros de Wasserstein. Además, se presentan algunas propiedades de la distancia de Wasserstein que serán de utilidad en el desarrollo de este trabajo. Es notable señalar que esta sección está basada en la Sección 1.1 del libro de Panaretos y Zemel [50], y en la Sección 2 del libro de Peyré y Cuturi [55].

## Notación

**Definición 1.0.1.** Se definen los siguientes espacios:

- Si $(\mathcal{X}, \text{dist})$ es un espacio métrico, completo y separable, entonces se dice que este es un *espacio métrico Polaco*.
- Si $(\mathcal{X}, \text{dist})$ es un espacio Polaco, $\mathcal{P}(\mathcal{X})$ denotará al *conjunto de medidas de probabilidad* en $\mathcal{X}$, utilizando la $\sigma$-álgebra de Borel.
- Si $(\mathcal{X}, \text{dist})$ es un espacio Polaco, $\mathcal{P}_{\text{ac}}(\mathcal{X})$ denotará al *conjunto de medidas de probabilidad absolutamente continuas* con respecto a una medida de referencia $\lambda$ (como por ejemplo, la de Lebesgue o la cuenta puntos), utilizando la $\sigma$-álgebra de Borel generada por dist.
- $\mathcal{C}(\mathcal{X})$ denotará al *conjunto de funciones continuas* en $\mathcal{X}$.
- $\mathcal{C}_{\text{b}}(\mathcal{X})$ denotará al *conjunto de funciones continuas y acotadas* en $\mathcal{X}$.
- $\text{Lip}_k(\mathcal{X})$ denotará al *conjunto de funciones k-Lipschitz* en $\mathcal{X}$. Mientras que se asumirá que $\text{Lip}(\mathcal{X})$ denotará al *conjunto de funciones* 1-*Lipschitz* en $\mathcal{X}$.
- Al conjunto de funciones de $A$ a $B$ se le denotará por $B^A$.

## 1.1. El Problema de Transporte

A lo largo de esta sección se presentan distintas formulaciones del problema de transporte, comenzando con sus orígenes en el trabajo de Monge [43], seguido de la formulación relajada de Kantorovich [37], y concluyendo al destacar que, en realidad, ambos problemas son equivalentes, como lo establece el teorema de Brenier [11]. Es notable señalar que esta sección está basada en la Sección 1.1 del libro de Panaretos y Zemel [50], y en la Sección 2 del libro de Peyré y Cuturi [55].

### 1.1.1. El problema de Monge

Supongamos que hay un trabajador que debe de transportar una pila de arena a un pozo de igual volumen, situación que se puede ver representada en la Figura 1.1. En esta situación, Monge [43] en el 1781 se preguntó: ¿Cuál es la manera de transportar la arena al pozo con *el menor esfuerzo* para el trabajador?, o dicho de otra manera, ¿Cuál es la forma *óptima* de transportar la arena al pozo?

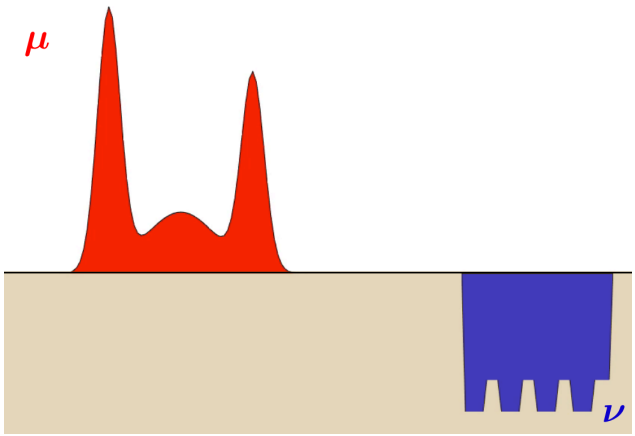



Figura 1.1: Representación de la pila de arena, representada en rojo por la medida $\mu$, y el pozo, representada en azul por la medida $\nu$. Imagen obtenida de [19].

Utilizando términos matemáticos, este problema se formula de la siguiente manera: sea un espacio de arena $\mathcal{X}$, un espacio de pozo $\mathcal{Y}$, y una función de costo $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, donde la función de costo representa el *esfuerzo* necesario para transportar una unidad de arena del punto $x \in \mathcal{X}$ a una posición en el pozo $y \in \mathcal{Y}$. La distribución de la arena se describe mediante una medida $\mu \in \mathcal{P}(\mathcal{X})$, mientras que la forma del pozo se caracteriza mediante una medida $\nu \in \mathcal{P}(\mathcal{Y})$.

La *decisión* sobre cómo transportar la arena se representa mediante una función $T : \mathcal{X} \to \mathcal{Y}$, que asigna a cada punto $x \in \mathcal{X}$ una posición $T(x) \in \mathcal{Y}$ en el pozo. El "esfuerzo total" que el trabajador debe realizar corresponde a "sumar" todos los costos asociados al transporte de la arena desde $x$ hasta la posición correspondiente en el pozo $T(x)$. Más formalmente, el *costo total* de transportar la arena al pozo se obtiene integrando el costo $c$ sobre todo el

espacio de arena $\mathcal{X}$:

$$C(T) \stackrel{\text{def}}{=} \int_{\mathcal{X}} c(x, T(x))\ \mu(\mathrm{d}x). \tag{1.1}$$

Una primera propiedad que se debe exigir a la función $T$ es que preserve la masa total de la arena. Es decir, para cualquier conjunto $B \subseteq \mathcal{Y}$ que represente una región en el pozo con volumen $\nu(B)$, se requiere que el volumen de arena asignado a la región $B$ por la "decisión" $T$ sea exactamente el mismo volumen que se rellenará en $B$. La cantidad de arena que ocupará $B$ bajo la "decisión" $T$ se expresa como $\{x \in \mathcal{X} : T(x) \in B\} = T^{-1}(B)$, y por lo tanto, la condición de preservación de masa implica que $\mu(T^{-1}(B)) = \nu(B)$ para todo $B \subseteq \mathcal{Y}$.

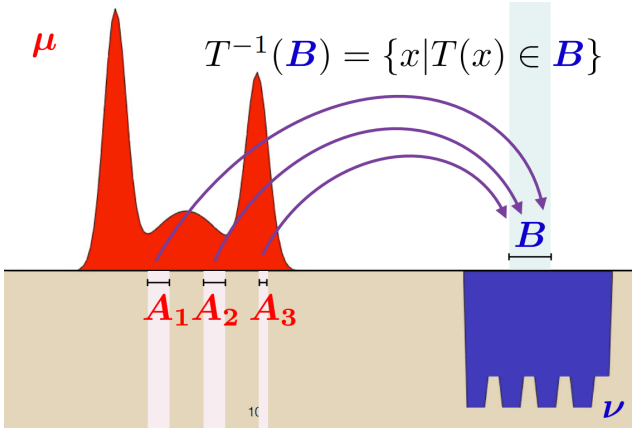



Figura 1.2: Representación de como la función $T$ ha de preservar la masa total de la arena. Imagen obtenida de [19].

Esta condición se ilustra gráficamente en la Figura 1.2 y se formula de manera más precisa mediante la definición del operador *push-forward*:

**Definición 1.1.1** (Operador *push-forward*)**.** Sean $\mu \in \mathcal{P}(\mathcal{X})$, $\nu \in \mathcal{P}(\mathcal{Y})$ dos medidas de probabilidad y sea una función medible $T : \mathcal{X} \to \mathcal{Y}$. Se define el *operador push-forward* de $T$ como la aplicación $T_\sharp : \mathcal{P}(\mathcal{X}) \to \mathcal{P}(\mathcal{Y})$ que satisface la siguiente relación:

$$T_\sharp \mu(B) \stackrel{\text{def}}{=} \mu(T^{-1}(B)) = \nu(B), \quad \forall B \subseteq \mathcal{Y},\ \nu\text{-medible}. \tag{1.2}$$

Equivalentemente, para toda función $\nu$-integrable $g$, el operador push-forward satisface

$$\int_{\mathcal{Y}} g(y)\ \nu(\mathrm{d}y) = \int_{\mathcal{X}} g(T(x))\ \mu(\mathrm{d}x). \tag{1.3}$$

En este caso, se dice que la función $T$ es un *cambio de variables desde $\mu$ hasta $\nu$*.

Ahora que se disponen de todas las herramientas matemáticas necesarias, se puede definir este problema como el de encontrar una función $T$, que representa la manera de transportar la arena al pozo, tal que minimice el costo total de transporte $C(T)$, sujeto a la condición de que la masa total de la arena sea preservada. De manera más formal:

**Definición 1.1.2** (Problema de Monge [43])**.** Dadas dos medidas $\mu \in \mathcal{P}(\mathcal{X})$ y $\nu \in \mathcal{P}(\mathcal{Y})$ y una función de coste $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, el *problema de transporte óptimo de Monge* se define como el problema de encontrar una función medible $T : \mathcal{X} \to \mathcal{Y}$ que minimice el costo total de transporte $C(T)$. Es decir, que minimice la siguiente expresión:

$$\inf_{T:T_\sharp\mu=\nu} \int_{\mathcal{X}} c(x, T(x))\ \mu(\mathrm{d}x). \tag{1.4}$$

A aquella función $T$ que resuelva este problema se le llamará *función de transporte* o *mapa de transporte*, y se denotará por $T_{\mu\to\nu}$, o simplemente $T$ si es que no existe confusión.

El problema introducido por Monge [43] es, en general, considerablemente difícil. Esto se debe principalmente a que el conjunto de mapas de transporte $\{T : T_\sharp\mu = \nu\}$ es intratable. Además, puede suceder que no exista ninguna solución, o en el caso de que exista, esta podría no ser única, como veremos en los siguientes ejemplos:

**Ejemplo 1.1.3.** Consideremos $\mathcal{X} = \{-1, 1\}$, $\mathcal{Y} = \{0\}$, con $\mu = \frac{1}{2}\delta_{-1} + \frac{1}{2}\delta_1$ y $\nu = \delta_0$. En este caso, la función de transporte óptima que minimiza (1.4) es

$$T_{\mu\to\nu}(-1) = 0 \qquad\qquad T_{\mu\to\nu}(1) = 0.$$

Sin embargo, es importante destacar que no existe un transporte óptimo $T_{\nu\to\mu}$ que transporte la masa de $\nu$ a $\mu$. Esto se debe a que la masa de $\nu$ está concentrada en un solo punto, mientras que la masa de $\mu$ está distribuida en dos puntos. No es posible "dividir" la masa debido a la naturaleza determinista de los mapas de transporte. Este ejemplo ilustra un caso en el que el problema de Monge no tiene solución.

**Ejemplo 1.1.4.** Consideremos ahora $\mathcal{X} = \{(1, 1), (-1, -1)\}$, $\mathcal{Y} = \{(-1, 1), (1, -1)\}$, con $\mu = \frac{1}{2}\delta_{(1,1)} + \frac{1}{2}\delta_{(-1,-1)}$ y $\nu = \frac{1}{2}\delta_{(-1,1)} + \frac{1}{2}\delta_{(1,-1)}$. Este ejemplo correspondería a uno en que los puntos de $\mathcal{X}$ y $\mathcal{Y}$ forman un cuadrado. En este escenario, se observa que existen dos funciones de transporte óptimo $T^1_{\mu\to\nu}$ y $T^2_{\mu\to\nu}$ que minimizan (1.4). Estas funciones se expresan como:

$$\begin{aligned} T^1_{\mu\to\nu}(1, 1) &= (1, -1) & T^1_{\mu\to\nu}(-1, -1) &= (-1, 1) \\ T^2_{\mu\to\nu}(1, 1) &= (-1, 1) & T^2_{\mu\to\nu}(-1, -1) &= (1, -1). \end{aligned}$$

Este ejemplo muestra que, en ciertos casos, el problema de Monge puede tener más de una solución.

### 1.1.2. El problema de Kantorovich

Como se pudo apreciar en los Ejemplos 1.1.3 y 1.1.4, el problema de Monge no siempre tiene solución y, en caso de tenerla, puede que esta no sea única. Motivado por esto, Kantorovich [37] propuso una formulación relajada del problema de Monge.

La idea principal de Kantorovich es relajar la naturaleza determinista del mapa de transporte, es decir, el hecho de que la masa de un punto $x$ sea transportada a un único punto $T(x)$, como se ilustró en el Ejemplo 1.1.3. En cambio, Kantorovich propone que la masa de un punto $x$ puede ser potencialmente transportada a múltiples destinos.

Para representar formalmente esta idea, consideremos una medida de probabilidad $\pi \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$. En este contexto, la cantidad $\pi(A \times B)$ representaría la cantidad de arena transportada desde el conjunto $A \subseteq \mathcal{X}$ hasta la región del pozo representada por el conjunto $B \subseteq \mathcal{Y}$. La masa total enviada desde $A$ sería $\pi(A \times \mathcal{Y})$, y la masa total enviada a $B$ sería $\pi(\mathcal{X} \times B)$. En este contexto, $\pi$ estaría preservando la masa total si, y solo si, se cumple que

$$\begin{aligned} \pi(A \times \mathcal{Y}) &= \mu(A), \quad \forall A \subset \mathcal{X} \text{ medible;} \\ \pi(\mathcal{X} \times B) &= \nu(B), \quad \forall B \subset \mathcal{Y} \text{ medible.} \end{aligned}$$

Las medidas $\pi$ que cumplen esta condición se conocen como *couplings*, y se pueden definir de manera más formal de la siguiente manera:

**Definición 1.1.5** (Coupling)**.** Sean $(\mathcal{X}, \mu)$ y $(\mathcal{Y}, \nu)$ dos espacios de probabilidad. Un *coupling* entre $\mu$ y $\nu$ es una medida de probabilidad $\pi \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ tal que sus proyecciones marginales sean $\mu$ y $\nu$, es decir, que cumpla que

$$\pi(A \times \mathcal{Y}) = \mu(A), \quad \pi(\mathcal{X} \times B) = \nu(B), \quad \forall A \subseteq \mathcal{X}, B \subseteq \mathcal{Y} \text{ medibles.} \tag{1.5}$$

Al conjunto de couplings entre $\mu$ y $\nu$ se le denotará por $\mathrm{Cpl}(\mu, \nu)$. Usualmente se les llama a $\mu$ y $\nu$ como la primera y segunda *distribución marginal*, o simplemente *marginales* de $\pi$.

Al igual que en el problema de Monge, se puede definir el costo total de transporte de un coupling $\pi \in \mathrm{Cpl}(\mu, \nu)$, utilizando una función de coste $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$:

$$C(\pi) = \int_{\mathcal{X} \times \mathcal{Y}} c(x, y)\, \pi(\mathrm{d}x, \mathrm{d}y). \tag{1.6}$$

Y utilizando esta definición, se puede definir el problema de transporte óptimo de Kantorovich de forma análoga al problema de Monge:

**Definición 1.1.6** (Problema de Kantorovich [37])**.** Dadas dos medidas $\mu \in \mathcal{P}(\mathcal{X})$ y $\nu \in \mathcal{P}(\mathcal{Y})$ y una función de coste $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, el *problema de transporte óptimo de Kantorovich* se define como el problema de encontrar un coupling $\pi \in \mathrm{Cpl}(\mu, \nu)$ que minimice el costo total de transporte, es decir, que minimice la siguiente expresión:

$$\inf_{\pi \in \mathrm{Cpl}(\mu,\nu)} \int_{\mathcal{X} \times \mathcal{Y}} c(x, y)\, \pi(\mathrm{d}x, \mathrm{d}y). \tag{1.7}$$

Al conjunto de couplings que resuelven este problema se le llama *planes de transporte* o *couplings óptimos*, y un coupling que minimice este problema se denotará por $\pi_{\mu \to \nu}$, o simplemente $\pi$ si es que no existe confusión.

En diferencia con el problema de Monge, el problema de Kantorovich siempre tiene solución, si es que $(\mathcal{X}, \mathcal{Y})$ son espacios compactos y $c$ es continuo. En efecto, $\mathrm{Cpl}(\mu, \nu)$ es compacto para la topología débil de las medidas, $\pi \mapsto \int c\, \mathrm{d}\pi$ es una función continua para esta topología, y la restricción $\pi \in \mathrm{Cpl}(\mu, \nu)$ es no vacía (la medida producto $\mu \otimes \nu$ pertenece a $\mathrm{Cpl}(\mu, \nu)$). Como se busca un ínfimo en un compacto de una función continua, se sigue que existe un coupling óptimo $\pi_{\mu \to \nu}$ que resuelve (1.7).

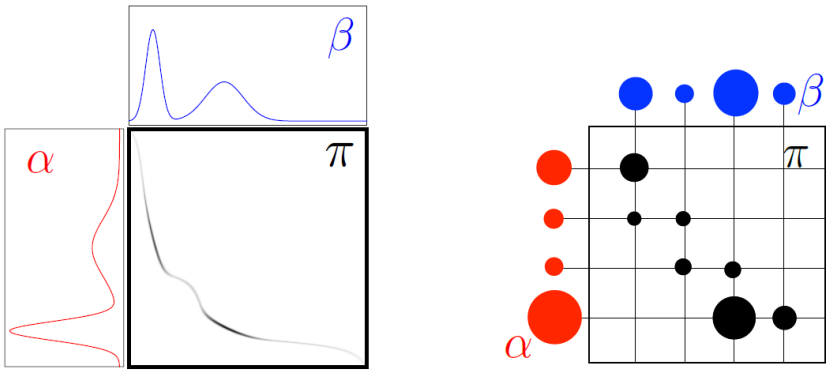



Figura 1.3: Izquierda: coupling óptimo entre dos medidas 1-D continuas con densidad. El coupling está localizado a lo largo del grafo del mapa de transporte óptimo $(x, T(x))$. Derecha: coupling óptimo entre dos medidas discretas. El radio del disco negro es proporcional a la masa transportada en esa coordenada. Imagen obtenida de [55].

Continuando con los ejemplos 1.1.3 y 1.1.4, podemos encontrar el coupling óptimo para cada uno de estos problemas:

**Ejemplo 1.1.7.** Sean los espacios $\mathcal{X}$ y $\mathcal{Y}$ y las medidas $\mu$ y $\nu$ como en el Ejemplo 1.1.3. En este caso, el coupling óptimo entre $\mu$ y $\nu$ corresponde a:

$$\pi_{\mu\to\nu}\{(-1,0)\} = \frac{1}{2} \qquad \pi_{\mu\to\nu}\{(1,0)\} = \frac{1}{2}.$$

Del mismo modo, el coupling óptimo entre $\nu$ y $\mu$ corresponde a:

$$\pi_{\nu\to\mu}\{(0,-1)\} = \frac{1}{2} \qquad \pi_{\nu\to\mu}\{(0,1)\} = \frac{1}{2}.$$

**Ejemplo 1.1.8.** Del mismo modo, Sean los espacios $\mathcal{X}$ y $\mathcal{Y}$ y las medidas $\mu$ y $\nu$ como en el Ejemplo 1.1.4. En este caso, el coupling óptimo entre $\mu$ y $\nu$ corresponde a:

$$\pi_{\mu\to\nu}\left\{\Big((1,1),(-1,1)\Big)\right\} = \frac{1}{4} \qquad \pi_{\mu\to\nu}\left\{\Big((1,1),(1,-1)\Big)\right\} = \frac{1}{4}$$
$$\pi_{\mu\to\nu}\left\{\Big((-1,-1),(-1,1)\Big)\right\} = \frac{1}{4} \qquad \pi_{\mu\to\nu}\left\{\Big((-1,-1),(1,-1)\Big)\right\} = \frac{1}{4}.$$

Para finalizar esta sección, una pregunta que surge naturalmente es: ¿Cuál es relación entre los problemas de Monge y Kantorovich? ¿Si encuentro una solución para el problema de Kantorovich, puedo tener una solución para el problema de Monge? La respuesta es que, bajo algunas condiciones particulares, estos dos problemas están estrechamente relacionados, como lo establece el teorema de Brenier [11]:

**Teorema 1.1.9** (Brenier [11])**.** *Sea $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$, $c(x,y) = \|x-y\|^2$, $\mu \in \mathcal{P}(\mathcal{X})$ y $\nu \in \mathcal{P}(\mathcal{Y})$. Si al menos una de las dos medidas (que s.p.g. supondremos que es $\mu$) tiene densidad con respecto a la medida de Lebesgue, entonces el plan de transporte óptimo $\pi$ es único y esta*

*soportado en el grafo $(x, T(x))$ de un mapa de transporte óptimo $T : \mathcal{X} \to \mathcal{Y}$. Esto significa que el plan de transporte es $\pi = (\mathrm{id}, T)_\sharp \mu$, y cumple la siguiente ecuación:*

$$\forall h \in \mathcal{C}(\mathcal{X} \times \mathcal{Y}), \quad \int_{\mathcal{X}\times\mathcal{Y}} h(x,y)\, \pi(\mathrm{d}x, \mathrm{d}y) = \int_{\mathcal{X}} h(x, T(x))\, \mu(\mathrm{d}x). \tag{1.8}$$

*Más aún, el mapa de transporte óptimo $T$ está únicamente definido como el gradiente de una función convexa $\varphi$, i.e. $T(x) = \boldsymbol{\nabla}\varphi(x)$, donde $\varphi$ es la única función convexa (salvo una constante aditiva) tal que $(\boldsymbol{\nabla}\varphi)_\sharp \mu = \nu$.*

*A aquellos mapas de transporte $\pi \in \mathrm{Cpl}(\mu, \nu)$ que se encuentran definidos por $\pi = (\mathrm{id}, T)_\sharp \mu$, para algún mapa de transporte $T$, se les llama* mapas deterministas*, pues cumple con la propiedad de que si $(X, Y) \sim \pi$, entonces $Y$ queda determinado por $Y = T(X)$.*

Demostración. Se puede encontrar una demostración de este teorema en [55, p. 27] □

## 1.2. La Distancia y el Espacio de Wasserstein

Como se revisó en la sección anterior, cuando se considera el problema de Kantorovich, este problema usualmente tiene solución. Si es que se considera $(\mathcal{X}, \mathrm{dist})$ un espacio métrico, se podría interpretar la distancia como una forma de representar el costo de transportar la masa de un punto $x$ a un punto $y$. Pero una vez que se hace esto, naturalmente surge la pregunta: ¿Qué propiedades pueden surgir al considerar la distancia como una función de coste? Esta sección responde a estas preguntas. Para ello, se empieza definiendo la distancia de Wasserstein, para luego revisar algunas de sus propiedades. Esta sección estará basada en la Sección 6 del libro de Villani [65].

**Definición 1.2.1** (La distancia de Wasserstein)**.** Sea $(\mathcal{X}, \mathrm{dist})$ un espacio Polaco y sea $p \geq 1$. Para dos medidas $\mu, \nu$ sobre $\mathcal{X}$, la distancia de Wasserstein de orden $p$ entre $\mu$ y $\nu$ es definida por medio de la fórmula

$$W_p(\mu, \nu) \stackrel{\text{def}}{=} \left( \inf_{\pi \in \mathrm{Cpl}(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} \mathrm{dist}(x,y)^p\, \pi(\mathrm{d}x, \mathrm{d}y) \right)^{\frac{1}{p}}. \tag{1.9}$$

**Ejemplo 1.2.2.** $W_p(\delta_x, \delta_y) = \mathrm{dist}(x, y)$. Notemos que en este ejemplo, se puede interpretar que la distancia de Wasserstein metriza el "esfuerzo" de llevar la masa del punto $x$ al punto $y$, y que además, es independiente del valor de $p$.

Como el nombre de $W_p$ puede sugerir, esta es una distancia sobre medidas de probabilidad. Sin embargo, si esta es definida sobre todo el espacio $\mathcal{P}(\mathcal{X})$, entonces puede tomar valores de $+\infty$, de forma que en estricto rigor, $W_p$ aún no puede ser considerada una distancia. Para remediar este problema, se definirá un espacio de medidas de probabilidad en el que la distancia de Wasserstein tome valores finitos.

**Definición 1.2.3** (El espacio de Wasserstein)**.** Con los mismos supuestos que en la Definición 1.2.1, se define el espacio de Wasserstein de orden $p$ por medio de

$$\mathcal{W}_p(\mathcal{X}) \stackrel{\text{def}}{=} \left\{ \mu \in \mathcal{P}(\mathcal{X}) \colon \int_{\mathcal{X}} \mathrm{dist}(x, x_0)^p\, \mu(\mathrm{d}x) < \infty \right\}, \tag{1.10}$$

para algún punto fijo $x_0 \in \mathcal{X}$. De esta forma, $W_p$ define una distancia (finita) sobre $\mathcal{W}_p(\mathcal{X})$.

En palabras simples, el espacio de Wasserstein de orden $p$ es el conjunto de medidas de probabilidad en $\mathcal{X}$ cuyo momento de orden $p$ es finito.

En [65, p. 94] se puede encontrar una demostración de que $\mathcal{W}_p(\mathcal{X})$ satisface con los tres axiomas de una distancia. Es más, se puede demostrar que $\mathcal{W}_p(\mathcal{X})$ puede heredar varias propiedades del espacio base $\mathcal{X}$, como lo presenta el siguiente teorema:

**Teorema 1.2.4** (Topología del espacio de Wasserstein [65])**.** *Si* $(\mathcal{X}, \mathrm{dist})$ *es un espacio Polaco, entonces el espacio de Wasserstein* $\mathcal{W}_p(\mathcal{X})$*, metrizado por la distancia de Wasserstein* $W_p$*, es también un espacio Polaco.*

Demostración. Revisar la demostración del Teorema 6.18 en [65, p. 105] □

A partir de ahora, se asumirá que el espacio $\mathcal{W}_p(\mathcal{X})$ siempre estará equipado con su respectiva distancia $W_p$.

*Observación* 1.2.5. A través de la desigualdad de Hölder, se puede demostrar que para $p \leq q$, se tiene que $W_p(\mu, \nu) \leq W_q(\mu, \nu)$, para toda $\mu, \nu \in \mathcal{W}_p(\mathcal{X})$. Y por tanto, las topologías inducidas por las distancias de Wasserstein se van encajonando.

En particular, la distancia de Wasserstein de orden 1, es la más débil de todas. Como norma general, la distancia $W_1$ es la más flexible y fácil de acotar, mientras que la distancia $W_2$ posee mejores propiedades geométricas, pero es más difícil de trabajar.

Vista la distancia y el espacio de Wasserstein, se presentará una caracterización de convergencia en este espacio. Para ello, se definirá la convergencia débil entre medidas de probabilidad.

**Definición 1.2.6** (Convergencia Débil)**.** Sea $(\mathcal{X}, \mathrm{dist})$ un espacio Polaco y sea $p \geq 1$. Se dice que una sucesión de medidas de probabilidad $(\mu_n)_{n\in\mathbb{N}} \subset \mathcal{W}_p(\mathcal{X})$ converge débilmente a $\mu \in \mathcal{W}_p(\mathcal{X})$ si

$$\forall \varphi \in \mathcal{C}_{\mathrm{b}}(\mathcal{X}), \quad \int_{\mathcal{X}} \varphi(x)\, \mathrm{d}\mu_n(x) \to \int_{\mathcal{X}} \varphi(x)\, \mu(\mathrm{d}x). \tag{1.11}$$

y lo denotaremos por $\mu_n \Rightarrow \mu$.

*Nota* 1.2.7. Intuitivamente, que una sucesión de medidas de probabilidad converjan débilmente a una medida $\mu$ significa que es la forma "más fácil" que tiene la sucesión de converger a $\mu$.

**Teorema 1.2.8** (La Distancia de Wasserstein Metriza la Convergencia Débil)**.** *Sea* $(\mathcal{X}, \mathrm{dist})$ *un espacio Polaco y sea* $p \geq 1$*. Entonces, la distancia de Wasserstein* $W_p$ *metriza la convergencia débil en* $\mathcal{W}_p(\mathcal{X})$*.*

*Observación* 1.2.9. En otras palabras, si $(\mu_n)_{n\in\mathbb{N}}$ es una sucesión de medidas de probabilidad en $\mathcal{W}_p(\mathcal{X})$ y $\mu \in \mathcal{W}_p(\mathcal{X})$ otra medida, entonces $\mu_n \Rightarrow \mu$ si y sólo si $W_p(\mu_n, \mu) \to 0$.

**Ejemplo 1.2.10.** Consideremos las siguientes distancias y divergencias entre medidas de probabilidad:

$$\mathrm{TV}(\mu,\nu) \stackrel{\text{def}}{=} \sup_{A\subseteq\mathcal{X}} |\mu(A)-\nu(A)|,$$
$$\mathrm{KL}(\mu \mid \nu) \stackrel{\text{def}}{=} \int_{\mathcal{X}} \log\left(\frac{\mathrm{d}\mu}{\mathrm{d}\nu}\right)\, \mu(\mathrm{d}x),$$
$$\mathrm{JS}(\mu,\nu) \stackrel{\text{def}}{=} \mathrm{KL}\left(\mu \mid \frac{\mu+\nu}{2}\right) + \mathrm{KL}\left(\nu \mid \frac{\mu+\nu}{2}\right),$$

donde la primera es la distancia total variación, la segunda es la divergencia de Kullback-Leibler, y la tercera es la divergencia de Jensen-Shannon.

Si consideramos $\delta_\theta$ y $\delta_0$ medidas de Dirac centradas en $\theta$ y 0 respectivamente, entonces se puede demostrar que

$$W_1(\delta_\theta,\delta_0) = |\theta| \qquad \mathrm{TV}(\delta_\theta,\delta_0) = \begin{cases} 1 & \text{si } \theta \neq 0 \\ 0 & \text{si } \theta = 0 \end{cases}$$
$$\mathrm{KL}(\delta_\theta \mid \delta_0) = \begin{cases} +\infty & \text{si } \theta \neq 0 \\ 0 & \text{si } \theta = 0 \end{cases} \qquad \mathrm{JS}(\delta_\theta,\delta_0) = \begin{cases} \log(2) & \text{si } \theta \neq 0 \\ 0 & \text{si } \theta = 0 \end{cases}$$

Entonces, si tomamos $\theta = \frac{1}{n}$ y dejamos que $n \to \infty$, se tiene que $W_1(\delta_\theta,\delta_0) \to 0$, pero el resto de distancias y divergencias no convergen a 0. Por tanto, se puede notar que la distancia de Wasserstein es la única que es capaz de distinguir entre medidas de probabilidad que no tienen soporte en el mismo punto, gracias a que metriza la convergencia débil.

## 1.3. El Baricentro de Wasserstein

Esta sección presenta el concepto de baricentro de Wasserstein, el cual es una generalización del concepto de promedio para medidas de probabilidad. Este concepto es clave para definir el baricentro de Wasserstein Bayesiano, el cual es el principal objeto de estudio de esta tesis. Para ello, se empieza revisando el concepto de media de Fréchet, luego se define el baricentro de Wasserstein, se revisan técnicas de cálculo de baricentros de población, y finalmente se define el baricentro de Wasserstein Bayesiano. La escritura de esta sección se basa en ejemplos de la Sección 3 de Panaretos y Zemel [50] y en definiciones de la Sección 9.2 de Peyré y Cuturi [55].

### 1.3.1. La Media de Fréchet

**Definición 1.3.1** (Funcional y Media de Fréchet [24])**.** Sea $(\mathcal{X},\mathrm{dist})$ un espacio Polaco. Sean $(x_i)_{i=1}^n \in \mathcal{X}^n$ puntos en $\mathcal{X}$ y sean $(\gamma_i)_{i=1}^n \in \mathbb{R}_+^n$ los pesos asociados a esos puntos. Para cada $p \in \mathcal{X}$, se define el *funcional de Fréchet* por

$$\Psi(p) \stackrel{\text{def}}{=} \sum_{i=1}^n \gamma_i \,\mathrm{dist}(p,x_i)^2. \tag{1.12}$$

Y, en caso de que exista un punto $\bar{x} \in \mathcal{X}$ que minimice el funcional $\Psi$, entonces este se definirá como la *media de Fréchet* de los puntos $(x_i)_{i=1}^n$. Es decir, es aquel punto tal que minimiza el siguiente problema:

$$\bar{x} \stackrel{\text{def}}{=} \arg\min_{p \in \mathcal{X}} \sum_{i=1}^{n} \gamma_i \operatorname{dist}(p, x_i)^2. \tag{1.13}$$

**Ejemplo 1.3.2.** Tomemos $x_1, x_2, x_3 \in \mathbb{R}^2$ tres puntos en el plano, formando un triángulo. Si se define el promedio (o el *baricentro*, en el contexto de la geometría) de estos puntos por $\bar{x} = \frac{1}{3}(x_1 + x_2 + x_3)$, entonces se comprueba fácilmente que este es el único que minimiza el funcional de Fréchet:

$$F(p) = \frac{1}{3} \sum_{i=1}^{3} \|p - x_i\|^2, \tag{1.14}$$

puesto que este funcional se descompone de la siguiente manera:

$$F(p) = F(\bar{x}) + \|p - \bar{x}\|^2. \tag{1.15}$$

Con este ejemplo, se aprecia que la media de Fréchet generaliza la noción de promedio.

La razón por la que resulta interesante estudiar este concepto, es que únicamente utiliza nociones métricas, desligándose de nociones vectoriales. Como se vió en el Ejemplo 1.3.2, el promedio $\bar{x}$ utiliza conceptos vectoriales (suma y ponderación de vectores) mientras que la media de Fréchet utiliza nociones métricas (a través de la distancia $\operatorname{dist}(x, y) = \|x - y\|$), resultando en el mismo promedio.

Lo interesante de la media de Fréchet, es que este concepto es mucho más general, pues la media depende ahora de la distancia que se esté utilizando.

### 1.3.2. El Baricentro de Wasserstein

Como se vió en la sección anterior, la media de Fréchet permite definir una noción de promedio para espacios métricos. Además, el Teorema 1.2.4 establece que si $(\mathcal{X}, \operatorname{dist})$ es un espacio Polaco, entonces $(\mathcal{W}_p(\mathcal{X}), W_p)$ es un espacio Polaco, y en particular, un espacio métrico. Por tanto, se puede definir su respectivo "promedio" utilizando la media de Fréchet:

**Definición 1.3.3** (Baricentro de Wasserstein [1])**.** Sean $(\mu_i)_{i=1}^n$ medidas en $\mathcal{W}_p(\mathcal{X})$ y sean $(\gamma_i)_{i=1}^n \in \mathbb{R}_+^n$ sus pesos asociados. El *baricentro de Wasserstein* se define por medio de

$$\bar{\mu} \stackrel{\text{def}}{=} \arg\inf_{\nu \in \mathcal{W}_p(\mathcal{X})} \sum_{i=1}^{n} \gamma_i W_p(\nu, \mu_i)^p. \tag{1.16}$$

Un ejemplo de distintos baricentros, variando los pesos $(\gamma_i)_i$, se observa en la Figura 1.4.

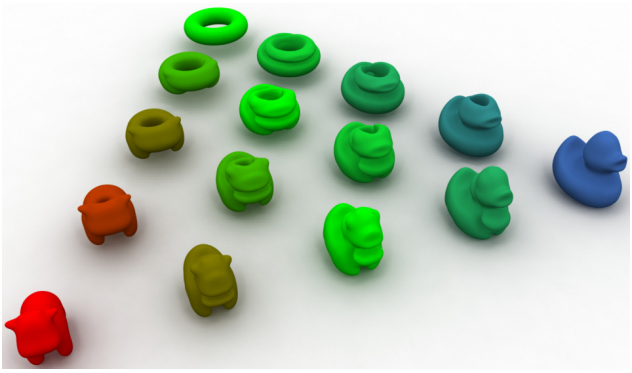


Figura 1.4: Baricentro entre tres figuras en 3 dimensiones, donde se variaron los pesos $\gamma_i$ de forma bilineal. Imagen obtenida de [62].

Cabe destacar que para el caso en donde se considera $n = 2$, el baricentro de Wasserstein recibe un nombre especial: *interpolación geodésica de McCann*, el cuál se define a continuación:

**Definición 1.3.4** (Interpolación Geodésica de McCann [41])**.** Sea $\mu_0, \mu_1 \in \mathcal{W}_p(\mathcal{X})$ dos medidas y $t \in [0, 1]$ el parámetro de interpolación. La *$t$-interpolación geodésica de McCann* (o simplemente, *interpolación geodésica*) se define por medio de:

$$\mu_t \overset{\text{def}}{=} \underset{\mu \in \mathcal{W}_p(\mathcal{X})}{\arg\inf} (1-t) \cdot W_p(\mu_0, \mu)^p + t \cdot W_p(\mu, \mu_1)^p. \tag{1.17}$$

El parámetro $t$ controla la cantidad de influencia de cada medida en el baricentro, en particular, para $t = 0$ se recupera $\mu_0$, y para $t = 1$ se recupera $\mu_1$.

Es posible generalizar aún más la noción de baricentro de Wasserstein a una colección infinita de medidas. Para esto, se considera una medida $\Gamma \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$, que cumple el rol de los pesos $(\gamma_i)_i$ en la definición anterior, lo que se formaliza en la siguiente definición:

**Definición 1.3.5** (Baricentro de población de Wasserstein [8])**.** Sea $\Gamma \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$ una medida. El *baricentro de población de Wasserstein* se define por medio de:

$$\bar{\mu} \overset{\text{def}}{=} \underset{\mu \in \mathcal{P}(\mathcal{X})}{\arg\inf} \int_{\mathcal{P}(\mathcal{X})} W_p(\mu, \nu)^p \, \Gamma(\mathrm{d}\nu). \tag{1.18}$$

La razón por la que la definición anterior es una generalización, es que si se considera $\Gamma = \sum_{i=1}^n \gamma_i \delta_{\mu_i}$, entonces se recupera la primera definición. Además, cabe destacar que no es necesario especificar las medidas $(\mu_i)_{i=1}^n$, pues se encuentran implícitas en la medida $\Gamma$.

*Observación* 1.3.6. Cabe destacar que en la definición anterior, el baricentro de Wasserstein es la medida que minimiza el promedio de las medidas generadas por $\Gamma$. De esta forma, la Ecuación (1.18) se puede reescribir como:

$$\bar{\mu} \overset{\text{def}}{=} \underset{\mu \in \mathcal{P}(\mathcal{X})}{\arg\inf} \left\{ \mathbb{E}_{\nu \sim \Gamma} \left[ W_p(\mu, \nu)^p \right] \right\}. \tag{1.19}$$

### 1.3.3. El Descenso del Gradiente Estocástico en el Espacio de Wasserstein

Para calcular una estimación del baricentro de Wasserstein de una colección finita de medidas $\{\mu_i\}_{i=1}^n$, existen múltiples algoritmos que se pueden utilizar. Por ejemplo, para el caso en que las medidas sean a soporte discreto, se puede deducir un problema de programación lineal [55, ver Cap. 3], y resolver el problema de forma exacta [9], o utilizando una estimación de la solución por medio de métodos iterativos, como el algoritmo de Sinkhorn [17]. En el caso en que las medidas provengan de imágenes, se pueden utilizar métodos convolucionales, como en [62] o en su versión mejorada e insesgada [35].

Sin embargo, para el caso en que la colección de medidas sean infinitas, como lo es en el caso del baricentro de población de Wasserstein, estos métodos no son aplicables. Por este motivo, en los trabajos [5], [6], [56] proponen un algoritmo para encontrar este baricentro, por medio del Descenso del Gradiente Estocástico (SGD).

En los trabajos mencionados, se hacen los siguientes supuestos (que serán válidos sólo para esta sección de la tesis).

*Supuesto* 1.3.7. Se considera $p = 2$, $\mathcal{X} = \mathbb{R}^q$, dist la distancia Euclidiana, y $\lambda$ la medida de Lebesgue. Es más, con estos supuestos se tiene que $\Gamma \in \mathcal{W}_2(\mathcal{W}_{2,\mathrm{ac}}(\mathbb{R}^q))$ y existe un espacio de modelos $\mathcal{M} \subseteq \mathcal{W}_{2,\mathrm{ac}}(\mathbb{R}^q)$ para el cual $\Gamma$ es débilmente cerrado.

*Supuesto* 1.3.8. $\Gamma$ tiene un espacio de modelos $W_2$-compacto: $K_\Gamma \subseteq \mathcal{W}_{2,\mathrm{ac}}(\mathbb{R}^q)$. Más aún, este conjunto es geodésicamente convexo: para cualesquiera $\mu_0, \mu_1 \in K_\Gamma$, la geodésica $\mu_t$ entre $\mu_0$ y $\mu_1$ está contenida en $K_\Gamma$.

*Supuesto* 1.3.9. Para un esquema de paso $(\eta_k)_k \in [0,1]^{\mathbb{N}}$, se asume que $\sum_{k=0}^{\infty} \eta_k = \infty$ y $\sum_{k=0}^{\infty} \eta_k^2 < \infty$.

**Definición 1.3.10** (Secuencia de SGDW [5], [6])**.** Dado $\mu_0 \in K_\Gamma$, $\tilde{\mu}_k \overset{\text{iid}}{\sim} \Gamma$ y $\eta_k > 0$ para $k \in \mathbb{N}$. Se define iterativamente la *Secuencia de Descenso del Gradiente Estocástico sobre el Espacio de Wasserstein* (SGDW) por medio de:

$$\mu_{k+1} \overset{\text{def}}{=} \Big[(1-\eta_k)\,\mathrm{id} + \eta_k T_{\mu_k \to \tilde{\mu}_k}\Big]_\sharp (\mu_k), \quad \text{para } k \in \mathbb{N}, \tag{1.20}$$

donde $\eta_k \in [0,1]$ es el tamaño de paso en la iteración $k$, y $T_{\mu_k \to \tilde{\mu}_k}$ es el transporte óptimo entre $\mu_k$ y $\tilde{\mu}_k$ como en la Definición 1.1.1.

*Observación* 1.3.11. Gracias al supuesto de que $K_\Gamma$ es geodésicamente convexo, se tiene que $\mu_k \in K_\Gamma$ para toda $k \in \mathbb{N}$.

El teorema principal que le da sentido a la definición anterior es el siguiente:

**Teorema 1.3.12.** *Se asumen los Supuestos 1.3.7, 1.3.8, 1.3.9, y que* $\Gamma$ *admite una única media de Karcher. Entonces la secuencia de SGDW* $\{\mu_k\}_k$ *de la Ecuación* (1.20) *converge c.s. al único 2-baricentro de Wasserstein* $\bar{\mu}$ *de* $\Gamma$*. Más aún, se tiene que* $\bar{\mu} \in K_\Gamma$*.*

Además de la Secuencia de SGDW, en [6] se propone una versión de esta secuencia por lotes (o *batched*, en inglés):

**Definición 1.3.13** (Secuencia de SGDW por lotes [6])**.** Dado $\mu_0 \in K_\Gamma$, $\tilde{\mu}_k^{(1)}, \ldots, \tilde{\mu}_k^{(S_k)} \overset{\text{iid}}{\sim} \Gamma$ y $\eta_k > 0$ para $k \in \mathbb{N}$. Se define iterativamente la *Secuencia de Descenso del Gradiente Estocástico sobre el Espacio de Wasserstein por lotes* (BSGDW) por medio de:

$$\mu_{k+1} \overset{\text{def}}{=} \left[ (1-\eta_k)\,\mathrm{id} + \eta_k \frac{1}{S_k} \sum_{j=1}^{S_k} T_{\mu_k \to \tilde{\mu}_k^{(j)}} \right]_\sharp (\mu_k) \quad \text{para } k \in \mathbb{N}, \tag{1.21}$$

donde $\eta_k \in [0,1]$ es el tamaño de paso, y $T_{\mu_k \to \tilde{\mu}_k^{(j)}}$ es el transporte óptimo entre $\mu_k$ y $\tilde{\mu}_k^{(j)}$ como en la Definición 1.1.1.

Los siguientes dos resultados dicen que la Secuencia de BSGDW aún converge, mientras que la segunda dice que los lotes ayudan a reducir la varianza de la estimación:

**Proposición 1.3.14.** *Bajo los supuestos del Teorema 1.3.12, la secuencia de BSGDW* $\{\mu_k\}_k$ *de la Ecuación* (1.21) *converge c.s. al único 2-baricentro de Wasserstein* $\bar{\mu}$ *de* $\Gamma$*.*

**Proposición 1.3.15.** *El estimador por lotes hace que la varianza integrada decrezca linealmente con el tamaño de los lotes.*

Se termina esta sección presentando el algoritmo del SGDW que se presenta en [5].

**Algoritmo 1.1** SGD sobre el Espacio de Wasserstein (SGDW) [5]

**Require:** Acceso a las muestras de $\Gamma(\mathrm{d}\mu) \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$, un esquema de paso $(\eta_k)_k \in [0,1]^{\mathbb{N}}$.

1: $k \leftarrow 0$
2: Muestrear $\mu_0 \sim \Gamma$
3: **repeat**
4: Muestrear $\tilde{\mu}_k \sim \Gamma$ independiente de $\mu_0, \tilde{\mu}_1, \ldots, \tilde{\mu}_{k-1}$.
5: Definir $\mu_k$ de la siguiente manera:

$$\mu_{k+1} \leftarrow \left[ (1-\eta_k)\,\mathrm{id} + \eta_k T_{\mu_k \to \tilde{\mu}_k} \right]_\sharp (\mu_k) \tag{1.22}$$

6: $k \leftarrow k+1$
7: **until** un criterio de detención se alcance.
8: **return** $\mu_k$

El Algoritmo 1.1 posee dos cuellos de botella: el primero es que se debe poder muestrear a partir de $\Gamma(\mathrm{d}\mu)$, y el segundo es la capacidad de calcular el baricentro entre medidas. Para el primer problema, se pueden utilizar técnicas de Markov Chain Monte Carlo (MCMC) [3], [13], [28] para muestrear de $\Gamma$, o alguna otra técnica de muestreo; mientras que para el segundo, se pueden utilizar métodos iterativos como el algoritmo de Sinkhorn [17], o en el caso de imágenes, métodos convolucionales [35], [62].

Como parte del criterio de detención, se pueden utilizar algunos criterios clásicos (como por ejemplo, limitar el número máximo de iteraciones, o tener un tiempo de ejecución máximo), como también definir un criterio de convergencia basada en la distancia de Wasserstein

entre iteraciones consecutivas. En particular, en [5] se describe una manera de calcular la distancia de Wasserstein para este caso:

$$W_2(\mu_{k+1}, \mu_k)^2 = \eta_k^2 \int_{\mathbb{R}^q} |x - T_{\mu_k \to \tilde{\mu}_k}(x)|^2 \, \mu_k(\mathrm{d}x). \tag{1.23}$$

### 1.3.4. El Baricentro de Wasserstein Bayesiano

En esta sección se define el baricentro de Wasserstein Bayesiano, el cual es una variante del baricentro de Wasserstein. La ventaja de este baricentro, es que permite encontrar una medida (el cuál, en este contexto, llamaremos *modelo*) que mejor explique las muestras tomadas. En esta sección se utilizará como referencia principal la sección 5 de la Tesis de G. Ríos [56] y de la Sección 1 de J. Backhoff et al. [5].

Se considera una muestra $D = (x_1, \ldots, x_n)$ que pertenecen a un espacio $\mathcal{X}$ y un conjunto de modelos factibles $\mathcal{M} \subseteq \mathcal{P}(\mathcal{X})$. Aprender un modelo (también conocido como la *selección de un modelo*) utilizando los datos $D$ consiste en escoger un elemento $\mu$ de $\mathcal{M}$ que *mejor explique los datos*, si es que estos hubieran sido generados por $\mu$.

Para este objetivo, se adoptará un enfoque Bayesiano [7], el cuál provee de un marco probabilística para manejar la incertidumbre sobre los modelos. Por tanto, se empezará considerando una medida de probabilidad $\Pi \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$, entendida con una distribución a *priori* sobre un espacio de modelos $\mathcal{M}$. En particular, se tiene que el prior se encuentra soportada en el espacio de modelos $\mathcal{M}$, es decir, $\Pi(\mathcal{M}) = \Pi(\mathcal{P}_{\mathrm{ac}}(\mathcal{X})) = 1$.

Para cada $n \in \mathbb{N}_*$, $\Pi$ induce una ley canónica $\mathbf{\Pi}$ sobre $\mathcal{X}^n \times \mathcal{M}$, representando una ley conjunta de escoger un modelo $\mu$ de acuerdo a la ley $\Pi$, y una muestra i.i.d $D = (x_1, \ldots, x_n) \in \mathcal{X}^n$ generadas por $\mu$. Esto es

$$\mathbf{\Pi}(\mathrm{d}x_1, \ldots, \mathrm{d}x_n, \mathrm{d}\mu) \overset{\text{def}}{=} \mu(\mathrm{d}x_1) \cdots \mu(\mathrm{d}x_n) \, \Pi(\mathrm{d}\mu) \tag{1.24}$$

$$= \prod_{i=1}^{n} \rho_\mu(x_i) \;\; \lambda(\mathrm{d}x_1) \cdots \lambda(\mathrm{d}x_n) \, \Pi(\mathrm{d}\mu), \tag{1.25}$$

donde $\rho_\mu \overset{\text{def}}{=} \frac{\mathrm{d}\mu}{\mathrm{d}\lambda}$ denota la derivada de Radon-Nikodym [47] de $\mu$ con respecto a la medida de referencia $\lambda$. Considerando que $\mathbf{\Pi}(\mathrm{d}x_1, \ldots, \mathrm{d}x_n, \mathrm{d}\mu) = \mathbf{\Pi}(\mathrm{d}x_1, \ldots, \mathrm{d}x_n \mid \mu)\Pi(\mathrm{d}\mu)$, donde $\mathbf{\Pi}(\mathrm{d}x_1, \ldots, \mathrm{d}x_n \mid \mu)$ es su respectivo kernel de transición, se deduce entonces que la ley sobre $\mathcal{X}^n$ de los datos $D$, condicional al modelo $\mu$, está dado por

$$\mathbf{\Pi}(\mathrm{d}x_1, \ldots, \mathrm{d}x_n \mid \mu) \overset{\text{def}}{=} \prod_{i=1}^{n} \rho_\mu(x_i) \;\; \lambda(\mathrm{d}x_1) \cdots \lambda(\mathrm{d}x_n), \tag{1.26}$$

donde se define la *verosimilitud* como la densidad de $\mathbf{\Pi}(\mathrm{d}x_1, \ldots, \mathrm{d}x_n \mid \mu)$ con respecto a $\lambda^{\otimes n}$, la cual se denotará por $\mathcal{L}_n(\mu)$:

$$\mathcal{L}_n(\mu) \overset{\text{def}}{=} \rho_{\mathbf{\Pi}}(x_1, \ldots, x_n \mid \mu) = \prod_{i=1}^{n} \rho_\mu(x_i) \tag{1.27}$$

y la verosimilitud marginal, también conocido como la *evidencia*, que se define por:

$$\rho_{\mathbf{\Pi}}(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \int_{\mathcal{M}} \mathcal{L}_n(\mu)\ \Pi(\mathrm{d}\mu). \tag{1.28}$$

La *densidad posterior* $\mathbf{\Pi}(\mathrm{d}\mu \mid x_1, \ldots, x_n)$, denotada por $\Pi_n(\mathrm{d}\mu)$ por simplicidad, es un elemento de $\mathcal{P}(\mathcal{P}(\mathcal{X}))$, y en virtud a la regla de Bayes, está dado por

$$\Pi_n(\mathrm{d}\mu) = \frac{\rho_{\mathbf{\Pi}}(x_1, \ldots, x_n \mid \mu)}{\rho_{\mathbf{\Pi}}(x_1, \ldots, x_n)}\ \Pi(\mathrm{d}\mu) = \frac{\mathcal{L}_n(\mu)}{\int_{\mathcal{M}} \mathcal{L}_n(\nu)\ \Pi(\mathrm{d}\nu)}\ \Pi(\mathrm{d}\mu). \tag{1.29}$$

La derivada de Radon-Nikodym de $\Pi_n(\mathrm{d}\mu)$ con respecto a $\Pi(\mathrm{d}\mu)$ es la *verosimilitud normalizada* y se denotará por $\Lambda_n = \frac{\mathcal{L}_n(\mu)}{\int_{\mathcal{M}} \mathcal{L}_n(\nu)\, \Pi(\mathrm{d}\nu)}$ , donde se puede notar que, dado que $\mathcal{M}$ es un espacio de modelos para $\Pi$, y que $\Pi_n \ll \Pi$, entonces $\mathcal{M}$ es un espacio de modelos para $\Pi_n$ también.

Ahora que se ha definido la distribución posterior sobre un conjunto de modelos $\mathcal{M}$, podemos considerar que el modelo que mejor explique los datos $D$, correspondería al modelo que minimice el riesgo Bayesiano que utiliza como función de pérdida la distancia de Wasserstein:

$$\bar{\mu}^{(n)} = \operatorname*{arg\,inf}_{\mu \in \mathcal{M}} \int_{\mathcal{P}(\mathcal{X})} W_p(\mu, \nu)^p\ \Pi_n(\mathrm{d}\nu). \tag{1.30}$$

La definición anterior, corresponde justamente con la definición del baricentro de Wasserstein con respecto a $\Pi_n$, por lo que se puede definir el baricentro de Wasserstein Bayesiano de la siguiente manera:

**Definición 1.3.16** (Baricentro de Wasserstein Bayesiano [5])**.** Sea $\Pi \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$ una medida de probabilidad sobre un espacio de modelos $\mathcal{M} \subseteq \mathcal{P}(\mathcal{X})$. Sea $D = (x_1, \ldots, x_n) \in \mathcal{X}^n$ una muestra de tamaño $n$. El *baricentro de Wasserstein Bayesiano* se define como aquel modelo que minimice el siguiente problema:

$$\bar{\mu} \stackrel{\text{def}}{=} \operatorname*{arg\,inf}_{\mu \in \mathcal{M}} \int_{\mathcal{P}(\mathcal{X})} W_p(\mu, \nu)^p\ \Pi_n(\mathrm{d}\nu). \tag{1.31}$$

# Capítulo 2

# Redes Neuronales, Modelos Generativos y sus Aplicaciones

Este capítulo introduce los modelos generativos, empezando por las redes neuronales, viendo las versiones más simples, y terminando con sus variantes basadas en la distancia de Wasserstein. Para ello, se empieza por definir lo que es una red neuronal.

## 2.1. Redes Neuronales

En el último tiempo se ha visto un auge en el uso de las redes neuronales en diversas áreas de la ciencia y la tecnología. Esto se debe a que las redes neuronales han demostrado ser muy efectivas en la resolución de problemas complejos, como la clasificación de imágenes, el procesamiento de lenguaje natural, y la generación de texto e imágenes, entre otros.

Este capítulo introduce los conceptos básicos de las redes neuronales, aunque no se profundiza en los detalles de su funcionamiento. Para ello, se recomienda al lector revisar la literatura especializada en el tema, tales como "Deep learning" [26] y "Deep learning architectures" [14].

El proceso de definición de una red neuronal comienza estableciendo algunas constantes preliminares. El origen de los nombres de las siguientes definiciones se entienden con el Ejemplo 2.1.2. Se define por $L \in \mathbb{N}$ la *profundidad* de la red, la cuál determina el número de *capas ocultas* y la *capa de salida* que tiene la red. Para $\ell \in \{0, \dots, L\}$, se define $d_\ell \in \mathbb{N}$ como el *número de neuronas de la capa* $\ell$.

De esta manera, una red neuronal prealimentada (FFNN por sus siglas en inglés: *feed-forward neural net*) es simplemente una composición de funciones lineales y no lineales. Más precisamente, para $L$ funciones lineales $\{g_{\theta_\ell}^{(\ell)} \colon \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell} \mid \ell = 1, \dots, L\}$, definidas por

$$\begin{aligned} g_{\theta_\ell}^{(\ell)} \colon \mathbb{R}^{d_{\ell-1}} &\to \mathbb{R}^{d_\ell} \\ x_\ell &\mapsto g_{\theta_\ell}^{(\ell)}(x_\ell) = W_\ell x_\ell + b_\ell, \end{aligned}$$

donde $\theta_\ell = (W_\ell, b_\ell)$ corresponden a los *parámetros*, $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ a los *pesos* y $b_\ell \in \mathbb{R}^{d_\ell}$ al *sesgo*[1] de la capa $\ell$. Se considera además $L$ funciones no lineales $\{\sigma^{(\ell)} \colon \mathbb{R} \to \mathbb{R} \mid \ell = 1, \dots, L\}$, a las cuales llamaremos *funciones de activación*. Por convención, se asume que las funciones de activación son aplicadas elemento a elemento, es decir, que si $\sigma$ es una función de activación, entonces $\sigma(x) = (\sigma(x_1), \dots, \sigma(x_n))$ para $x = (x_1, \dots, x_n) \in \mathbb{R}^n$.

Estas funciones lineales y no lineales se pueden componer para formar una red neuronal:

$$f_\theta(x) = \sigma^{(L)} \circ g^{(L)}_{\theta_L} \circ \sigma^{(L-1)} \circ g^{(L-1)}_{\theta_{L-1}} \circ \cdots \circ \sigma^{(1)} \circ g^{(1)}_{\theta_1}(x). \tag{2.1}$$

A este tipo de modelos se les suele representar como grafos dirigidos acíclicos, describiendo cómo las funciones se encuentran compuestas entre sí.

*Observación* 2.1.1. Cabe destacar que es necesario que las funciones de activación $\sigma$ sean no lineales. Pues si lo fueran, entonces la red sería una composición de funciones lineales, lo que resulta en una única función lineal. La problemática que se tendría con esta definición es que no habría ninguna ganancia con hacer la red más profunda, pues todo colapsaría a una única capa. Es por este motivo que a estas funciones se les llama **funciones de activación**, pues "activan" la no-linealidad de la red, permitiendo que esta pueda aprender funciones más complejas.

**Ejemplo 2.1.2.** En la Figura 2.1 se puede observar una representación de una red neuronal con tres capas ocultas y una capa de salida.

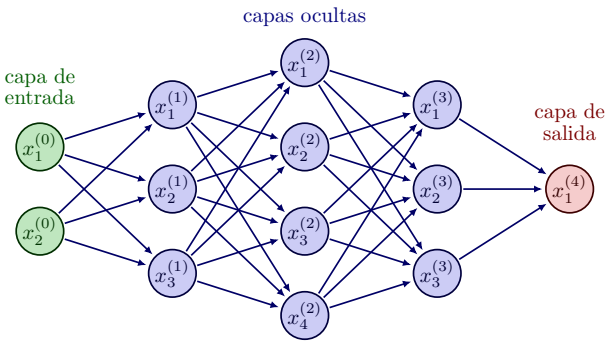



Figura 2.1: Representación de una FFNN con cuatro capas. Elaboración propia.

En este caso, la red neuronal se encuentra compuesta por cinco capas: una de entrada con $d_0 = 2$, tres ocultas con $d_1 = 3$, $d_2 = 4$, $d_3 = 3$, y una de salida con $d_4 = 1$. Cada uno de los nodos $x_i^{(\ell)}$ corresponde a la $i$-ésima *neurona* de la capa $\ell$. Es este el motivo de porqué a $d_\ell$ se les refiere como el número de neuronas de la capa $\ell$. Además, se puede notar que, sin contar la capa de entrada, la red neural tiene una profundidad de $L = 4$ capas, de ahí que se le llame profundidad de la red.

En este caso, la función $g^{(\ell)}_{\theta_\ell}$ es representada a través de los arcos que conectan las neuronas de la capa $\ell - 1$ con las neuronas de la capa $\ell$. Por otro lado, las funciones de activación $\sigma^{(\ell)}$ se encuentran implícitamente utilizadas al definir recursivamente los vectores de neuronas $x^{(\ell)} \in \mathbb{R}^{d_\ell}$ de la siguiente manera:

$$x^{(\ell)} = \begin{cases} x, & \text{si } \ell = 0, \\ \sigma^{(\ell)} \circ g^{(\ell)}_{\theta_\ell}(x^{(\ell-1)}), & \text{en cualquier otro caso.} \end{cases} \qquad \forall \ell = 0, \dots, L. \tag{2.2}$$

[1]En español también se suele referir al sesgo por su anglicismo: *bias*.

Es evidente concluir que la salida de la red neuronal es $f_\theta(x) = x^{(L)}$.

La razón por la que las redes neuronales han tenido tanto éxito en los últimos años se debe a que estas son capaces de aprender funciones muy complejas, siempre y cuando estas posean el número de parámetros suficientes y la cantidad de datos necesario. Este hecho es demostrado por George Cybenko en 1989 en su Teorema de Aproximación Universal (UAT por sus siglas en inglés) [20].

A pesar de que la primera versión del UAT fue propuesta por Cybenko en 1989, el teorema fue popularizado por Kurt Hornik en 1991 [34], debido a que en el teorema de Cybenko utiliza una función de activación sigmoide, mientras que en la versión de Hornik lo extiende a una función de activación continua. Por este motivo, es que a continuación se presenta la versión de Hornik del UAT.

**Teorema 2.1.3** (Teorema de Aproximación Universal [34])**.** *Sea $\sigma \in \mathcal{C}(\mathbb{R}, \mathbb{R})$ una función de activación continua. Sean $d_0, d_2 \in \mathbb{N}$, números naturales, $K \subseteq \mathbb{R}^{d_0}$ un conjunto compacto y $f \in \mathcal{C}\left(K, \mathbb{R}^{d_2}\right)$ la función a aproximar. Entonces, $\sigma$ no es polinomial sí y sólo sí para cada $\varepsilon > 0$, existe $d_1 \in \mathbb{N}$, $W \in \mathbb{R}^{d_1 \times d_0}$, $b \in \mathbb{R}^{d_1}$ y $C \in \mathbb{R}^{d_2 \times d_1}$ tal que:*

$$\sup_{x \in K} \|f(x) - C \cdot \sigma(W \cdot x + b)\| < \varepsilon. \tag{2.3}$$

*Observación* 2.1.4. El Teorema 2.1.3 establece que una red neuronal de una única capa oculta con una función de activación no polinomial (como por ejemplo, una sigmoide) es capaz de aproximar cualquier función continua en un conjunto compacto $K$ con un error arbitrariamente pequeño. Sin embargo, este teorema no establece cuántas neuronas en la capa oculta son necesarias para aproximar dicha función, ni tampoco establece cómo se deben de escoger los parámetros de la red neuronal.

Con el paso de los años, se han propuesto diversas versiones del UAT. Sin embargo, el Teorema 2.1.3 ya ilustra de manera efectiva por qué las redes neuronales son tan efectivas en la aproximación de funciones.

Este capítulo concluye destacando la existencia de múltiples variantes de las redes neuronales, entre las que se incluyen las Redes Neuronales Convolucionales (CNN por sus siglas en inglés) y las Redes Neuronales Recurrentes (RNN por sus siglas en inglés). Estas variantes han ganado popularidad en los campos de la visión computacional y del procesamiento de lenguaje natural, respectivamente. Específicamente, en el presente trabajo se emplean las Redes Neuronales Convolucionales.

## 2.2. Redes Generativas Adversarias

Comencemos esta sección imaginando la siguiente situación[2]:

> *Supongamos que hay un ladrón que desea engañar a un policía entregándole un billete falso. El ladrón, que es un inexperto, le entrega una servilleta, con una cara dibujada en ella, y que en el otro lado de la servilleta tiene escrito: "*esto vale un millón de dólares*". El policía, que ha sido entrenado en la detección de billetes falsos, revisa el billete para comprobar que, efectivamente, es un billete falso.*
>
> *Sin embargo, en vez de enviar a la cárcel al ladrón, lo que hace es decirle al ladrón cuales fueron sus fallos, y de qué manera puede este mejorar en sus falsificaciones. Por su parte, el policía también se entrena más y más en la detección de billetes falsos, pues puede que en algún momento, el ladrón se vuelva tan bueno en la elaboración de billetes falsos, que llegue a engañar al policía con uno de sus billetes.*

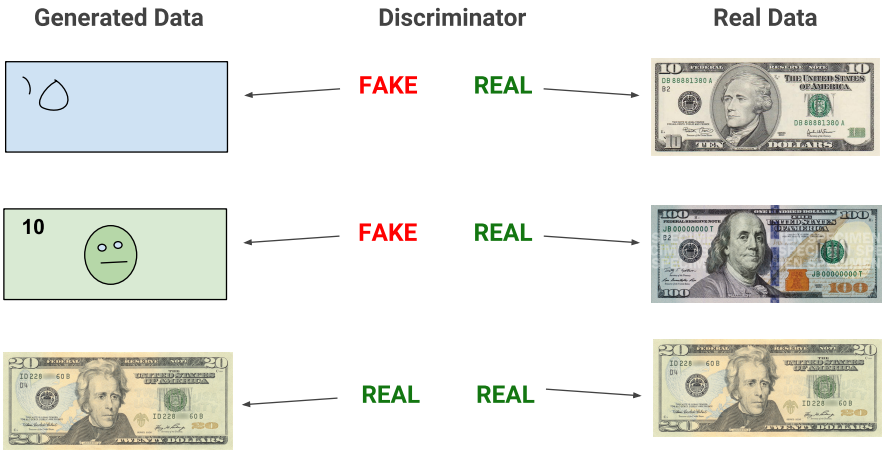



Figura 2.2: Visualización gráfica de la analogía del ladrón y el policía en las GAN. Imagen obtenida de [48].

El marco de las Redes Generativas Adversarias (GAN), introducido en el trabajo de Goodfellow et al. [27], es en esencia, la analogía del ladrón y el policía. La GAN define un juego donde el objetivo de la *generadora* (el ladrón) es la de generar muestras que parezcan reales, mientras que el objetivo de la *discriminadora* (el policía) es el de clasificar las muestras como verdaderas o falsas. En este caso, la generadora se entrena para engañar a la discriminadora, y la discriminadora se entrena para detectar la falsedad de las muestras generadas.

En la siguiente subsección define formalmente una GAN, presenta los teoremas de convergencia de la GAN, y se describe el algoritmo de entrenamiento de una GAN. En las subsecciones posteriores a esta se presentan las variantes de las GAN, como las WGAN, WGAN-GP y WGAN-LP, elementos necesarios para una de las partes de esta tesis.

[2]Este ejemplo es una adaptación y fue inspirado de [59, min. 4:32]

### 2.2.1. GAN

En este trabajo de tesis se plantea una definición formal de una GAN, en el contexto no paramétrico, y utilizando medidas de probabilidades generales. Este enfoque es diferente al que se plantea en el *paper* original [27], donde se utiliza una formulación no paramétrica, pero asumiendo que las medidas de probabilidad poseen densidad. Por tanto, se utiliza la formulación presentada en [25], la cuál se basa en medidas de probabilidad generales, aunque se reescriben los teoremas y demostraciones para garantizar su formalidad. Estos se pueden encontrar en el Anexo A.

Desde el punto de vista de la teoría de juegos, la GAN se define como un juego de dos jugadores: una *generadora* $\mathbb{P}_G$, donde su conjunto de estrategias es $\mathcal{P}(\mathcal{X})$ (con $\mathcal{X}$ el *conjunto de referencia*, e.g. $\mathcal{X} = [0,1]^{n_1 \times n_2}$ para un conjunto de imágenes), y una *discriminadora* $\mathbb{P}_D(\mathrm{d}y \mid x)$, donde su conjunto de estrategias es el conjunto de kernels Markovianos. Con esto, la GAN es un juego de suma cero con la siguiente función valor objetivo:

$$V(\mathbb{P}_G, \mathbb{P}_D) = \mathbb{E}_{X \sim \mathbb{P}_G} \mathbb{E}_{Y \sim \mathbb{P}_D(\mathrm{d}y|X)} [\ln Y] + \mathbb{E}_{\tilde{X} \sim \mathbb{P}_G} \mathbb{E}_{Y \sim \mathbb{P}_D(\mathrm{d}y|\tilde{X})} [\ln(1-Y)], \tag{2.4}$$

donde la generadora busca minimizar la función valor, mientras que la discriminadora busca maximizar esta cantidad.

El objetivo final de la generadora es el de encontrar una distribución $\mathbb{P}_G$ tal que se aproxime a una distribución de referencia $\mathbb{P}_X \in \mathcal{P}(\mathcal{X})$ (del cuál se tiene acceso a través de una medida empírica $\hat{\mathbb{P}}_X = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}$). Por el otro lado, el objetivo de la discriminadora es el de clasificar las muestras verdaderas y falsas, asignándole valores cercano a 1 y 0 respectivamente.

El siguiente teorema nos habla acerca de la naturaleza del discriminador óptimo:

**Teorema 2.2.1** (El discriminador óptimo calcula la divergencia de JS)**.** *Para cualquier estrategia de generador $\mathbb{P}_G$ fijo, existe un único discriminador óptimo $\mathbb{P}_D^*$ que maximiza la función objetivo* (2.4)*, el cuál toma una forma determinista por medio de $\mathbb{P}_D^*(\mathrm{d}y \mid x) = \delta_{D^*(x)}(\mathrm{d}y)$, donde $D^*(x) = \frac{\mathrm{d}\,\mathbb{P}_X}{\mathrm{d}(\mathbb{P}_G + \mathbb{P}_X)}$. En tal caso, la función valor toma la siguiente forma:*

$$V(\mathbb{P}_G, \mathbb{P}_D^*) = \mathrm{JS}\left(\mathbb{P}_X, \mathbb{P}_G\right) - 2\ln 2. \tag{2.5}$$

Demostración. La demostración de este teorema se encuentra en el Anexo A. □

Este teorema nos dice que el discriminador óptimo es aquel que calcula la divergencia de Jensen-Shannon entre la distribución de referencia $\mathbb{P}_X$ y la distribución generada $\mathbb{P}_G$, salvo una constante. Además, nos dice que el discriminador toma una forma determinista, de forma que bastaría con buscar una función (como por ejemplo, una red neuronal) $D\colon \mathcal{X} \to [0,1]$ tal que la aproxime.

Dado que para cada generador $\mathbb{P}_G$ existe un único discriminador óptimo $\mathbb{P}_D^*$, entonces tiene sentido definir la siguiente función:

$$C(\mathbb{P}_G) \overset{\text{def}}{=} V(\mathbb{P}_G, \mathbb{P}_D^*) = \mathrm{JS}\left(\mathbb{P}_X, \mathbb{P}_G\right) - 2\ln 2. \tag{2.6}$$

El siguiente teorema nos dice que existe un único generador óptimo $\mathbb{P}_G^*$ que minimiza la función $C(\mathbb{P}_G)$, y que este corresponde a la distribución de referencia $\mathbb{P}_X$:

**Teorema 2.2.2.** *El mínimo global de la función $C(\mathbb{P}_G)$ se alcanza en $\mathbb{P}_G^* = \mathbb{P}_X$, y el valor mínimo es $C(\mathbb{P}_X) = -\ln 4$.*

Demostración. La demostración de este teorema se encuentra en el Anexo A. □

En la práctica, la forma de implementar la generadora es a través de un *modelo generativo de espacio latente*. Esto es,

$$\mathbb{P}_G(\mathrm{d}x) \stackrel{\text{def}}{=} \int_{\mathcal{Z}} \mathbb{P}_G(\mathrm{d}x \mid z)\ \mathbb{P}_Z(\mathrm{d}z)\,, \tag{2.7}$$

donde $\mathcal{Z}$ es el espacio de las variables latentes, y $\mathbb{P}_Z \in \mathcal{P}(\mathcal{Z})$ es la distribución del espacio latente (típicamente se utiliza la distribución Gaussiana o Uniforme).

*Observación* 2.2.3. En el caso en que el modelo generativo $\mathbb{P}_G(\mathrm{d}x \mid z)$ se mapee de forma determinista a través de $\mathbb{P}_G(\mathrm{d}x \mid z) = \delta_{G(z)}(\mathrm{d}x)$, donde $G\colon \mathcal{Z} \to \mathcal{X}$ es una función medible, entonces la Ecuación (2.7) se simplifica a $\mathbb{P}_G(\mathrm{d}x) = G_\sharp\, \mathbb{P}_Z(\mathrm{d}x)$, donde $G_\sharp$ es el *push-forward* de la función $G$. Intuitivamente, para generar una muestra del modelo generativo, primero se debe muestrear una variable latente $z \sim \mathbb{P}_Z$, para luego mapearla a través de la función $G$ y así obtener una muestra $x$ del modelo generativo.

Por simplicidad, la función $G\colon \mathcal{Z} \to \mathcal{X}$ se estima a través de una red neuronal $G_\theta$, y en virtud al Teorema 2.2.2, se sabe que, en el óptimo, el discriminador también toma una forma determinista. En otras palabras, basta con buscar una función $D\colon \mathcal{X} \to [0,1]$, la cual se estima a través de una red neuronal $D_\phi$.

Con el fin de definir un algoritmo para entrenar una GAN, se definen las siguientes funciones de pérdida para la generadora y la discriminadora:

$$\mathcal{L}_{\text{disc}} = -\frac{1}{N}\sum_{i=1}^{N}\Big[\ln D_\phi(x_i) + \ln\big(1 - D_\phi(G_\theta(z_i))\big)\Big], \tag{2.8}$$

$$\mathcal{L}_{\text{gen}} = -\frac{1}{N}\sum_{i=1}^{N}\ln\big(D_\phi(G_\theta(z_i))\big), \tag{2.9}$$

donde $\{x_i\}_{i=1}^N \sim \mathbb{P}_X$ y $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ son muestras del conjunto de entrenamiento y del espacio latente, respectivamente. La función de pérdida $\mathcal{L}_{\text{disc}}$ proviene de buscar maximizar la función objetivo (2.4), mientras que la función de pérdida $\mathcal{L}_{\text{gen}}$ se deriva de buscar minimizar la misma función objetivo.

De esta manera, el Algoritmo 2.1 describe la forma de estimar los parámetros $\theta$ y $\phi$ de la generadora y la discriminadora, respectivamente. Usualmente, el flujo de este algoritmo se realiza en dos pasos: primero se actualiza la discriminadora $D_\phi$ por $N_d$ iteraciones, y luego se actualiza la generadora $G_\theta$ por una iteración. Este proceso se repite hasta que los parámetros $\theta$ converjan.

**Algoritmo 2.1** Entrenamiento de una Red Generativa Adversaria [27]

**Require:** Tamaño del batch $N$ y número de iteraciones para el discriminador $N_d$.

Inicializar los parámetros de la generadora $G_\theta$ y la discriminadora $D_\phi$.
**while** $\theta$ no ha convergido **do**
  **for** $t = 1, \ldots, N_d$ **do**
    Muestrear $\{x_i\}_{i=1}^N \sim \mathbb{P}_X$ desde el conjunto de entrenamiento.
    Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
    $\mathcal{L}_{\text{disc}} \leftarrow -\frac{1}{N}\sum_{i=1}^N \left[\ln D_\phi(x_i) + \ln\left(1 - D_\phi(G_\theta(z_i))\right)\right]$
    Actualizar $D_\phi$ por medio de descenso de gradiente en $\frac{\partial}{\partial\phi}\mathcal{L}_{\text{disc}}$.
  **end for**
  Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
  $\mathcal{L}_{\text{gen}} \leftarrow -\frac{1}{N}\sum_{i=1}^N \ln\left(D_\phi(G_\theta(z_i))\right)$
  Actualizar $G_\theta$ por medio de descenso de gradiente en $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{gen}}$.
**end while**

*Observación* 2.2.4. El Algoritmo 2.1 empieza por determinar el óptimo de $\max_D V(\mathbb{P}_G, \mathbb{P}_D)$, el cuál proporciona una aproximación para la divergencia de Jensen-Shannon entre $\mathbb{P}_X$ y $\mathbb{P}_G$. Luego, actualiza la generadora $G$ para minimizar dicha divergencia. Con este procedimiento, la distribución $\mathbb{P}_G$ converge a la distribución de referencia $\mathbb{P}_X$ **utilizando la divergencia de Jensen-Shannon**.

Es común que las arquitecturas de tipo GAN sean representadas como en la Figura 2.3. En esta figura, se puede observar que la generadora $G$ toma una variable latente $z$ y la mapea a una muestra $\tilde{x}$, mientras que la discriminadora $D$ toma una muestra $x$ y la clasifica como verdadera o falsa.

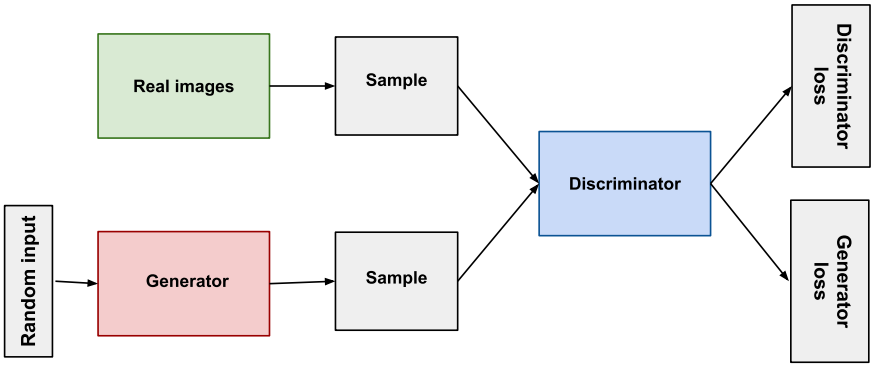



Figura 2.3: Representación gráfica de la arquitectura de una GAN. Imagen obtenida de [48].

### 2.2.2. Wasserstein GAN

La Wasserstein GAN (WGAN) [4] es una variante de las GANs que propone una nueva función de pérdida basada en la distancia de Wasserstein, en lugar de la divergencia de Jensen-Shannon. La principal ventaja de la WGAN es que es más estable y efectiva en la

generación de datos, evitando problemas como el modo de colapso y el desvanecimiento del gradiente.

La función de pérdida se basa en el teorema de dualidad de Kantorovich-Rubinstein, el cuál se enuncia a continuación:

**Teorema 2.2.5** (Teorema de dualidad de Kantorovich-Rubinstein [65])**.** *Sean $\mathbb{P}, \mathbb{Q} \in \mathcal{W}_1(\mathcal{X})$ dos medidas de probabilidad en un espacio métrico $\mathcal{X}$. Entonces, la distancia de Wasserstein entre $\mathbb{P}$ y $\mathbb{Q}$ se puede expresar como:*

$$W_1(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathrm{Lip}(\mathcal{X})} \left\{ \mathbb{E}_{X \sim \mathbb{P}}[f(X)] - \mathbb{E}_{\tilde{X} \sim \mathbb{Q}}[f(\tilde{X})] \right\}. \tag{2.10}$$

Inspirados en este teorema, los autores de la WGAN [4] reemplazan el anterior juego de la GAN por el siguiente:

$$\min_G \max_{f \in \mathrm{Lip}(\mathcal{X})} \mathbb{E}_{X \sim \mathbb{P}_X}[f(X)] - \mathbb{E}_{\tilde{X} \sim \mathbb{P}_G}[f(\tilde{X})], \tag{$P_{\mathrm{WGAN}}$}$$

donde la función $G$ es la *generadora* y $f$ es la *función crítica*. Se puede destacar que, en este juego, el primer objetivo es obtener una estimación de la distancia de Wasserstein, en función de utilizar esta estimación, para minimizar la distancia entre la distribución real y la generada.

En la práctica, se utilizan las redes neuronales $G_\theta : \mathcal{Z} \to \mathcal{X}$ y $f_\omega : \mathcal{X} \to \mathbb{R}$ para aproximar la generadora y la función crítica, respectivamente. Primero, la función crítica $f_\omega$ busca maximizar la discrepancia media de (2.10), con la restricción de que sea 1-Lipschitz para obtener una estimación de la distancia de Wasserstein. Por otro lado, la generadora $G_\theta$ busca, como en el caso de la GAN, minimizar la distancia de Wasserstein entre la distribución real y la generada.

Este procedimiento define a la función de pérdida como la diferencia entre la media de la función crítica evaluada en las muestras reales y generadas, lo que proporciona una estimación de la distancia de Wasserstein:

$$\widehat{W}_N^{(1)}(\omega, \theta, x, z) \overset{\mathrm{def}}{=} \frac{1}{N} \sum_{i=1}^{N} f_\omega(x_i) - \frac{1}{N} \sum_{i=1}^{N} f_\omega(G_\theta(z_i)), \tag{2.11}$$

donde $x = (x_i)_{i=1}^N \sim \mathbb{P}_X$ y $z = (z_i)_{i=1}^N \sim \mathbb{P}_Z$ son muestras de las distribuciones reales y del espacio latente, respectivamente.

Más específicamente, las funciones de pérdida para la generadora y la función crítica son las siguientes:

$$\mathcal{L}_{\mathrm{critic}}(\omega) = -\widehat{W}_N^{(1)}(\omega, \theta, x, z) = \frac{1}{N} \sum_{i=1}^{N} f_\omega(G_\theta(z_i)) - \frac{1}{N} \sum_{i=1}^{N} f_\omega(x_i), \tag{2.12}$$

$$\mathcal{L}_{\mathrm{gen}}(\theta) = \widehat{W}_N^{(1)}(\omega, \theta, x, z) = -\frac{1}{N} \sum_{i=1}^{N} f_\omega(G_\theta(z_i)). \tag{2.13}$$

Destacando que, como la función de pérdida busca maximizar $\mathcal{L}$, entonces este cambia su signo. Por otro lado, la función de pérdida de la generadora busca minimizar $\mathcal{L}$, pero descarta los términos que no dependan de $\theta$.

Con el fin de asegurarse que la función crítica $f_\omega$ sea 1-Lipschitz, en el trabajo original se proponía la restricción de que los pesos de la red fueran acotados. Esta restricción empeora el desempeño de la red, por lo que se han propuesto otras alternativas para garantizar que la red neuronal $f_\omega$ sea 1-Lipschitz. Estas variaciones se revisarán en las siguientes subsecciones.

Paso seguido, la generadora utiliza la aproximación de la distancia de Wasserstein para minimizar la distancia entre la distribución real y la generada (recordando que esta se obtiene a través de un modelo generativo de espacio latente).

De este modo, el algoritmo de entrenamiento de la WGAN se resume en el Algoritmo 2.2.

**Algoritmo 2.2** Entrenamiento de una Wasserstein GAN [4]

**Require:** Tamaño del batch $N$, número de iteraciones para el discriminador $N_d$ y el parámetro de clipping $c$.

Inicializar los parámetros de la generadora $G_\theta$ y la función crítica $f_\omega$.
**while** $\theta$ no ha convergido **do**
  **for** $t = 1, \dots, N_d$ **do**
    Muestrear $\{x_i\}_{i=1}^N \sim \mathbb{P}_X$ desde el conjunto de entrenamiento.
    Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
    $\mathcal{L}_{\text{critic}} \leftarrow \frac{1}{N}\sum_{i=1}^N f_\omega(G_\theta(z_i)) - \frac{1}{N}\sum_{i=1}^N f_\omega(x_i)$ ▷ $\mathcal{L}_{\text{critic}} = -\widehat{W}_N^{(1)}$
    Actualizar $f_\omega$ por medio de descenso de gradiente en $\frac{\partial}{\partial\omega}\mathcal{L}_{\text{critic}}$.
    $\omega \leftarrow \text{clip}(\omega, -c, c)$
  **end for**
  Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
  $\mathcal{L}_{\text{gen}} \leftarrow -\frac{1}{N}\sum_{i=1}^N f_\omega(G_\theta(z_i))$ ▷ $\mathcal{L}_{\text{gen}} = \widehat{W}_N^{(1)}$
  Actualizar $G_\theta$ por medio de descenso de gradiente en $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{gen}}$.
**end while**

### 2.2.3. Wasserstein GAN con Gradiente Penalizado

Los autores de la WGAN reconocen que la restricción de que los pesos de la red sean acotados no es la mejor forma de garantizar que la función crítica sea 1-Lipschitz. Por ello, proponen una nueva forma de garantizar esta restricción, mediante la penalización de gradientes [31]. Esta técnica es llamada Wasserstein GAN con Gradiente Penalizado (WGAN-GP).

La idea por detrás de esta técnica proviene del siguiente corolario:

**Corolario 2.2.6** (Corolario 1 de [31])**.** *Sea $f^*$ la función que minimiza el problema de optimización del Teorema de dualidad de Kantorovich-Rubinstein (ver la Ecuación* (2.10) *del Teorema 2.2.5). Entonces, la norma del gradiente de $f^*$ es* 1 *en casi todo punto bajo las distribuciones $\mathbb{P}$ y $\mathbb{Q}$. Es decir,*

$$\mathbb{E}_{Y\sim\tau}\left[\|\nabla f^*(Y)\|\right] = 1, \tag{2.14}$$

*donde $\tau$ es una distribución definida de forma que $u \sim \text{Unif}[0,1]$, y $Y = uX + (1-u)\tilde{X}$ con $X \sim \mathbb{P}$ y $\tilde{X} \sim \mathbb{Q}$.*

La idea que hay por detrás de este corolario, es que basta penalizar la función de pérdida de la función crítica $f_\omega$ cuando su gradiente no sea 1, para garantizar que esta sea 1-Lipschitz. Es decir, que intercambian el problema de optimización de ($P_{\text{WGAN}}$) por el siguiente:

$$\min_G \max_f \mathbb{E}_{X\sim\mathbb{P}_X}[f(X)] - \mathbb{E}_{\tilde{X}\sim\mathbb{P}_G}[f(\tilde{X})] + \lambda\, \mathbb{E}_{Y\sim\tau}\left[(\|\nabla f(Y)\| - 1)^2\right], \qquad (P_{\text{WGAN-GP}})$$

donde $\lambda$ es un coeficiente de penalización y $\tau$ es una distribución definida como en el Corolario 2.2.6.

Además, como este es una penalización para la función crítica, entonces la función de pérdida de la generadora se mantiene igual a la de la WGAN. De esta forma, la función de pérdida para la función crítica se define de la siguiente manera:

$$p(\omega, x, \tilde{x}) \overset{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N} (\|\nabla f_\omega(u_i x_i + (1-u_i)\tilde{x}_i)\| - 1)^2, \qquad (2.15)$$

$$\mathcal{L}_{\text{critic}}(\omega) \overset{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N} f_\omega(x_i) - \frac{1}{N}\sum_{i=1}^{N} f_\omega(\tilde{x}_i) + \lambda p(\omega), \qquad (2.16)$$

donde $u = (u_i)_{i=1}^N \sim \text{Unif}[0,1]$, $x = (x_i)_{i=1}^N \sim \mathbb{P}_X$ y $\tilde{x} = (\tilde{x}_i)_{i=1}^N \sim \mathbb{P}_G$ son muestras de las distribuciones reales y generadas, respectivamente.

**Algoritmo 2.3** Penalización del Gradiente

**function** GRADPENALTY($\omega$, $x$, $\tilde{x}$)
  Muestrear $u = (u_i)_{i=1}^N \sim \text{Unif}[0,1]$.
  $\hat{x} \leftarrow (u_i x_i + (1-u_i)\tilde{x}_i)_{i=1}^N$ ▷ Interpolación lineal
  **return** $\frac{1}{N}\sum_{i=1}^{N} (\|\nabla f_\omega(\hat{x}_i)\| - 1)^2$ ▷ $\mathbb{E}_{\hat{x}\sim\tau}\left[(\|\nabla f_\omega(\hat{x})\| - 1)^2\right]$
**end function**

Como este procedimiento se puede separar en dos pasos, se define un algoritmo para la penalización del gradiente, el cuál se presenta en el Algoritmo 2.3. Con este, se define el algoritmo de entrenamiento de la WGAN-GP en el Algoritmo 2.4.

**Algoritmo 2.4** Entrenamiento de una WGAN con Gradiente Penalizado [31]

**Require:** Tamaño del batch $N$, número de iteraciones para el discriminador $N_d$ y el parámetro de penalización $\lambda$.

Inicializar los parámetros de la generadora $G_\theta$ y la función crítica $f_\omega$.
**while** $\theta$ no ha convergido **do**
  **for** $t = 1, \ldots, N_d$ **do**
    Muestrear $\{x_i\}_{i=1}^N \sim \mathbb{P}_X$ desde el conjunto de entrenamiento.
    Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
    $\tilde{x} \leftarrow (G_\theta(z_i))_{i=1}^N$.
    $\mathcal{L}_{\text{critic}} \leftarrow \frac{1}{N}\sum_{i=1}^N f_\omega(\tilde{x}_i) - \frac{1}{N}\sum_{i=1}^N f_\omega(x_i) + \lambda \cdot \textsc{gradPenalty}(\omega, x, \tilde{x})$
    Actualizar $f_\omega$ por medio de descenso de gradiente en $\frac{\partial}{\partial \omega}\mathcal{L}_{\text{critic}}$.
  **end for**
  Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
  $\mathcal{L}_{\text{gen}} \leftarrow -\frac{1}{N}\sum_{i=1}^N f_\omega(G_\theta(z_i))$
  Actualizar $G_\theta$ por medio de descenso de gradiente en $\frac{\partial}{\partial \theta}\mathcal{L}_{\text{gen}}$.
**end while**

## 2.3. Auto-Encoders

Junto con las GANs, otro tipo de arquitectura que ha sido muy popular en la comunidad del aprendizaje profundo son los Auto-Encoders (AE). Los AE son una arquitectura de redes neuronales que se utiliza para aprender una codificación eficiente de datos no etiquetados. En otras palabras, aprende a "comprimir la información" del conjunto de datos.

### 2.3.1. AE

Al igual que en la sección anterior, se presenta una definición formal de un AE en el contexto no paramétrico, y utilizando medidas de probabilidad generales. En este caso, esta sección se basa en [64], aunque ha sido modificado en el contexto de los AE "vainilla" (y no en su versión Wasserstein, como se presenta en la referencia).

Un AE se compone de dos partes: un codificador (en inglés, *encoder*) y un decodificador (en inglés, *decoder*). Un *codificador* $\mathbb{Q}(\mathrm{d}z \mid x)$ corresponde a un kernel de transición tal que a cada $x \in \mathcal{X}$ le asigna una distribución en el espacio latente $\mathcal{P}(\mathcal{Z})$. Por el otro lado, un *decodificador* $\mathbb{P}_G(\mathrm{d}x \mid z)$ corresponde a un kernel de transición tal que a cada $z \in \mathcal{Z}$ le asigna una distribución en el espacio de referencia $\mathcal{P}(\mathcal{X})$. En este trabajo, se asume que el decodificador es determinista, es decir, que $\mathbb{P}_G(\mathrm{d}x \mid z) = \delta_{G(z)}(\mathrm{d}x)$, donde $G\colon \mathcal{Z} \to \mathcal{X}$ es una función.

En el objetivo de aprender una codificación eficiente, el AE busca minimizar el error de reconstrucción, denotado como $c(x, \tilde{x})$. Este proceso se lleva a cabo de la siguiente manera: primero, se obtiene una muestra $x \sim \mathbb{P}_X$, luego se adquiere una representación latente mediante $\tilde{z} \sim \mathbb{Q}(\mathrm{d}z \mid x)$ y finalmente se logra la reconstrucción a través de $\tilde{x} \sim \mathbb{P}_G(\mathrm{d}x \mid \tilde{z})$. En

este contexto, la función objetivo que el AE busca minimizar es:

$$V(\mathbb{Q}, \mathbb{P}_G) = \mathbb{E}_{X \sim \mathbb{P}_X} \mathbb{E}_{\tilde{Z} \sim \mathbb{Q}(\mathrm{d}z|X)} \left[ c(X, G(\tilde{Z})) \right]. \tag{2.17}$$

Cabe mencionar que en el caso en que el codificador también sea determinista, es decir, que $\mathbb{Q}(\mathrm{d}z \mid x) = \delta_{E(x)}(\mathrm{d}z)$ con $E \colon \mathcal{X} \to \mathcal{Z}$ una función, entonces la variable aleatoria $\mathbb{E}_{\tilde{Z} \sim \mathbb{Q}(\mathrm{d}z|X)} \left[ c(X, G(\tilde{Z})) \right]$ se reduce a simplemente a $c(X, G(E(X)))$. En este caso, la función objetivo se reduce a:

$$V(\mathbb{Q}, \mathbb{P}_G) = \mathbb{E}_{X \sim \mathbb{P}_X} \left[ c(X, G(E(X))) \right]. \tag{2.18}$$

En la práctica, se puede asumir que el codificador se estima por medio de una red neuronal $\mathbb{Q}_\phi(\mathrm{d}z \mid x)$, el cuál puede ser determinista o estocástica. Por otro lado, el decodificador $\mathbb{P}_G$ se puede aproximar por medio de una red neuronal $G_\theta$ de forma tal que $\mathbb{P}_G(\mathrm{d}x \mid z) = \delta_{G_\theta(z)}(\mathrm{d}x)$. En este caso, la función de pérdida del AE se puede estimar de la siguiente manera:

$$\mathcal{L}_{\mathrm{AE}} = \frac{1}{N} \sum_{i=1}^{N} c(x_i, G_\theta(\tilde{z}_i)), \tag{2.19}$$

donde $\{x_i\}_{i=1}^N \sim \mathbb{P}_X$ son muestras del conjunto de entrenamiento y $\{\tilde{z}_i\}_{i=1}^N \sim \mathbb{Q}_\phi(\mathrm{d}z \mid x_i)$ son propagaciones hacia adelante de las muestras $x_i$ a través de la red neuronal $\mathbb{Q}_\phi$.

Dado lo anterior, el Algoritmo 2.5 describe la forma de estimar los parámetros $\phi$ y $\theta$ del codificador y decodificador, respectivamente. Usualmente, el flujo de este algoritmo se realiza de forma iterativa hasta que los parámetros $\phi$ y $\theta$ converjan.

**Algoritmo 2.5** Entrenamiento de un Auto-Encoder

**Require:** Tamaño del batch $N$.
1: Inicializar los parámetros del codificador $\mathbb{Q}_\phi$ y del decodificador $G_\theta$.
2: **while** $\phi$ y $\theta$ no han convergido **do**
3: Muestrear $\{x_i\}_{i=1}^N \sim \mathbb{P}_X$ desde el conjunto de entrenamiento.
4: Muestrear $\tilde{z}_i \sim \mathbb{Q}_\phi(\mathrm{d}z \mid x_i)$ para $i = 1, \ldots, N$.
5: $\mathcal{L}_{\mathrm{AE}} \leftarrow \frac{1}{N} \sum_{i=1}^{N} c(x_i, G_\theta(\tilde{z}_i))$.
6: Actualizar $\mathbb{Q}_\phi$ y $G_\theta$ por medio de descenso de gradiente en $\frac{\partial}{\partial \phi} \mathcal{L}_{\mathrm{AE}}$ y $\frac{\partial}{\partial \theta} \mathcal{L}_{\mathrm{AE}}$.
7: **end while**

### 2.3.2. Wasserstein AE

Al igual que en el caso de la GAN, se puede definir una versión del AE que minimiza la distancia de Wasserstein entre la distribución real y la generada. En este caso, el Wasserstein AE (WAE) [64] busca minimizar la distancia de Wasserstein entre la distribución real y la generada, en lugar de solo minimizar el error de reconstrucción. Para ello, los autores de [64] proponen una función de pérdida basada en el siguiente teorema:

**Teorema 2.3.1** (Teorema Fundamental del WAE [64])**.** *Sea $\mathbb{P}_X \in \mathcal{P}(\mathcal{X})$ cualquiera y $\mathbb{P}_G \in \mathcal{P}(\mathcal{X})$ un modelo generativo definido como en la Ecuación* (2.7)*, donde $\mathbb{P}_G(\mathrm{d}x \mid Z) =$*

*$\delta_{G(Z)}(\mathrm{d}x)$ es determinista para cualquier función $G : \mathcal{Z} \to \mathcal{X}$. Entonces, se cumple lo siguiente:*

$$\inf_{\pi \in \mathrm{Cpl}(\mathbb{P}_X, \mathbb{P}_G)} \mathbb{E}_{(X,Y)\sim\pi}\left[c(X,Y)\right] = \inf_{\mathbb{Q}:\ \mathbb{Q}_Z = \mathbb{P}_Z} \mathbb{E}_{X\sim\mathbb{P}_X}\, \mathbb{E}_{\tilde{Z}\sim\mathbb{Q}(\mathrm{d}z|X)}\left[c(X, G(\tilde{Z}))\right], \tag{2.20}$$

*donde $\mathbb{Q}_Z(\mathrm{d}z) = \int_{\mathcal{X}} \mathbb{Q}(\mathrm{d}z \mid x)\ \mathbb{P}_X(\mathrm{d}x)$ corresponde a la marginal de $\mathbb{Q}$ en el espacio latente y $\mathbb{P}_Z$ corresponde a la distribución en el espacio latente.*

Este teorema establece que, para cualquier función de costo $c$, la distancia de Wasserstein entre la distribución real y la generada se puede calcular de forma similar a un AE, utilizando un codificador $\mathbb{Q}$ y un decodificador $G$. Es más, si es que se compara con la Ecuación (2.17), se puede notar que las funciones objetivos son similares, con la única diferencia de que en este caso, se tiene la restricción $\mathbb{Q}_Z = \mathbb{P}_Z$.

Por tanto, el objetivo del WAE es **minimizar el error de reconstrucción, con la restricción de que la marginal del codificador sea igual a la distribución del espacio latente**. Con esto en mente, el problema de optimización del WAE corresponde a:

$$\min_{G}\ \min_{\mathbb{Q}:\ \mathbb{Q}_Z = \mathbb{P}_Z} \mathbb{E}_{X\sim\mathbb{P}_X}\, \mathbb{E}_{\tilde{Z}\sim\mathbb{Q}(\mathrm{d}z|X)}\left[c(X, G(\tilde{Z}))\right]. \tag{$P_{\mathrm{WAE}}$}$$

Como se tiene una restricción de igualdad, se puede penalizar el problema anterior a través de alguna divergencia entre las distribuciones $\mathbb{Q}_Z$ y $\mathbb{P}_Z$. En el trabajo original, se propone utilizar la Discrepancia Promedio Máxima (MMD) [29] como divergencia. Este se define como:

$$\mathrm{MMD}_k(\mathbb{P}_Z, \mathbb{Q}_Z) = \left\| \mathbb{E}_{Z\sim\mathbb{P}_Z}\left[k(Z,\cdot)\right] - \mathbb{E}_{Z\sim\mathbb{Q}_Z}\left[k(Z,\cdot)\right] \right\|_{\mathcal{H}_k}, \tag{2.21}$$

donde $\mathcal{H}_k$ es el RKHS asociado al kernel $k$. De esta forma, el problema de optimización del WAE se puede reescribir como:

$$\min_{G}\min_{\mathbb{Q}} \mathbb{E}_{X\sim\mathbb{P}_X}\, \mathbb{E}_{\tilde{Z}\sim\mathbb{Q}(\mathrm{d}z|X)}\left[c(X, G(\tilde{Z}))\right] + \lambda \cdot \mathrm{MMD}_k(\mathbb{P}_Z, \mathbb{Q}_Z), \tag{$P_{\mathrm{WAE-MMD}}$}$$

para algún kernel $k$ y un coeficiente de penalización $\lambda > 0$.

La ventaja de utilizar la MMD como divergencia es que esta se comporta bien en el caso de que las distribuciones sean de alta dimensión, así como también para las distribuciones normales, además de poseer un estimador insesgado y consistente [30]:

$$\widehat{\mathrm{MMD}}_k(z, \tilde{z}) = \frac{1}{n(n-1)} \sum_{i\neq j} k(z_i, z_j) + \frac{1}{n(n-1)} \sum_{i\neq j} k(\tilde{z}_i, \tilde{z}_j) - \frac{2}{n^2} \sum_{i,j} k(z_i, \tilde{z}_j), \tag{2.22}$$

donde $z = (z_i)_{i=1}^n$ y $\tilde{z} = (\tilde{z}_i)_{i=1}^n$ son muestras de las distribuciones $\mathbb{P}_Z$ y $\mathbb{Q}_Z$, respectivamente.

**Algoritmo 2.6** Entrenamiento de una Wasserstein Auto-Encoder [64]

**Require:** Tamaño del batch $N$, kernel característico definido-positivo $k$ y el parámetro de penalización $\lambda$.

**function** MMD($z$, $\tilde{z}$)
  **return** $\frac{1}{N(N-1)}\sum_{i\neq j} k(z_i, z_j) + \frac{1}{N(N-1)}\sum_{i\neq j} k(\tilde{z}_i, \tilde{z}_j) - \frac{2}{N^2}\sum_{i,j} k(z_i, \tilde{z}_j)$
**end function**
Inicializar los parámetros del codificador $\mathbb{Q}_\phi$ y del decodificador $G_\theta$.
**while** $\phi$ y $\theta$ no han convergido **do**
  Muestrear $\{x_i\}_{i=1}^N \sim \mathbb{P}_X$ desde el conjunto de entrenamiento.
  Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
  Muestrear $\tilde{z}_i \sim \mathbb{Q}_\phi(\mathrm{d}z \mid x_i)$ para $i = 1 \dots N$. ▷ $\tilde{z} = \mathbb{Q}_\phi(x)$ si $\mathbb{Q}_\phi$ es determinista.
  $\mathcal{L}_{\mathrm{AE}} \leftarrow \frac{1}{N}\sum_{i=1}^N c(x_i, G_\theta(\tilde{z}_i)) + \lambda \cdot \mathrm{MMD}(z, \tilde{z})$.
  Actualizar $\mathbb{Q}_\phi$ y $G_\theta$ por medio de descenso de gradiente en $\frac{\partial}{\partial\phi}\mathcal{L}_{\mathrm{AE}}$ y $\frac{\partial}{\partial\theta}\mathcal{L}_{\mathrm{AE}}$.
**end while**

## 2.4. Baricentros de Wasserstein Restringidos

La interpolación de imágenes se refiere a la transición visual de dos imágenes de entrada. Para que aquella transición sea visualmente atractiva, es deseable que cumpla con las siguientes propiedades: (*i*) que sea suave, es decir, que no haya cambios bruscos en la transición; (*ii*) que aplique los mínimos cambios necesarios en la imagen; y (*iii*) que parezca "realista", evitando cambios no naturales en cada transición.

Varias de estas propiedades se pueden obtener a través de una interpolación geodésica con baricentros de Wasserstein. Sin embargo, el resultado de esta interpolación puede no verse natural. En base al trabajo de [67], que describe la manera en que una GAN aproxima una variedad de imágenes, y cómo se puede utilizar un AE para proyectar una imagen en esta variedad, los autores de [61] proponen utilizar una GAN y un AE para mantener la suavidad y realismo de la interpolación.

En la Figura 2.4 se observa un ejemplo donde se compara la interpolación de baricentros de Wasserstein con el método descrito en [61]. En la primera fila se muestra la interpolación de una zapatilla deportiva a una bota utilizando baricentros de Wasserstein, notando que esta es una interpolación difusa; en la segunda fila, se interpola en el espacio latente de una GAN, donde no es una transición suave; por último, en la tercera fila se muestra la interpolación utilizando el método propuesto, donde se puede notar que la transición es suave y realista.

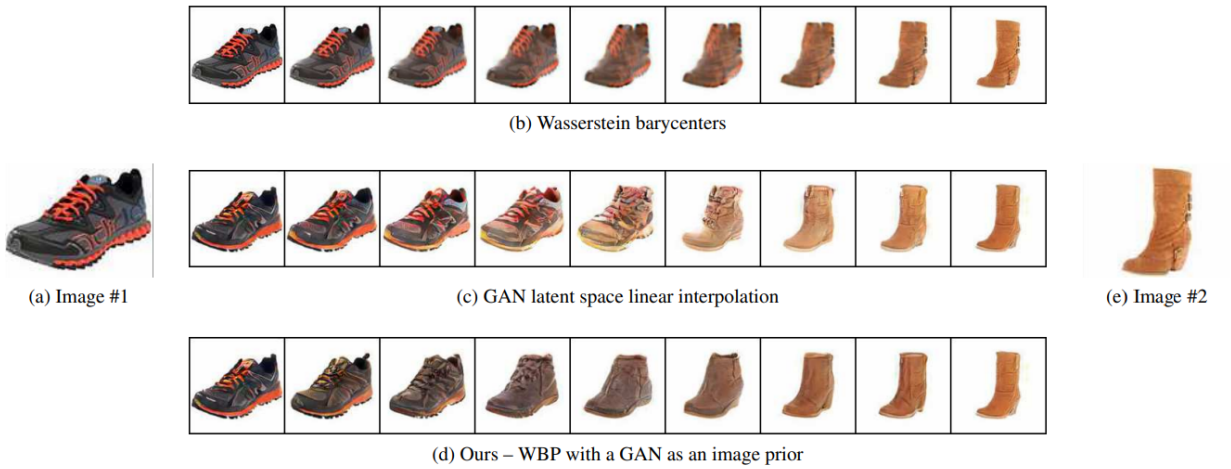



Figura 2.4: Interpolación de una zapatilla deportiva a una bota utilizando 3 métodos. En la Figura 2.4(b) las imágenes intermedias se ven borrosas y poco realistas. En la Figura 2.4(c) la zapatilla a penas cambia y entonces inmediatamente cambia a una bota. Por último, en la Figura 2.4(d) la interpolación es suave en la transición del color y de forma. Imagen obtenida de [61].

En [61], se empieza asumiendo que se trabajará con imágenes, y por tanto, el espacio $\mathcal{X}$ en que se trabaja corresponde a uno de $n_1 \times n_2 = N$ píxeles, donde las medidas $\mu_0, \mu_1 \in \mathcal{P}(\mathcal{X})$ se encuentran definidas:

$$\mu_0 = \sum_{i=1}^{N} p_i^{(0)} \delta_{x_i}, \qquad \text{y} \qquad \mu_1 = \sum_{i=1}^{N} p_i^{(1)} \delta_{x_i}.$$

Por tanto, basta con considerar a los vectores $p^{(0)}$ y $p^{(1)}$ como vectores de probabilidad en el simplex $\Sigma_N \overset{\text{def}}{=} \{p \in [0,1]^N : \sum_i p_i = 1\}$.

De esta forma, los autores de [61] proponen el siguiente problema de optimización:

$$\min_{q \in \Sigma_N} \quad (1-t) \cdot W_2(\mu_0, \nu)^2 + t \cdot W_2(\nu, \mu_1)^2 \tag{$P_{\text{CWB}}$}$$

$$\text{s.a.} \quad \nu = \sum_{i=1}^{N} q_i \delta_{x_i} \in \mathcal{M}, \tag{2.23}$$

donde se puede notar que este problema es similar a la definición de una interpolación geodésica (1.17), pero con la diferencia de que se restringe a que el baricentro $\nu$ pertenezca a la variedad $\mathcal{M}$. Por esta restricción, los autores llaman a este problema como *Baricentros de Wasserstein Restringidos* (CBW por sus siglas en inglés).

A partir del Problema ($P_{\text{CWB}}$), los autores imponen una restricción de igualdad en la variable $\nu$, de forma que $\tilde{\nu} = \nu$ con $\tilde{\nu} \in \mathcal{M}$. Usando este igualdad, se puede reescribir el

Problema ($P_{\text{CWB}}$) utilizando la técnica del Lagrangiano Aumentado [10, Sec. 2.3]:

$$\min_{q\in\Sigma_N,\tilde{q},u} \quad (1-t)\cdot W_2(\mu_0,\nu)^2 + t\cdot W_2(\nu,\mu_1)^2 + \frac{\rho}{2}\|\tilde{q}-q+u\|_2^2 \qquad (P_{\text{CWB}}^{\text{AL}})$$

$$\text{s.a.} \quad \tilde{\nu} = \sum_{i=1}^{N} \tilde{q}_i \delta_{x_i} \in \mathcal{M}\,. \qquad (2.24)$$

El Problema ($P_{\text{CWB}}^{\text{AL}}$) se puede resolver usando el Método de los Multiplicadores de Dirección Alterna (ADMM) [10, Cap. 3] el cuál se presenta a continuación:

**Algoritmo 2.7** Baricentros de Wasserstein Restringidos [61]

**Require:** Vectores de probabilidad $p^{(0)}, p^{(1)} \in \Sigma_N$, suposición inicial $q^{(0)}$, parámetro $t \in [0,1]$, parámetro de penalización $\rho > 0$ y umbral $\varepsilon > 0$.

1: Inicializar $u^{(0)} \leftarrow 0$, $\tilde{q}^{(0)} \leftarrow q^{(0)}$, $k \leftarrow 0$
2: **repeat**
3: $\quad q^{(k+1)} \leftarrow \arg\min_{q\in\Sigma_N} (1-t)W_2(\mu_0,\nu)^2 + tW_2(\nu,\mu_1)^2 + \frac{\rho}{2}\|\tilde{q}^{(k)} - q + u^{(k)}\|_2^2$
4: $\quad \tilde{q}^{(k+1)} \leftarrow \arg\min_{\tilde{q}:\,\tilde{\nu}\in\mathcal{M}} \|\tilde{q} - q^{(k+1)} + u^{(k)}\|_2^2$
5: $\quad u^{(k+1)} \leftarrow u^{(k)} + \tilde{q}^{(k+1)} - q^{(k+1)}$
6: $\quad k \leftarrow k+1$
7: **until** $\|q^{(k)} - \tilde{q}^{(k)}\| < \varepsilon$
8: **return** $q^{(k)}$

Al principio, en la línea 3 del Algoritmo 2.7, se encuentra una solución de la versión regularizada del problema de la Ecuación (1.17) (los autores siguen [18] para resolver este problema). En la línea 4, se proyecta la solución a la variedad $\mathcal{M}$ y en la línea 5 se actualiza la variable dual $u$. Este proceso se repite hasta que se alcance la convergencia.

En la Figura 2.5 se ilustra la diferencia entre su método, comparado con los otros dos métodos.

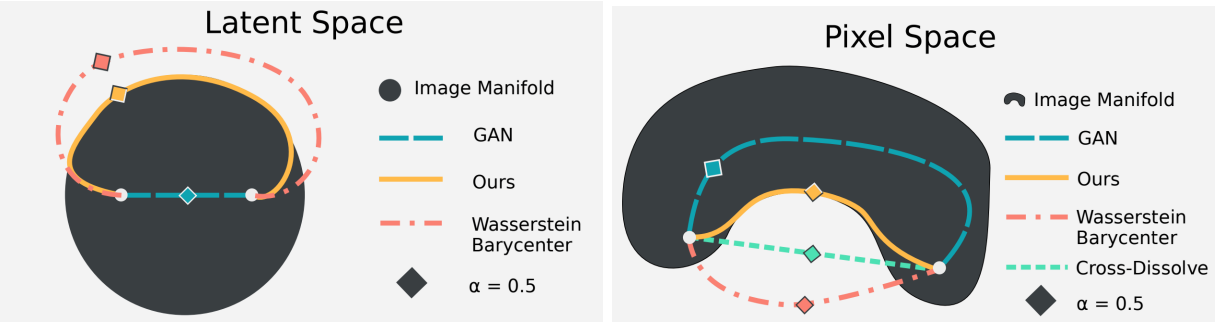



Figura 2.5: Ilustración del proceso de interpolación de una imagen, fijando $t = 0 \to 1$, usando varios métodos en el espacio de píxeles y en el espacio latente de una GAN entrenada. Imagen obtenida de [61].

# Capítulo 3

# Unificando la WGAN y el WAE

En capítulos anteriores se explica que, tanto las WGANs como las WAEs aproximan una distribución real $\mathbb{P}_X$, además permiten aproximar las variedades, como también proyectar sobre ellas. Por este motivo, en este trabajo se estima una distribución de imágenes $\mathbb{P}_X$ para tener acceso a un muestreo rápido de este, además, para que los baricentros sean naturales, se utiliza una estructura de AE para proyectar sobre la variedad de imágenes, siguiendo el trabajo de [61].

Con respecto a la estimación de la distribución real y la aproximación de variedades, la WGAN realiza estas dos tareas bastante bien, gracias a su componente adversaria. Sin embargo, este no posee un codificador (que es necesario para poder proyectar sobre la variedad de imágenes). Por el otro lado, el WAE posee este codificador, pero no es tan bueno generando muestras de la distribución real o aproximando la variedad de imágenes. Cada una de estas estructuras cumplen bien con una de las tareas, pero no están diseñadas para cumplir con ambas.

Por esta razón, se propone un modelo que unifique a la WGAN y al WAE, de manera que se pueda estimar la distribución real y proyectar sobre la variedad de imágenes. A continuación se detalla la estructura de este modelo.

## 3.1. Preparación del Conjunto de Datos

Para los experimentos, se utiliza el conjunto de datos *Quick, Draw*!, creada por Google [36]. Este conjunto de datos ha sido construido a través de un juego en línea, donde a los jugadores se les solicita dibujar objetos pertenecientes a una clase particular de objetos, en menos de 20 segundos. El conjunto de datos contiene 50 millones de dibujos en escala de grises divididos en cientos de tipos de clases.

Para la preparación del conjunto de datos, se ha desarrollado la librería *Quick, Torch!* [45] para obtener fácilmente los datos utilizando la API de Google.[1] En los experimentos se

[1]El repositorio con el código fuente se encuentra en `https://github.com/framunoz/quick-torch`

utilizan específicamente las categorías “Faces” y “Smiley Faces”, dado que poseen imágenes bastante similares. Sin embargo, en ambos conjuntos existen imágenes anómalas, además de que en la categoría “Smiley Faces” existe una mayor diversidad de imágenes. Por este motivo, es necesario realizar un tratamiento del conjunto de datos antes de utilizarlo. Un ejemplo de las imágenes de estas categorías se muestra en la Figura 3.1.

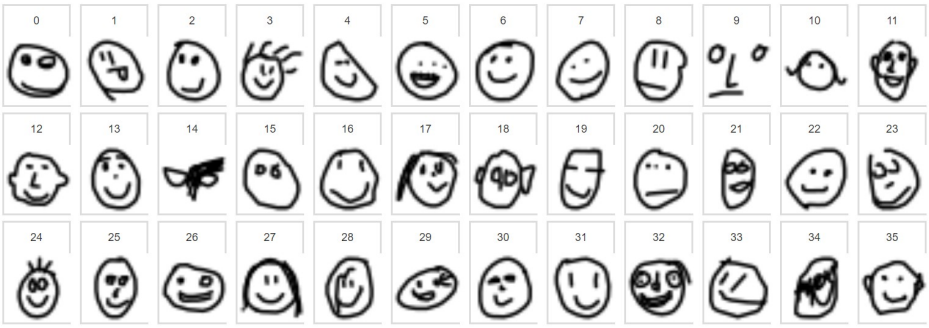
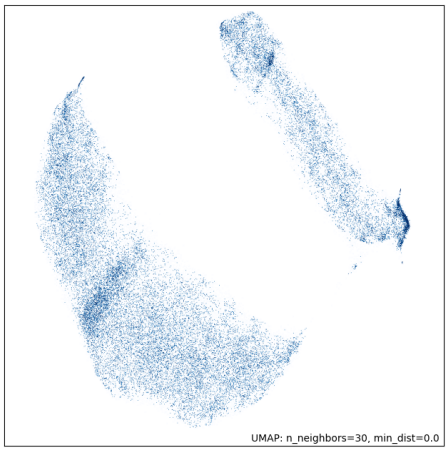


Figura 3.1: Ejemplo de imágenes de las categorías “Faces” y “Smiley Faces”.

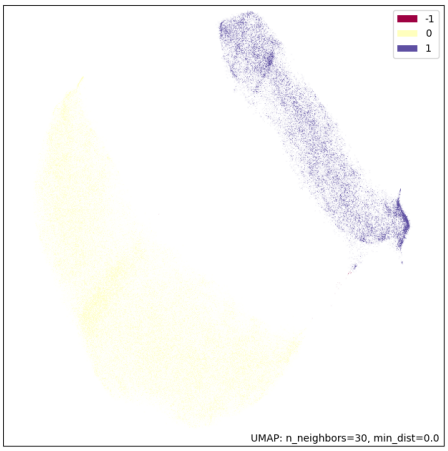
Para la preparación del conjunto de datos, se han utilizado tanto técnicas de agrupamiento (también conocido como *clustering*), como de reducción de dimensionalidad no lineal. Iterativamente se reduce la dimensionalidad a través de la técnica llamada Aproximación y Proyección de Variedades Uniformes (UMAP, por sus siglas en inglés) [42], para después agrupar o limpiar los datos anómalos sobre dicha agrupación. Para agrupar los datos, se utilizan las técnicas de Agrupación Espacial de Aplicaciones con Ruido Basada en la Densidad (DBSCAN, por sus siglas en inglés) [21], o su variante jerárquica (HDBSCAN, por sus siglas en inglés) [15]; mientras que para limpiar los datos anómalos, se usa el Factor Atípico Local (LOF, por sus siglas en inglés) [12]. Después del tratamiento, en el conjunto de datos quedan 230K imágenes en total. Las técnicas de agrupación y limpieza fueron utilizadas mediante la librería de *scikit-learn* [53].

A continuación se ilustran algunas imágenes en las que se utilizan las técnicas mencionadas. En la Figura 3.2a se puede observar la proyección UMAP en dos dimensiones del conjunto de datos, donde claramente se notan dos agrupaciones. En la Figura 3.2b se presentan las agrupaciones utilizando DBSCAN. Por último, en la Figura 3.3 se presentan las imágenes correspondientes a cada etiqueta de la Figura 3.3b. Se puede apreciar que en estas figuras se observa una tendencia clara en la forma en que se agrupan las imágenes.

Se redimensionan las imágenes de $28 \times 28$ a $32 \times 32$ píxeles y se normalizan para mejorar el entrenamiento de las redes neuronales. Se realizan diversas técnicas de aumento de datos (también conocido como *data augmentation*), tales como: volteo horizontal aleatorio (o *random horizontal flip*) con una probabilidad del 0,5; alejamiento aleatorio (o *random zoom out*) con una probabilidad del 0,3 con un aumento que distribuye como una uniforme de rango $[1; 1{,}25]$ y rotación aleatoria (o *random rotation*) con un ángulo aleatorio entre $[-10, 10]$.

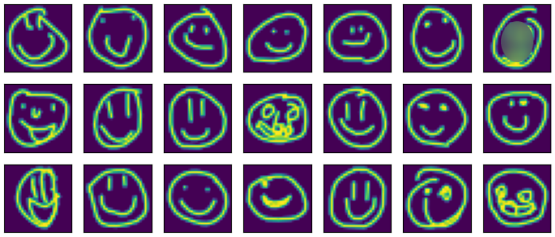



(a) Proyección UMAP del conjunto de datos.

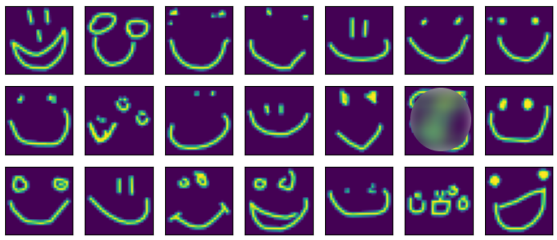



(b) Proyección UMAP agrupada.

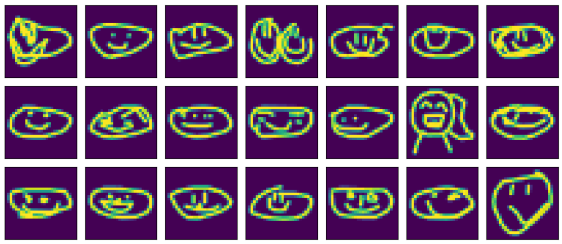
Figura 3.2: Visualización de la proyección UMAP y su agrupación.



(a) Imágenes de la etiqueta 0.



(b) Imágenes de la etiqueta 1.



(c) Datos anómalos.

Figura 3.3: Imágenes correspondiente a los etiquetados en la Figura 3.2b.

## 3.2. Deducción de la arquitectura

Como se menciona en los capítulos anteriores, la WGAN y el WAE tienen tareas similares: buscan minimizar una estimación de la distancia de Wasserstein entre una distribución de referencia $\mathbb{P}_X$ y una distribución generadora $\mathbb{P}_G$. En particular, la WGAN lo realiza utilizando la 1-distancia de Wasserstein, mientras que el WAE lo puede hacer con cualquier distancia definida a través de la función de costo $c$.

Por este motivo, la deducción del algoritmo es simple: dejar que la WGAN entrene la generadora y la función crítica, de manera que la generadora ya esté bastante cerca de la distribución real. Luego, se puede utilizar la generadora de la WGAN como decodificadora del WAE, y entrenar esta última arquitectura (donde la generadora ya está más cerca de parecerse a la distribución real) para que el codificador sea capaz de proyectar sobre la variedad de imágenes. Es importante que la función de costo del WAE sea $c(x, y) = |x - y|$ para que se esté calculando la 1-distancia de Wasserstein, y la distribución de la decodificadora converja en la misma topología que la WGAN.

Por este motivo, el algoritmo propuesto es el siguiente:

**Algoritmo 3.1** Entrenamiento de una WAE-WGAN, elaboración propia

**Require:** Tamaño del batch $N$, número de iteraciones para el discriminador $N_d$ y los parámetros de penalización $\lambda_{\text{WGAN}}$ y $\lambda_{\text{WAE}}$.

Inicializar los parámetros de la generadora $G_\theta$ y la función crítica $f_\omega$.
**while** $\theta$ no ha convergido **do**
    // Entrenamiento de la función crítica
    **for** $t = 1, \dots, N_d$ **do**
        Muestrear $\{x_i\}_{i=1}^N \sim \mathbb{P}_X$ desde el conjunto de entrenamiento.
        Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
        $\tilde{x} \leftarrow (G_\theta(z_i))_{i=1}^N$.
        $\mathcal{L}_{\text{critic}} \leftarrow \frac{1}{N}\sum_{i=1}^N f_\omega(\tilde{x}_i) - \frac{1}{N}\sum_{i=1}^N f_\omega(x_i) + \lambda_{\text{WGAN}} \cdot \text{PENALTY}(\omega, x, \tilde{x})$
        Actualizar $f_\omega$ por medio de descenso de gradiente en $\frac{\partial}{\partial \omega}\mathcal{L}_{\text{critic}}$.
    **end for**
    // Entrenamiento de la generadora por parte de la WGAN
    Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
    $\mathcal{L}_{\text{gen}} \leftarrow -\frac{1}{N}\sum_{i=1}^N f_\omega(G_\theta(z_i))$
    Actualizar $G_\theta$ por medio de descenso de gradiente en $\frac{\partial}{\partial \theta}\mathcal{L}_{\text{gen}}$.
    // Entrenamiento de la WAE
    Muestrear $\{z_i\}_{i=1}^N \sim \mathbb{P}_Z$ desde el espacio latente.
    Muestrear $\tilde{z}_i \sim \mathbb{Q}_\phi(\mathrm{d}z \mid x_i)$ para $i = 1 \dots N$.
    Obtener $\tilde{x}_i \leftarrow G_\theta(\tilde{z}_i)$ para $i = 1 \dots N$.
    $\mathcal{L}_{\text{AE}} \leftarrow \frac{1}{N}\sum_{i=1}^N |x_i - \tilde{x}_i| + \lambda_{\text{WAE}} \cdot \text{SIMILARITY}(z, \tilde{z})$
    Actualizar $\mathbb{Q}_\phi$ y $G_\theta$ por medio de descenso de gradiente en $\frac{\partial}{\partial \phi}\mathcal{L}_{\text{AE}}$ y $\frac{\partial}{\partial \theta}\mathcal{L}_{\text{AE}}$.
**end while**

Donde la función $\text{PENALTY}(\omega, x, \tilde{x})$ es alguna penalización para asegurar que $f_\omega \in \text{Lip}_1(\mathcal{X})$, y $\text{SIMILARITY}(z, \tilde{z})$ es alguna función de similitud entre densidades (como MMD o alguna

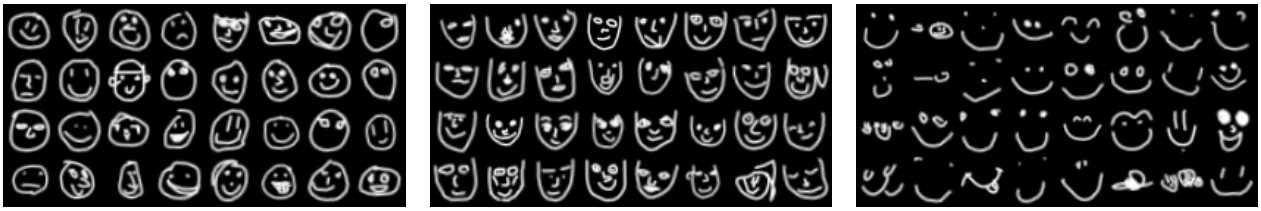
otra divergencia o distancia) entre las distribuciones $\mathbb{P}_Z$ y $\mathbb{Q}_{Z,\phi} \stackrel{\text{def}}{=} \int_{\mathcal{X}} \mathbb{Q}_\phi(\bullet \mid x)\, \mathbb{P}_X(\mathrm{d}x)$. Definir el algoritmo de esta manera permite tener una mayor generalidad a la hora de utilizarlo, y poder experimentar con distintas funciones de penalización y similitud.

Notar que el Algoritmo 3.1 realiza alternadamente tres procesos. Primero, un entrenamiento de la función crítica. Luego, un entrenamiento de la generadora, utilizando una estimación de la distancia de Wasserstein en su versión de la WGAN. Finalmente, un entrenamiento del WAE, el que entrena simultáneamente a la codificadora y generadora. Para estos efectos, la generadora se entrena con el gradiente de la distancia de Wasserstein por ambos algoritmos en una sola iteración.

Además, cuando la función crítica y la decodificadora alcanzan un máximo y mínimo aceptable, respectivamente, entonces la evaluación de estas redes en las funciones de pérdida provee una estimación de la distancia de Wasserstein para ambos casos. Esto resulta útil para monitorear el entrenamiento de la red.

### Implementación

El Algoritmo 3.1 se implementa en el repositorio [46][2] utilizando la librería de *PyTorch* [51] a través de las redes neuronales convolucionales (CNN por sus siglas en inglés) [39], siguiendo la estrategia de ResNet [32]. La red neuronal se entrena en tres variedades de imágenes diferentes, las cuales han sido obtenidas siguiendo los pasos explicados en la Sección 3.1. Algunas muestras de estas variedades se presentan en la Figura 3.4. Se separa el conjunto de datos en un conjunto de entrenamiento y otro de validación, con un 95 % y 5 % de las imágenes, respectivamente.



(a) 1era variedad de imágenes. (b) 2da variedad de imágenes. (c) 3era variedad de imágenes.

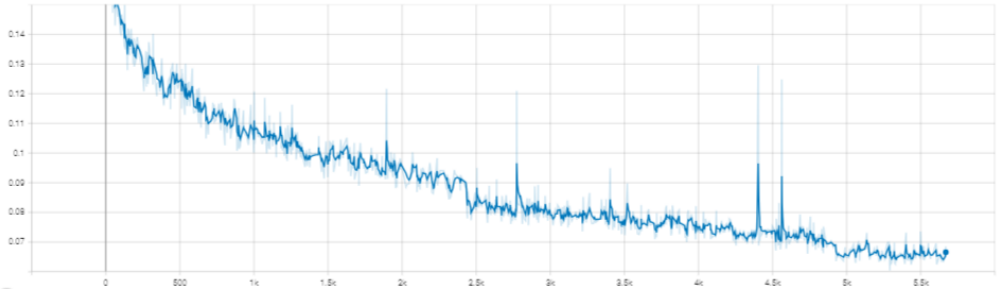
Figura 3.4: Variedades de imágenes utilizadas en los experimentos.

A lo largo del entrenamiento se han registrado las estimaciones de la distancia de Wasserstein para la WGAN y el WAE, sobre el conjunto de validación para analizar la convergencia. En la Figura 3.5 se presentan dichas estimaciones. Se puede observar que ambas estimaciones disminuyen a lo largo del entrenamiento, lo que corrobora que el algoritmo propuesto es estable, y además, indica que las redes están aprendiendo a aproximar la distancia de Wasserstein correctamente.

Para validar que las redes generativas están aprendiendo a reconstruir y generar imágenes de calidad, se presentan en la Figura 3.6 las imágenes reales, las decodificadas y las generadas

[2]El repositorio con el código fuente se encuentra en `https://github.com/framunoz/wgan-gp`

(a) Estimación de la distancia de Wasserstein para la WGAN.



(b) Estimación de la distancia de Wasserstein para el WAE.

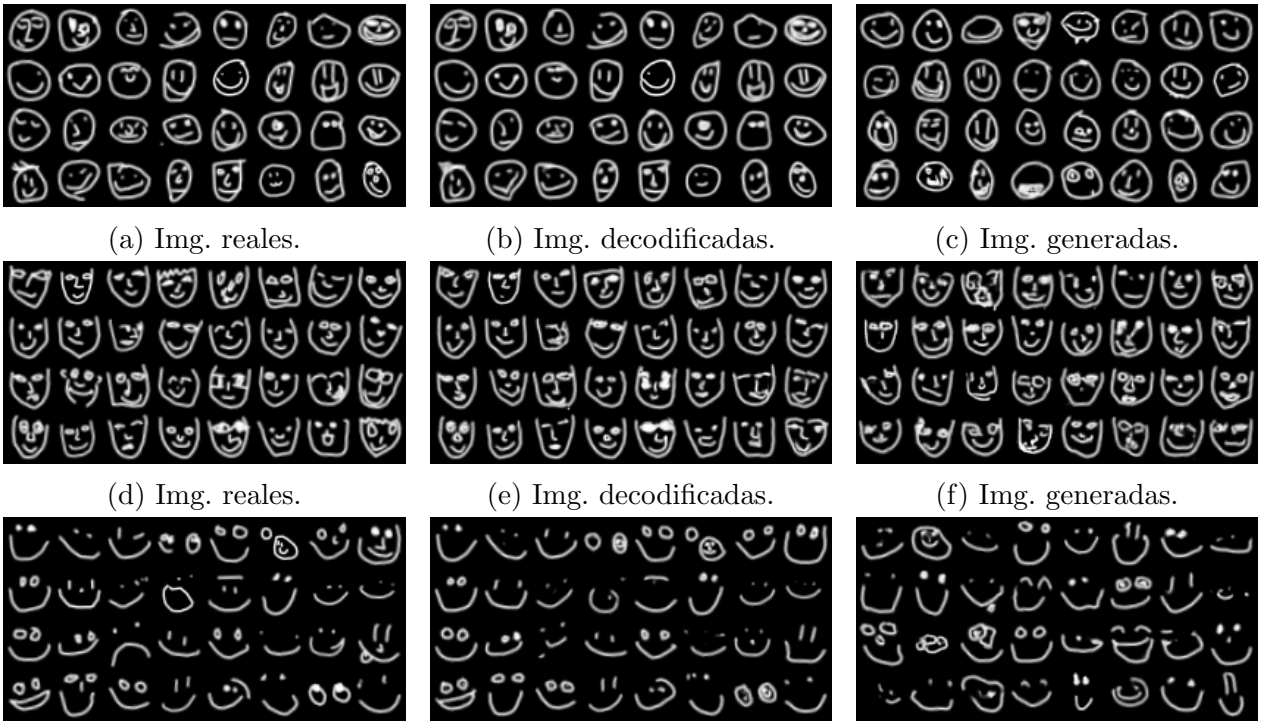
Figura 3.5: Estimación de la distancia de Wasserstein para la WGAN y el WAE.

por la WAE-WGAN.

Se puede observar que las imágenes generadas por la WAE-WGAN son bastante similares a las imágenes reales, lo que indica que las redes están aprendiendo a aproximar la distribución real. Además, las imágenes decodificadas por el WAE son bastante similares a las imágenes originales, lo que muestra que la red está aprendiendo a proyectar sobre la variedad de imágenes simultáneamente.

## 3.3. Conclusiones

A vista de los resultados, se puede concluir que la WAE-WGAN es capaz de generar imágenes de calidad y a la vez entrenar un codificador, de manera que se pueda proyectar sobre la variedad de imágenes. Además, se destaca que la WAE-WGAN es capaz de aproximar la distancia de Wasserstein, lo que indica que la red está aprendiendo a aproximar la distribución real.



(a) Img. reales. (b) Img. decodificadas. (c) Img. generadas.

(d) Img. reales. (e) Img. decodificadas. (f) Img. generadas.

(g) Img. reales. (h) Img. decodificadas. (i) Img. generadas.

Figura 3.6: Imágenes reales, decodificadas y generadas por la WGAN y el WAE en las tres variedades de imágenes. Cada fila corresponde a una variedad de imágenes.

# Capítulo 4

# Aplicaciones a los Baricentros de Wasserstein

En este capítulo se presenta el tema principal de la tesis: la implementación del Descenso del Gradiente Estocástico en el Espacio de Wasserstein (SGDW), y su aplicación a la estimación del Baricentro de Wasserstein Bayesiano (BWB). A lo largo de todo el capítulo, se presentan los resultados obtenidos a partir de la librería desarrollada en el repositorio [44].

Para ello, se empieza explicando la implementación del SGDW, y se realizan experimentos con algunos muestreadores de distribuciones. Luego, se propone una adaptación del SGDW, donde se proyecta el baricentro sobre alguna variedad $\mathcal{M}$ de medidas para que se vea más natural y atractivo. Finalmente, se utilizan estas implementaciones para el cálculo del BWB, donde se propone una técnica para poder estimarlo utilizando una GAN como prior.

## 4.1. Implementación del SGDW

En esta sección se presenta la implementación del SGDW. Para ello, se empieza explicando cómo se interpreta una imagen como una medida. Luego, se presentan las ligeras modificaciones al algoritmo original, para que se pueda trabajar con medidas discretas. Posteriormente, se presentan los experimentos realizados con distintas medidas, y se discuten los resultados obtenidos.

### 4.1.1. Interpretación de una Imagen como Medida

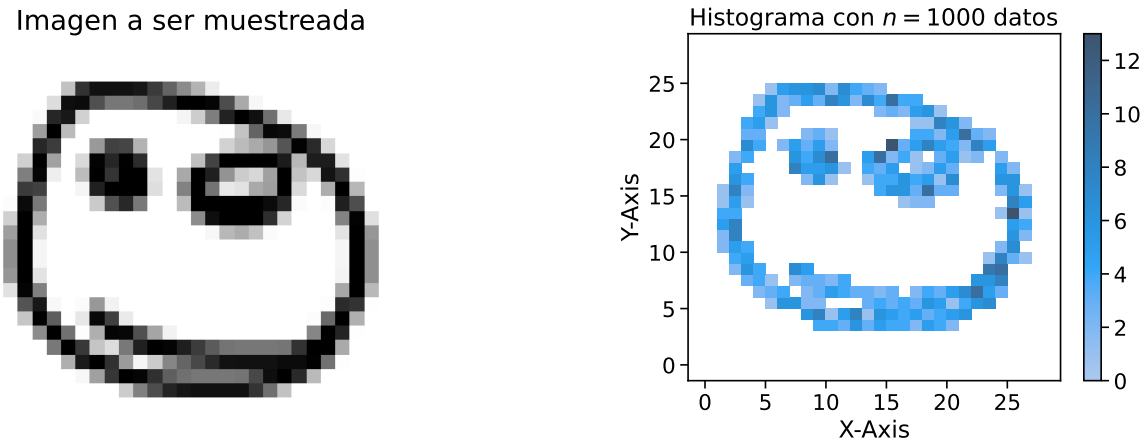
Recordar que si $\mu \in \mathcal{P}(\mathcal{X})$ es una medida discreta, entonces esta queda definida de la siguiente forma:

$$\mu = \sum_{i=1}^{n} m_i \delta_{x_i}, \tag{4.1}$$

donde $m = (m_1, \ldots, m_n) \in \Sigma_n$ es un vector de probabilidad en el Simplex y $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ son sus respectivas posiciones.

En el caso de una imagen, esta se puede interpretar como una medida discreta, si se define $\mathcal{X} \stackrel{\text{def}}{=} \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^2$ como las posiciones de los píxeles, con $n \stackrel{\text{def}}{=} n_1 \times n_2$; y $m \in \Sigma_n$ como el vector de probabilidad representando las intensidades de los píxeles. En este caso, si se asume que el espacio de imágenes es el espacio de probabilidad $\mathcal{P}(\mathcal{X})$, entonces las distribuciones que muestrean imágenes corresponden a distribuciones en el espacio $\mathcal{P}(\mathcal{P}(\mathcal{X}))$.

Las distribuciones que provienen de una imagen, se implementan de manera eficiente, tanto en tiempo como en memoria, utilizando su matriz de escala de grises. En la Figura 4.1 se muestra un ejemplo de una distribución que proviene de una imagen, donde se presenta la imagen y su histograma para $n = 1000$ muestras. Cabe recordar que la obtención de las imagenes se presentan en la Sección 3.1.




(a) Imagen de una cara.

(b) Histograma de $n = 1000$ muestras.

Figura 4.1: Ejemplo de una distribución que proviene de una imagen. Elaboración propia.

### 4.1.2. Implementación del Algoritmo

Dado que el Algoritmo 1.1 está diseñado para medidas absolutamente continuas, se reinterpreta este algoritmo para que se pueda trabajar con medidas discretas. Para ello, se destaca que la Definición 1.3.10 se puede reinterpretar como la $\eta_k$-interpolación geodésica entre las medidas $\mu_k$ y $\tilde{\mu}_k$, mientras que la Definición 1.3.13 se puede reinterpretar como el baricentro de las medidas $\left(\mu_k, \tilde{\mu}_k^{(1)}, \ldots, \tilde{\mu}_k^{(S_k)}\right)$ con pesos $\left(1 - \eta_k, \frac{\eta_k}{S_k}, \ldots, \frac{\eta_k}{S_k}\right) \in \Sigma_{S_k+1}$. De este modo, el Algoritmo 1.1 se extiende de la siguiente manera:

**Algoritmo 4.1** SGDW General

**Require:** Acceso a las muestras de $\Gamma(\mathrm{d}\mu) \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$, un esquema de paso $(\eta_k)_k \in [0,1]^{\mathbb{N}}$ y un esquema de paso $(S_k)_k \in \mathbb{N}^{\mathbb{N}}$.

$k \leftarrow 0$
Muestrear $\mu_0 \sim \Gamma$
**repeat**
    Muestrear $\tilde{\mu}_k^{(1)}, \ldots, \tilde{\mu}_k^{(S_k)} \overset{\text{iid}}{\sim} \Gamma$
    $\gamma \leftarrow \left(1 - \eta_k, \frac{\eta_k}{S_k}, \ldots, \frac{\eta_k}{S_k}\right)$
    Definir $\mu_k$ como el baricentro de $\left(\mu_k, \tilde{\mu}_k^{(1)}, \ldots, \tilde{\mu}_k^{(S_k)}\right)$ con pesos $\gamma$.
    $k \leftarrow k + 1$
**until** un criterio de detención ha sido alcanzado.
**return** $\mu_k$

Cabe destacar que en el Algoritmo 4.1 se mantiene la versión de la secuencia por lotes. Esto es para mantener una mayor generalidad, pues el caso original del Algoritmo 1.1 se recupera considerando $S_k = 1,\ \forall k \in \mathbb{N}$.

Como se está trabajando con imágenes, se puede aprovechar su estructura para calcular una estimación de los baricentros de manera más eficiente, utilizando el algoritmo de Baricentros de Wasserstein Convolucionales [62] o su versión Insesgada (*Debiased* en inglés) [35]. Sin embargo, el Algoritmo 4.1 es lo suficientemente general para ser aplicado a cualquier medida discreta, utilizando por ejemplo, la estimación del algoritmo de Sinkhorn [17].

Todos estos métodos de cálculo de baricentros se implementan de manera eficiente utilizando la librería de *Python Optimal Transport* (POT) [22], donde además esta librería admite la paralelización de los cálculos por medio de la computación de Propósito General en Unidades de Procesamiento Gráfico (GPGPU por sus siglas en inglés) [49]. En la implementación, se mantienen las versiones tanto para imágenes, como para medidas discretas genéricas, en función de obtener una mayor generalidad y versatilidad.

### Medidas de Probabilidad

Se considera una medida de referencia $\mathbb{P}_X \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$ que corresponde a una forma "idealizada" del conjunto de datos *Quick, Draw!* [36]. Esto es, si es que se tuviera un acceso ilimitado a muestras parecidas a este conjunto de datos, entonces este sería $\mathbb{P}_X$. Se desea calcular el baricentro de esta medida, utilizando el Algoritmo 4.1.

Para ello, se consideran dos aproximaciones de $\mathbb{P}_X$. La primera, es la empírica:

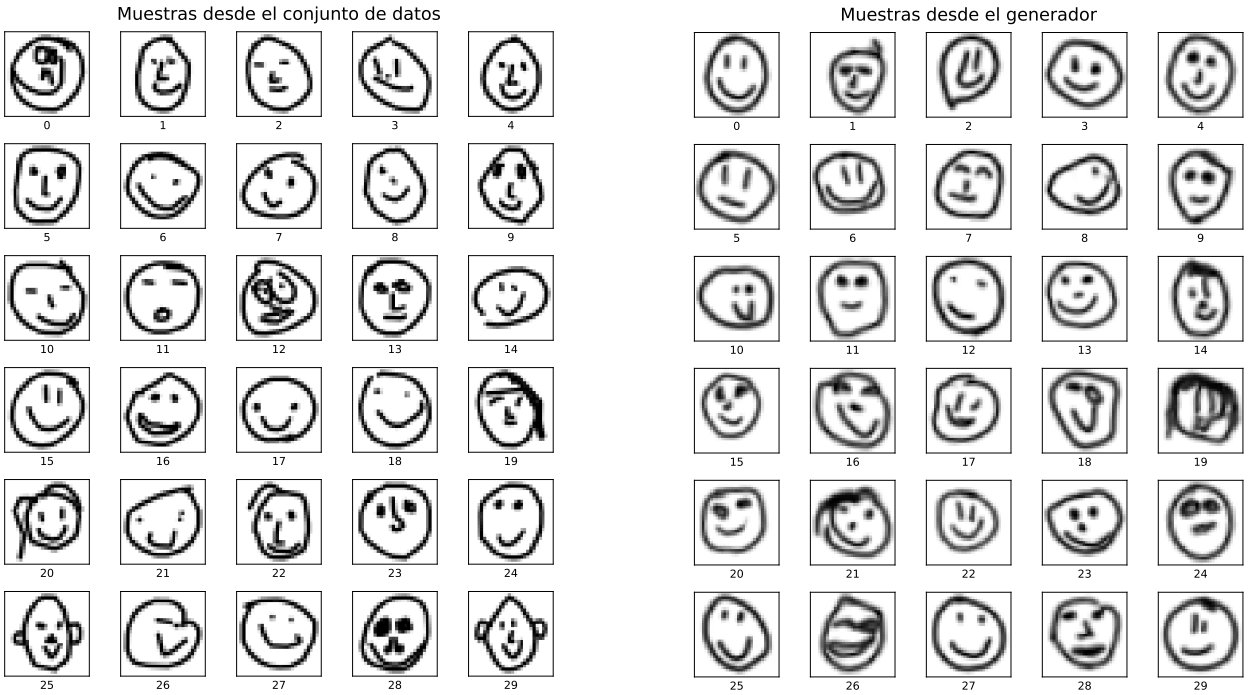
$$\hat{\mathbb{P}}_X = \frac{1}{N} \sum_{i=1}^{N} \delta_{\mu_i}, \tag{4.2}$$

donde $\{\mu_i\}_{i=1}^N \subseteq \mathcal{P}(\mathcal{X})$ es el conjunto de datos obtenido de la Sección 3.1, y $N$ es el tamaño del conjunto de datos. La segunda, es la medida generada por la WGAN $G_\theta : \mathcal{Z} \to \mathcal{P}(\mathcal{X})$,

entrenada como se explica en el Capítulo 3:

$$\tilde{\mathbb{P}}_X = G_{\theta\sharp}\,\mathbb{P}_Z \tag{4.3}$$

donde $\mathbb{P}_Z$ es la medida del espacio latente (en este caso, una distribución normal estándar). En la Figura 4.2a se presentan muestras de la medida empírica $\hat{\mathbb{P}}_X$ y en la Figura 4.2b de la medida generadora $\tilde{\mathbb{P}}_X$.




(a) Muestras desde la medida empírica $\hat{\mathbb{P}}_X$ (b) Muestras desde la medida generadora $\tilde{\mathbb{P}}_X$

Figura 4.2: Muestras de las medidas $\hat{\mathbb{P}}_X$ y $\tilde{\mathbb{P}}_X$ para el conjunto de datos *Quick, Draw.*

**Experimentos**

Como $\hat{\mathbb{P}}_X$ y $\tilde{\mathbb{P}}_X$ son estimaciones de la medida de referencia $\mathbb{P}_X$, se espera que el baricentro de población de estas medidas sean similares entre sí. Por este motivo, se calculan los baricentros utilizando el Algoritmo 4.1 para ambas medidas, y se comparan los resultados obtenidos.

Para el algoritmo, se utiliza el siguiente esquema de paso:

$$\eta_k = \frac{a}{(b^{1/c} + k)^c}, \quad \forall k \in \mathbb{N}. \tag{4.4}$$

Este esquema es una reparametrización del que se presenta en [66, Secc. 2]. Se puede demostrar que, cuando $a > 1$, $b > 0$ y $0{,}5 < c \leq 1$, entonces $(\eta_k)_k$ cumple con las siguientes condiciones:

$$\sum_{k=0}^{\infty} \eta_k = \infty, \qquad \sum_{k=0}^{\infty} \eta_k^2 < \infty. \tag{4.5}$$

Estas dos condiciones son usuales en la optimización convexa, pues garantizan la convergencia del algoritmo. En los experimentos se utilizan $a = 3$, $b = 3,01$ y $c = 0,501$. Se escogen estos parámetros para que se cumplan las siguientes condiciones:

$$\forall k \in \mathbb{N}\colon \eta_k \in [0,1] \qquad \text{y} \qquad \eta_0 = \frac{a}{b} \lesssim 1. \tag{4.6}$$

Además, para estas dos medidas, se consideran dos esquemas para el número de lotes $S_k$: uno simple $S_k = 1$, que correspondería al caso original, y otro más complejo $S_k = 5$, para observar el efecto de una convergencia más rápida.

Para la estimación del baricentro de Wasserstein en el paso 6 del Algoritmo 4.1, se utiliza el método del Baricentro de Wasserstein Convolucional Insesgado [35]. Para mayor detalle de los parámetros utilizados, se puede consultar el Anexo B.1.

Los resultados de las iteraciones se presentan a continuación:


(a) Baricentro de la medida empírica $\hat{\mathbb{P}}_X$.

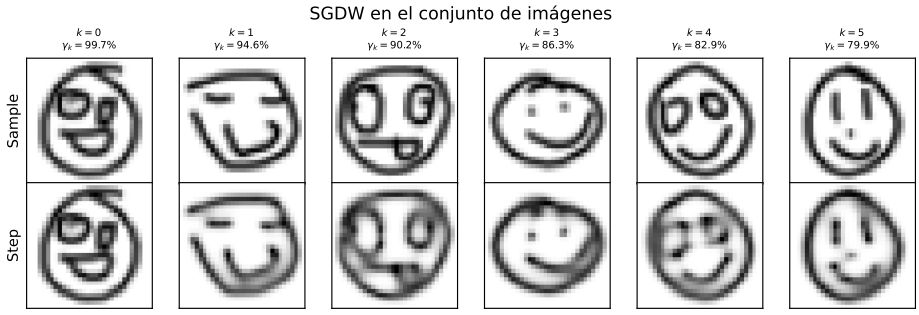

(b) Baricentro de la medida generadora $\tilde{\mathbb{P}}_X$.


(c) Baricentro de la medida empírica $\hat{\mathbb{P}}_X$, versión por lotes.

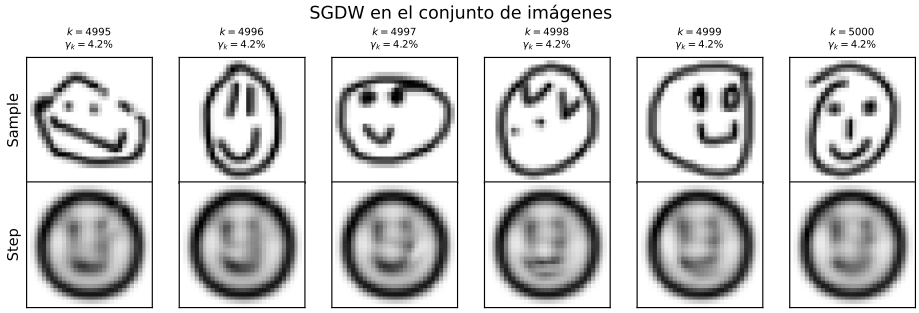

(d) Baricentro de la medida generadora $\tilde{\mathbb{P}}_X$, versión por lotes.

Figura 4.3: Baricentros de población de las medidas $\hat{\mathbb{P}}_X$ y $\tilde{\mathbb{P}}_X$ para el conjunto de datos *Quick, Draw*, utilizando $S_k = 1$ y $S_k = 5$.

En las Figuras 4.4, 4.5, 4.6 y 4.7 se presentan las primeras y últimas iteraciones para el cálculo de los baricentros anteriormente mencionados.

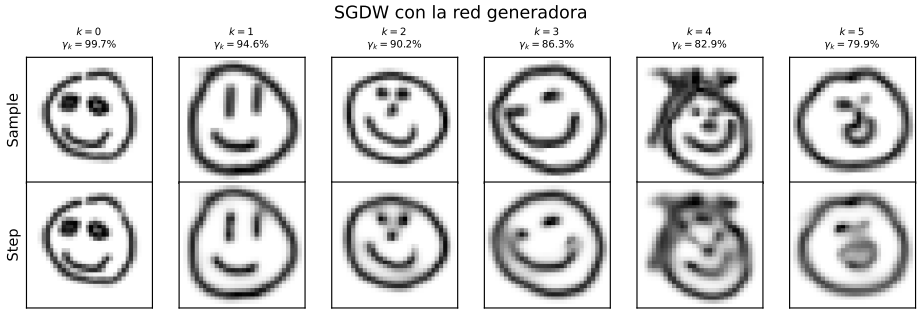



(a) Primeras iteraciones.

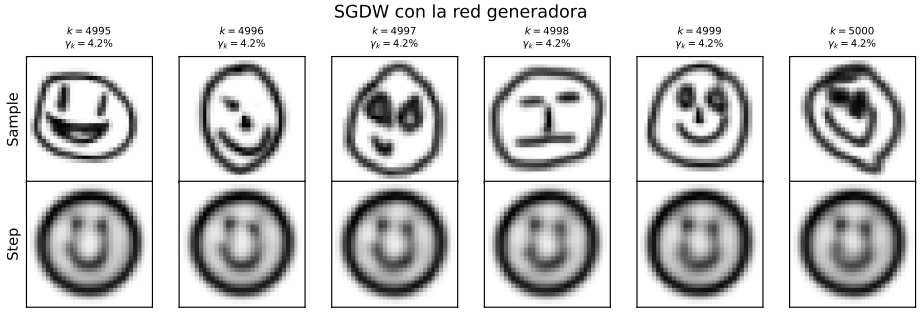



(b) Últimas iteraciones.

Figura 4.4: Iteraciones del cálculo del baricentro de $\hat{\mathbb{P}}_X$.

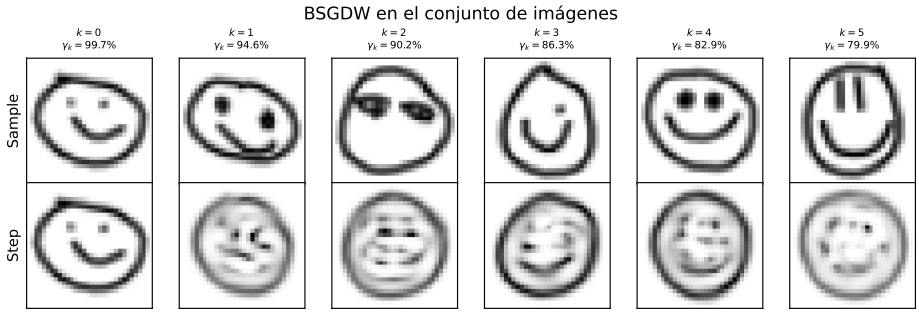



(a) Primeras iteraciones.

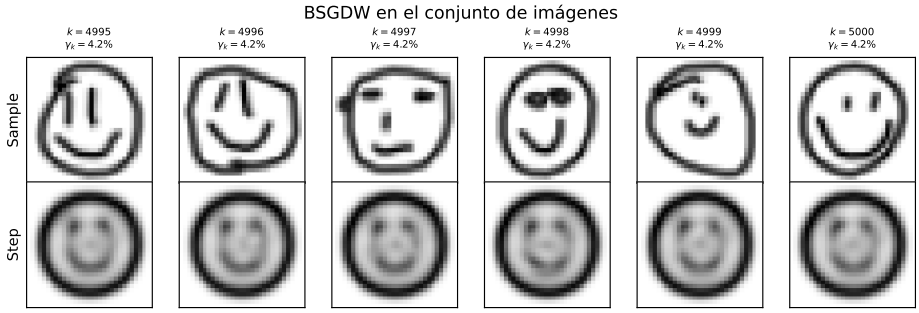



(b) Últimas iteraciones.

Figura 4.5: Iteraciones del cálculo del baricentro de $\tilde{\mathbb{P}}_X$.

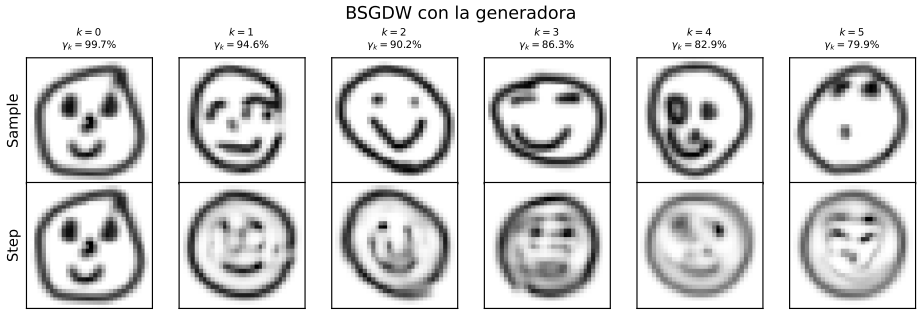



(a) Primeras iteraciones.

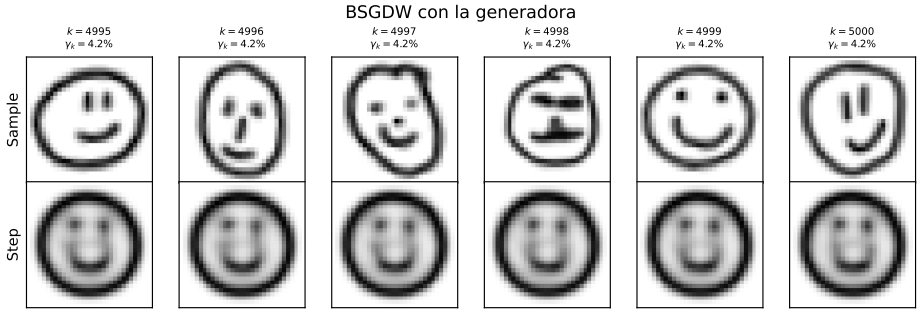



(b) Últimas iteraciones.

Figura 4.6: Iteraciones del cálculo del baricentro de $\hat{\mathbb{P}}_X$ en su versión por lotes.




(a) Primeras iteraciones.




(b) Últimas iteraciones.

Figura 4.7: Iteraciones del cálculo del baricentro de $\tilde{\mathbb{P}}_X$ en su versión por lotes.

Se puede notar cómo en las primeras iteraciones de las Figuras 4.4a y 4.5a, la estimación del baricentro varía mucho entre una iteración y otra, debido al esquema de paso utilizado. Sin

embargo, a medida que se avanza en las iteraciones, la estimación del baricentro se estabiliza. Además, se puede observar en las Figuras 4.6a y 4.7a que el método por lotes provee imágenes más difusas en las primeras iteraciones, debido a la mayor cantidad de muestras utilizadas en el cálculo del baricentro. No obstante, proporciona resultados muy similares a su versión simplificada en un menor tiempo de ejecución.

### 4.1.3. Conclusiones

Se puede observar que los baricentros obtenidos para los cuatro experimentos (los obtenidos en la Figura 4.3) son muy similares entre sí, comprobando la hipótesis inicial. Es lo esperable, pues tanto $\hat{\mathbb{P}}_X$ como $\tilde{\mathbb{P}}_X$ son aproximaciones de la medida de referencia $\mathbb{P}_X$. Además, a partir de las Figuras 4.6a y 4.7a se puede observar que efectivamente el método por lotes converge más rápido que el método original, resultando en baricentros muy similares a su versión simplificada.

## 4.2. SGDW Proyectado

A pesar de que la implementación del SGDW entrega los resultados esperados, el baricentro no parece ser lo suficientemente natural. Por este motivo, se propone una adaptación del SGDW, donde se busca proyectar el baricentro sobre alguna variedad $\mathcal{M}$ de medidas, con el objetivo de hacerlo ver más natural y atractivo.

Para ello, se recapitula la definición de CWB de la Sección 2.4, donde en el trabajo [61] se explica la forma de proyectar una medida $\mu$ sobre una variedad específica de medidas $\mathcal{M}$, aprendidas por una red generativa.

En este sentido, el conjunto de modelos $\mathcal{M} \subseteq \mathcal{P}(\mathcal{X})$ correspondería a la variedad generada por la red $G_\theta$, es decir,

$$\mathcal{M} \stackrel{\text{def}}{=} \{G_\theta(z) \in \mathcal{P}(\mathcal{X}) \colon z \in \mathcal{Z}\} \subseteq \mathcal{P}(\mathcal{X}). \tag{4.7}$$

Este cambio puede hacer que el Supuesto 1.3.8 no se cumpla, dado que es posible que $\mathcal{M}$ no sea geodésicamente convexo. Esto provocaría que el Teorema 1.3.12 y la Proposición 1.3.14 no garanticen la existencia o la unicidad del baricentro proyectado.

### 4.2.1. Integración de la Proyección al SGDW

A pesar de que los autores de [61] explican que, para obtener los resultados de su artículo utilizan el Algoritmo 2.7, la realidad es que al revisar su código fuente [60] se observa que sólo utilizan una iteración del algoritmo anterior (es decir, no utilizan el Lagrangiano Aumentado). De este modo, el algoritmo se simplifica de la siguiente manera: para dos medidas $\mu_0, \mu_1 \in \mathcal{P}(\mathcal{X})$ y un número $t \in [0, 1]$, empiezan por calcular la $t$-interpolación geodésica de estas medidas (paso 3 del Alg. 2.7), y luego proyectan este baricentro sobre la variedad $\mathcal{M}$ a través

del AE (paso 4 del Alg. 2.7). El resto de pasos que conciernen al Lagrangiano Aumentado (los pasos 5, 6 y el bucle 2-7) son omitidos en su implementación.

Motivados por esta simplificación, se propone una adaptación del Algoritmo 4.1 para que proyecte el baricentro en la variedad $\mathcal{M}$, donde además se agrega el parámetro $n_P \in \mathbb{N}$ para proyectar cada $n_P$ iteraciones del algoritmo. De este modo, se deduce el Algoritmo 4.2 para el SGDW Proyectado.

**Algoritmo 4.2** SGDW Proyectado (SGDWP)

**Require:** Acceso a las muestras de $\Gamma(\mathrm{d}\mu) \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$, un esquema de paso $(\eta_k)_k \in [0,1]^{\mathbb{N}}$, un esquema de paso $(S_k)_k \in \mathbb{N}^{\mathbb{N}}$, un proyector $P : \mathcal{P}(\mathcal{X}) \to \mathcal{M} \subseteq \mathcal{P}(\mathcal{X})$ y un número $n_P \in \mathbb{N}$.

1: $k \leftarrow 0$
2: Muestrear $\mu_0 \sim \Gamma$
3: **repeat**
4: Muestrear $\tilde{\mu}_k^{(1)}, \ldots, \tilde{\mu}_k^{(S_k)} \overset{\text{iid}}{\sim} \Gamma$
5: $\gamma \leftarrow \left(1 - \eta_k, \frac{\eta_k}{S_k}, \ldots, \frac{\eta_k}{S_k}\right)$
6: Definir $\mu_k$ como el baricentro de $\left(\mu_k, \tilde{\mu}_k^{(1)}, \ldots, \tilde{\mu}_k^{(S_k)}\right)$ con pesos $\gamma$.
7: **if** $k \mod n_P = 0$ **then**
8: $\mu_k \leftarrow P(\mu_k)$ ▷ Proyectar $\mu_k$ sobre $\mathcal{M}$
9: **end if**
10: $k \leftarrow k + 1$
11: **until** un criterio de detención ha sido alcanzado.
12: $\mu_k \leftarrow P(\mu_k)$ ▷ Terminar con una última proyección antes de retornar.
13: **return** $\mu_k$

Cabe destacar que el Algoritmo 4.2 tiene una mecánica similar al Método del Gradiente Proyectado (MPG) [2, Secc. 5.1], puesto que este es un método de optimización con restricciones que, si en una iteración se viola alguna restricción, entonces se proyecta dicha iteración sobre el conjunto de restricciones. Este es el caso del Algoritmo 4.2, donde se proyecta el baricentro en la variedad $\mathcal{M}$. Sin embargo, se diferencia del MPG, en que el dominio $\mathcal{M}$ puede no ser (geodésicamente) convexo.

### 4.2.2. Experimentos

Para validar el Algoritmo 4.2, se realizan los siguientes experimentos. Cabe recordar que los siguientes experimentos se realizan utilizando las redes neuronales presentadas en el Capítulo 3, donde en ese capítulo se explica el conjunto de datos utilizado y la deducción de las redes neuronales.

### Único Baricentro

Con este primer experimento se busca contestar a la siguiente pregunta:

¿Existe un único baricentro para este problema?

Como se sabe, y los experimentos de la Sección 4.1 corroboran, el baricentro que computa el SGDW es único y consistente. Sin embargo, la variedad $\mathcal{M}$ en este caso puede no ser geodésicamente convexa, por lo que no se puede garantizar la unicidad del baricentro proyectado. Por este motivo, para comprobar si existe alguna convergencia en el baricentro proyectado, se ejecuta el Algoritmo 4.2 múltiples veces para observar los distintos resultados, y verificar si son similares o no.

Para este experimento, se utiliza el mismo esquema $(\eta_k)_k$ de la Ecuación 4.4, usando los mismos parámetros mencionados. Para el esquema de lotes, se utiliza $S_k = 1,\ \forall k \in \mathbb{N}$. Para definir el proyector, se utilizan las redes neuronales expuestas en el Capítulo 3, de manera que el proyector se define por $P = \mathbb{Q}_\phi(G_\theta(\bullet))$. Además se fija $n_P = 1$ para proyectar en cada iteración del algoritmo.

El resultado de este experimento se puede observar en la Figura 4.8, donde se muestran los baricentros proyectados en diez experimentos distintos.

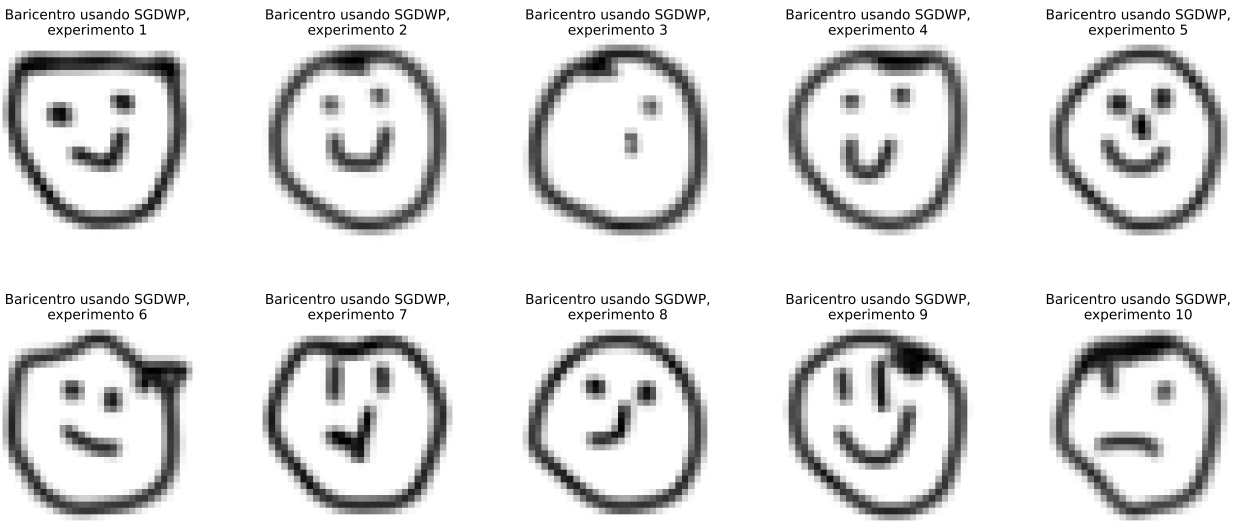



Figura 4.8: Baricentros proyectados obtenidos por el Algoritmo 4.2 en diez experimentos distintos.

Se puede observar que los baricentros proyectados son distintos entre sí, lo que sugiere que no existe un único baricentro proyectado para el problema planteado. Además, se puede observar que los baricentros obtenidos difieren bastante en la forma del baricentro clásico (Fig. 4.3). Una explicación a este fenómeno es debido a que se proyecta el baricentro en cada iteración del algoritmo, sin dar una ventana para que el baricentro se estabilice antes de proyectarlo.

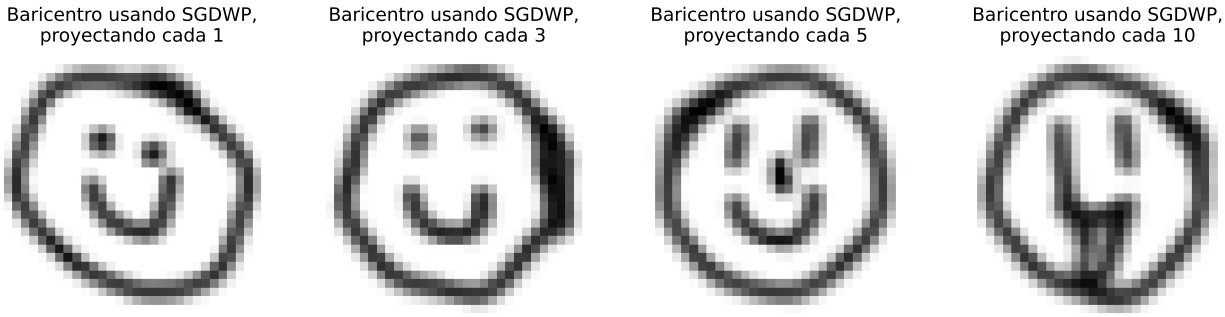
### Variando el Parámetro de Proyectar Cada $n_P$

En el segundo experimento, se busca contestar a la siguiente pregunta:

¿Cómo afecta el parámetro $n_P$ en la convergencia del baricentro proyectado?

La hipótesis plantea que la ejecución de las iteraciones del algoritmo "clásico", permitirán tener una mejor estimación del baricentro original antes de proyectarlo en la variedad $\mathcal{M}$. Para verificar esta hipótesis, se ejecuta el Algoritmo 4.2 con distintos valores de $n_P$ y se observa cómo afecta en la convergencia del baricentro proyectado.

En particular, se ejecutará con los valores $n_P \in \{1, 3, 5, 10\}$.




(a) $n_P = 1$ (b) $n_P = 3$ (c) $n_P = 5$ (d) $n_P = 10$

Figura 4.9: Baricentros proyectados obtenidos por el Alg. 4.2 con distintos valores de $n_P$.

En el Anexo C.1 se presentan las iteraciones por las que pasaron estos baricentros.

A partir de la Figura 4.9, se observa que utilizar otros valores de $n_P$, permite obtener una estimación del baricentro más similar al baricentro clásico (Fig. 4.3), pero con una mayor naturalidad. En este caso, se destaca que el baricentro de la Figura 4.9c es el que más se asemeja al baricentro original.

### Paseo Aleatorio

Dado que el baricentro proyectado no es único, se propone realizar un paseo aleatorio sobre la variedad $\mathcal{M}$ para explorar las distintas medidas que se encuentren en esta variedad. Para ello, se plantea un esquema de paso constante $\eta_k = 0{,}1$ donde $n_P = 1$. Se llevan a cabo 1000 iteraciones del Algoritmo 4.2, registrando una muestra cada 100. Estos resultados se pueden observar en la Figura 4.10.

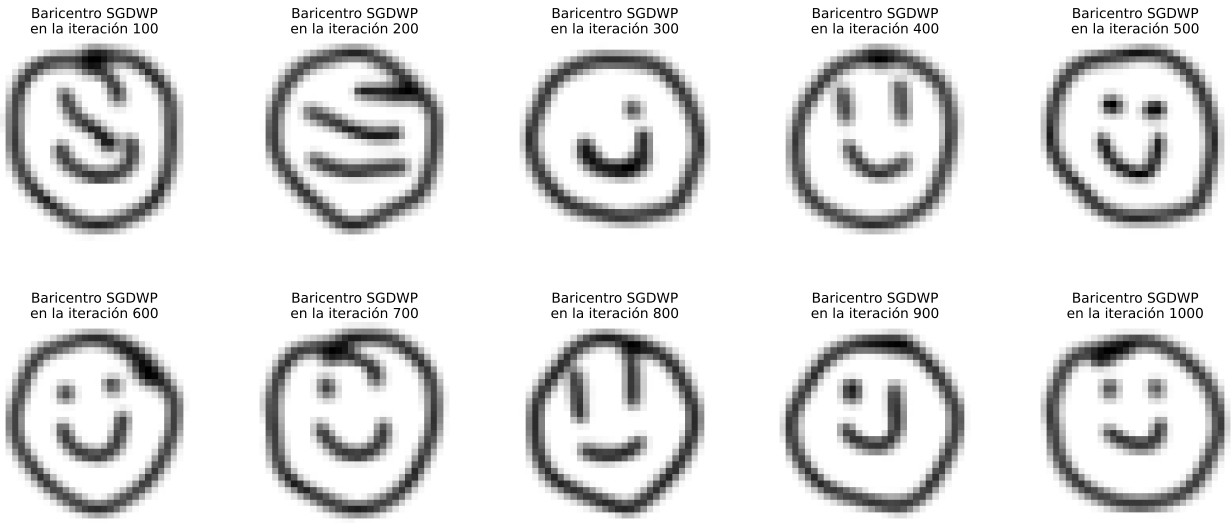



Figura 4.10: Paseo aleatorio en la variedad $\mathcal{M}$ obtenido por el Algoritmo 4.2.

A partir de la Figura 4.10, se observa que el paseo aleatorio permite explorar distintas medidas similares entre sí, que se encuentran en la variedad $\mathcal{M}$, evitando caer en un mínimo local. Esto puede ser útil para obtener distintas representaciones de la imagen que se desea obtener.

### 4.2.3. Conclusiones

En vista a los experimentos, se puede observar que el baricentro proyectado provee una representación más natural de la imagen que se desea obtener. Sin embargo, se puede notar que el baricentro proyectado no es único, lo que sugiere que la variedad $\mathcal{M}$, en este caso, no es geodésicamente convexa. Esta es una diferencia significativa con respecto al baricentro clásico, que sí es único y consistente.

Además, se observa que modificar el parámetro $n_P$ permite obtener baricentros más similares al baricentro clásico, pero con una mayor naturalidad. En particular, se destaca que el baricentro proyectado con $n_P = 5$ es el que más se asemeja al baricentro clásico. Por último, se observa que realizar un paseo aleatorio en la variedad $\mathcal{M}$ permite explorar diversas medidas que se encuentren en esta variedad, lo que puede ser útil para obtener distintas representaciones de la imagen que se desea obtener.

## 4.3. Baricentro de Wasserstein Bayesiano

Si es que se tiene un conjunto de modelos finito $\mathcal{M} \subseteq \mathcal{P}(\mathcal{X})$ con $|\mathcal{M}| = N < +\infty$, por definición, la medida posterior corresponde a una medida discreta. En este caso, se puede calcular el vector de verosimilitudes $L_n = (\mathcal{L}_n(\mu))_{\mu \in \mathcal{M}} \in \mathbb{R}^N$ y normalizarlo, para obtener

el vector de probabilidades de la posterior. En este enfoque, se muestrea un índice $i$ con probabilidad proporcional a $L_n$ y devuelve el modelo $\mu_i$.

Esta estrategia resulta muy conveniente cuando los modelos de $\mathcal{M}$ poseen densidad en $\mathbb{R}^d$ y los datos $D$ se encuentran en los soportes de $\mu \in \mathcal{M}$. Sin embargo, para el contexto de esta tesis, en la que se desea calcular el baricentro de un conjunto de imágenes, es muy frecuente que la verosimilitud sea nula. Esto resulta en que el soporte de la medida posterior sea bastante menor al original. Es más, cuando el número de datos $D$ aumenta, incluso para números muy pequeños, como $n = 20$, el soporte de la posterior se reduce rápidamente. Por estos motivos, se decide tomar otro enfoque, el cual se presenta en la siguiente sección.

### 4.3.1. Construcción de la Posterior Usando una GAN

Como se explicó en secciones anteriores, dada una medida de referencia $\mathbb{P}_X$[1], lo que hacen las redes generativas es aproximarla por medio de un modelo generativo $\mathbb{P}_G$. Gracias a esta propiedad, se propone utilizar una GAN como prior para poder calcular la posterior.

Cabe destacar que la idea de utilizar una GAN como prior se encuentra en un trabajo previo [52], pero este presenta fallos en su planteamiento. Por lo tanto, en este trabajo de tesis se solucionan y formalizan los errores cometidos en el trabajo anterior.

Dada una red generadora $G_\theta \colon \mathcal{Z} \to \mathcal{P}(\mathcal{X})$ con una medida en el espacio latente $\mathbb{P}_Z$, se propone utilizar como prior a la medida

$$\Pi^G \stackrel{\text{def}}{=} G_{\theta\sharp}\, \mathbb{P}_Z \,. \tag{4.8}$$

De esta manera, la posterior tendría la siguiente forma:

$$\Pi_n(\mathrm{d}\mu) \stackrel{\text{def}}{=} \frac{\mathcal{L}_n(\mu)}{\int_{\mathcal{P}(\mathcal{X})} \mathcal{L}_n(\nu)\ \Pi^G(\mathrm{d}\nu)}\ \Pi^G(\mathrm{d}\mu) = C^{-1}\mathcal{L}_n(\mu)\ \Pi^G(\mathrm{d}\mu), \tag{4.9}$$

donde $C \stackrel{\text{def}}{=} \int_{\mathcal{P}(\mathcal{X})} \mathcal{L}_n(\nu)\ \Pi^G(\mathrm{d}\nu)$ es una constante de normalización.

Dada una función arbitraria $g$ $\Pi_n$-integrable (como por ejemplo, la función $\mu \mapsto W_p(\mu,\nu)^p$), se puede comprobar lo siguiente:

$$\int_{\mathcal{P}(\mathcal{X})} g(\mu)\ \Pi_n(\mathrm{d}\mu) \tag{4.10}$$

$$= C^{-1} \int_{\mathcal{P}(\mathcal{X})} g(\mu)\mathcal{L}_n(\mu)\ \Pi^G(\mathrm{d}\mu) \tag{4.11}$$

$$= C^{-1} \int_{\mathcal{Z}} g(G_\theta(z))\mathcal{L}_n(G_\theta(z))\ \ \mathbb{P}_Z(\mathrm{d}z) \tag{4.12}$$

$$= \int_{\mathcal{Z}} g(G_\theta(z))\ \Pi_n^Z(\mathrm{d}z), \tag{4.13}$$

[1]Del cuál se tiene acceso a través de una estimación por medio de la medida empírica $\hat{\mathbb{P}}_X = \frac{1}{N}\sum_{i=1}^N \delta_{x_i}$, donde $x_i \sim \mathbb{P}_X$.

donde $\Pi_n^Z(\mathrm{d}z)$ es la *medida posterior en el espacio latente*, y se define por

$$\Pi_n^Z(\mathrm{d}z) \stackrel{\text{def}}{=} C^{-1}\mathcal{L}_n(G_\theta(z))\ \mathbb{P}_Z(\mathrm{d}z). \tag{4.14}$$

Por la Definición 1.1.1 del operador push-forward, se concluye la siguiente propiedad de la medida posterior:

$$\Pi_n = G_{\theta\sharp}\Pi_n^Z. \tag{4.15}$$

La forma de interpretar la ecuación anterior, es que para obtener un muestreo de la posterior $\mu \sim \Pi_n$ basta con hacer un muestreo $z \sim \Pi_n^Z$ y después aplicar la red generadora $G_\theta$ sobre $z$.

Esto resulta beneficioso, pues delega la tarea de realizar un muestreo a partir de la posterior al espacio latente $\mathcal{Z}$ (el cuál, en la práctica, es $\mathbb{R}^{d_z}$), la cual es mucho más fácil de simular. Por ejemplo, una manera de obtener muestreos a partir de $\Pi_n^Z$, es por medio del método de Markov Chain Monte Carlo (MCMC) [3], [13], [28].

### Simulación de la Posterior

Sea un modelo $\tilde{\mu} \in \mathcal{M}$ fijo. Si $D \stackrel{\text{def}}{=} \{x_i\}_{i=1}^n \sim$ son $n$ datos a partir de $\tilde{\mu}$, se desea comprobar si las muestras de la posterior $\Pi_n$ son parecidos al modelo original $\tilde{\mu}$. Se espera que a medida que $n$ aumente, las muestras de la posterior se parezcan más al modelo original.

Para ello, se seleccionará un modelo $\tilde{\mu} \in \mathcal{M}$ que corresponde a una imgaen y se muestrearán $n$ datos $D = \{x_i\}_{i=1}^n$ a partir de $\tilde{\mu}$ para $n \in \{5, 20, 50, 100\}$. En la Figura 4.11 se muestra el modelo $\tilde{\mu}$ seleccionado.


Figura 4.11: Imagen $\tilde{\mu} \in \mathcal{M}$ de donde se muestrearán los datos.

A partir de la Ecuación 4.15, se puede determinar la función de densidad de la medida $\Pi_n^Z$:

$$\rho_{\Pi_n^Z}(z) = \frac{1}{C}\mathcal{L}_n(G_\theta(z))\frac{1}{\sqrt{2\pi}^{d_z}}\exp\left(-\frac{1}{2}\|z\|_2^2\right) \tag{4.16}$$

$$\propto \mathcal{L}_n(G_\theta(z))\exp\left(-\frac{1}{2}\|z\|_2^2\right). \tag{4.17}$$

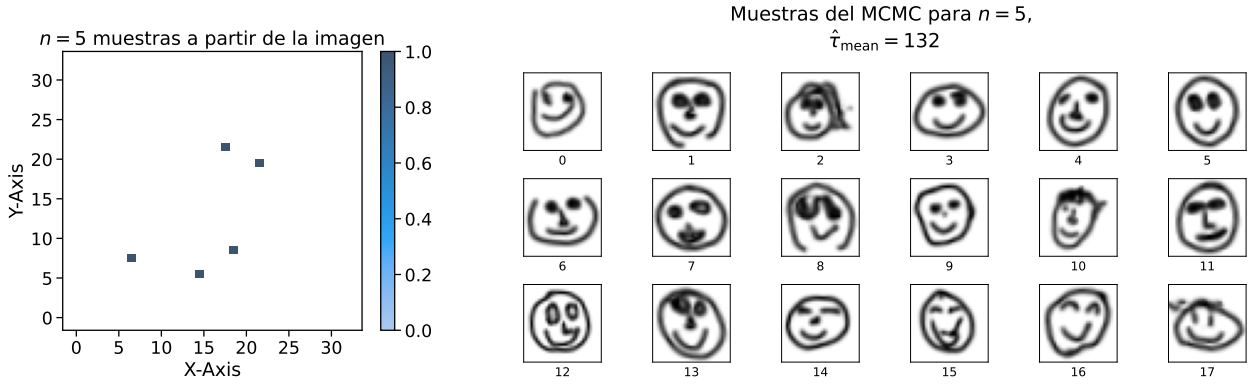
donde se asume que la distribución del espacio latente $\mathbb{P}_Z$ es una distribución normal estándar en $\mathbb{R}^{d_z}$. Gracias a que se tiene acceso a la función de densidad de $\Pi_n^Z$, se puede utilizar la técnica de MCMC para simular muestras de la posterior $\Pi_n$. En particular, se utilizará el Muestreador No-U-Turn (NUTS por sus siglas en inglés) [33] utilizando la implementación de *hamiltorch* [16].

Para la función potencial, se tomará el negativo del logaritmo de la Ecuación 4.17, salvo las constantes de normalización:

$$U(z) = -\log\left(\mathcal{L}_n(G_\theta(z))\right) - \log\left(\exp\left(-\frac{1}{2}\|z\|_2^2\right)\right) \tag{4.18}$$

$$= -\sum_{i=1}^{n} \log\left(\nu_z(x_i)\right) + \frac{1}{2}\|z\|_2^2, \tag{4.19}$$

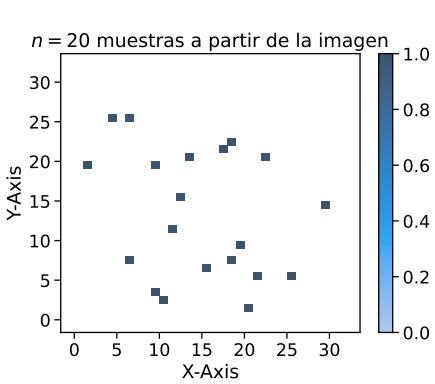
donde $\nu_z = G_\theta(z)$, mientras que para el primer término se utiliza el hecho de que $\mathcal{L}_n(\nu_z) = \prod_{i=1}^n \nu_z(x_i)$. Para los expermientos, se realizan muestreos para $n \in \{5, 20, 50, 100\}$, en donde cada experimento se ejecutaron 8 cadenas en paralelo, donde cada una realiza 150,000 pasos del MCMC con un *burn-in* de 2,000 pasos. Los resultados de los experimentos se pueden observar en las Figuras 4.12, 4.13, 4.14 y 4.15 respectivamente. En cada una de las figuras, se adjunta una estimación del tiempo de autocorrelación promedio, calculado con una adaptación de la función `autocorr` de la librería *emcee* [23].



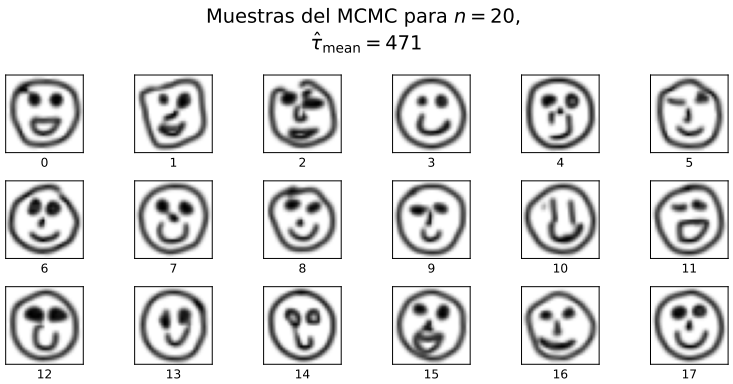

(a) Histograma de los datos.

(b) Muestras de la posterior.

Figura 4.12: Muestras de la posterior $\Pi_n^Z$ a partir de $n = 5$ datos.

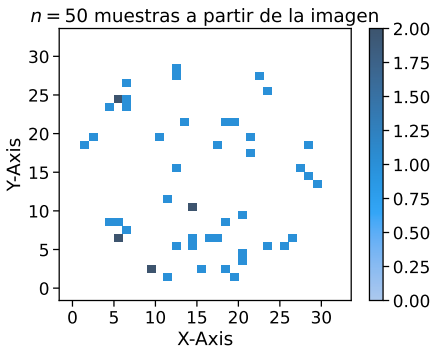



(a) Histograma de los datos.

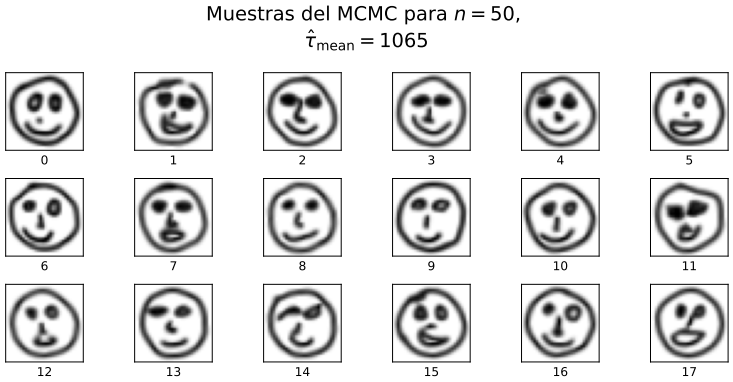



(b) Muestras de la posterior.

Figura 4.13: Muestras de la posterior $\Pi_n^Z$ a partir de $n = 20$ datos.

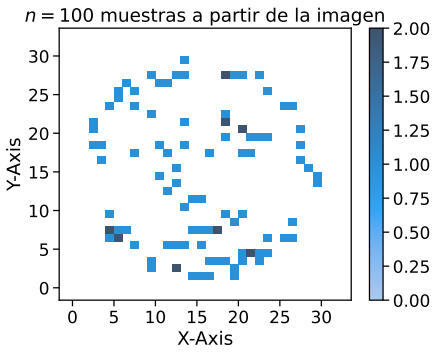



(a) Histograma de los datos.

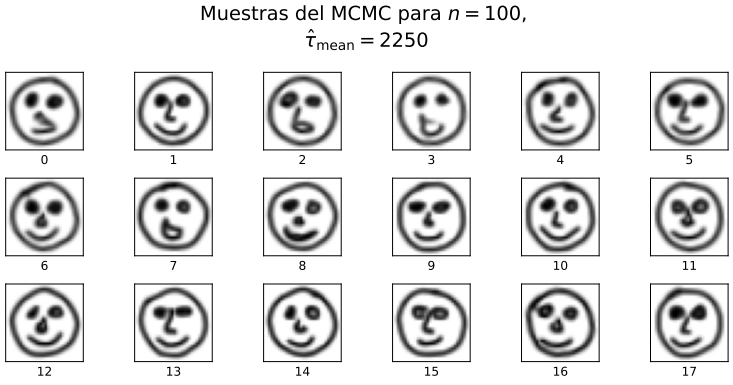



(b) Muestras de la posterior.

Figura 4.14: Muestras de la posterior $\Pi_n^Z$ a partir de $n = 50$ datos.




(a) Histograma de los datos.




(b) Muestras de la posterior.

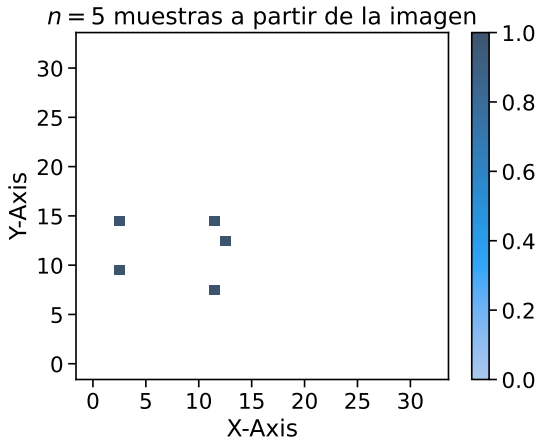
Figura 4.15: Muestras de la posterior $\Pi_n^Z$ a partir de $n = 100$ datos.

A partir de las Figuras 4.12, 4.13, 4.14 y 4.15, se puede observar que a medida que se aumenta el número de datos $n$, las muestras de la posterior $\Pi_n^Z$ se parecen más al modelo original $\tilde{\mu}$. Esto es lo esperable en un enfoque Bayesiano.

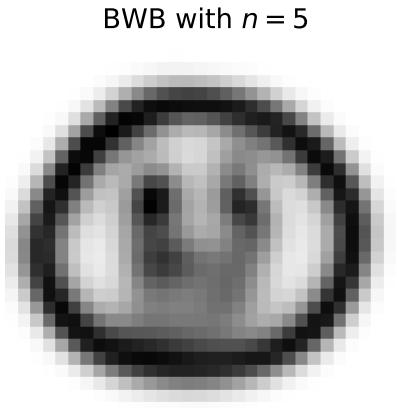
### 4.3.2. Cálculo del Baricentro de Wasserstein Bayesiano

Ahora que se conoce que se pueden simular muestras de la posterior $\Pi_n$ a partir de una red generativa, se puede calcular el baricentro de Wasserstein Bayesiano. Para ello, se utilizará el algoritmo de SGDW presentada en la Sección 4.1 con distintos valores de $n$ para observar cómo se comporta el baricentro a medida que se aumenta el número de datos.

En la sección anterior se observa que con $n = 50$ ya hay muestras muy similares a la imagen original. Por este motivo, se realizarán experimentos con $n \in \{5, 10, 25, 50\}$. Para el MCMC se utilizará 16 cadenas diferentes de un largo de 25 000 cada una, quemando los primeros 2 500 pasos. Y para los parámetros del SGDW se utilizarán los mismos que en la Sección 4.1.

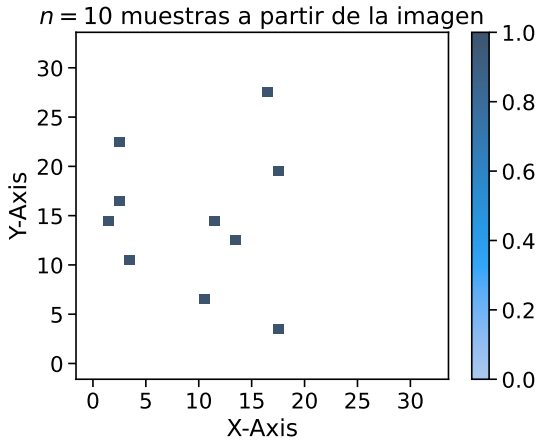



(a) Histograma de los datos con $n = 5$.

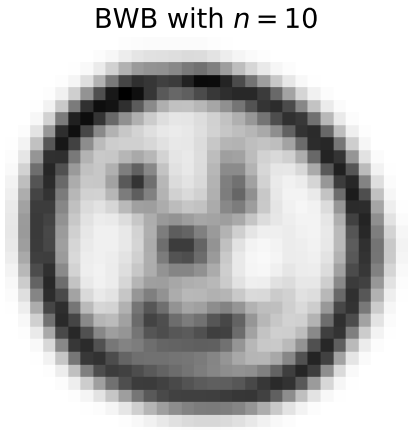



(b) Baricentro con $n = 5$.

Figura 4.16: Resultados de los experimentos para el cálculo del BWB con $n = 5$.

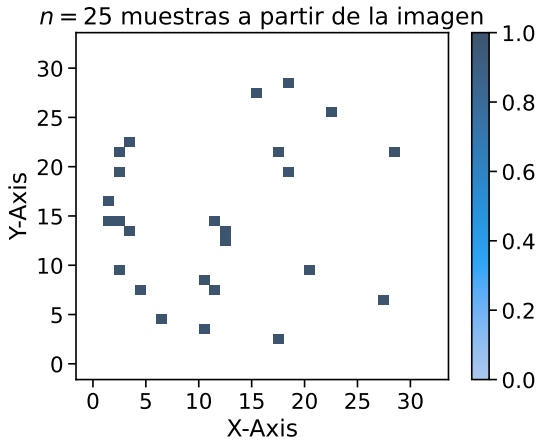



(a) Histograma de los datos con $n = 10$.

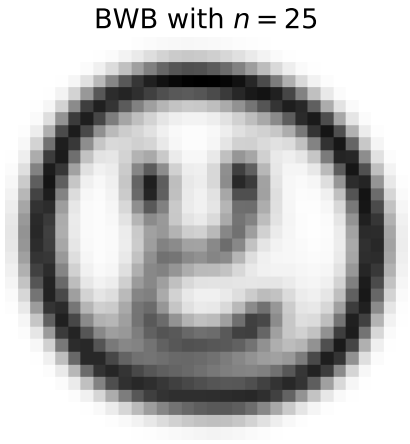



(b) Baricentro con $n = 10$.

Figura 4.17: Resultados de los experimentos para el cálculo del BWB con $n = 10$.

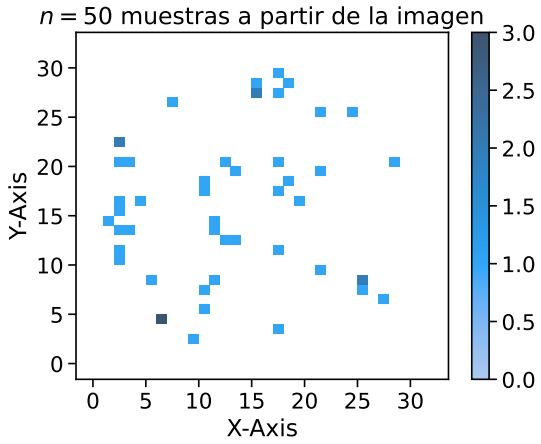



(a) Histograma de los datos con $n = 25$.

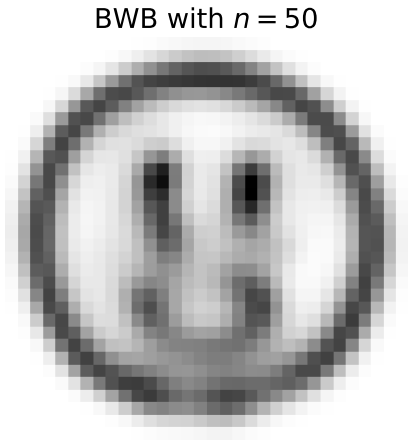



(b) Baricentro con $n = 25$.

Figura 4.18: Resultados de los experimentos para el cálculo del BWB con $n = 25$.




(a) Histograma de los datos con $n = 50$.




(b) Baricentro con $n = 50$.

Figura 4.19: Resultados de los experimentos para el cálculo del BWB con $n = 50$.

A partir de las figuras anteriores se puede observar que a medida que se aumenta el número de datos $n$, el baricentro de Wasserstein Bayesiano se parece más al modelo original $\tilde{\mu}$, pero de manera difuminada, como lo son los baricentros clásicos (Fig. 4.3). Esto es lo esperable, pues a medida que se aumenta el número de datos, la posterior se concentra más en el modelo original, lo que se traduce en un baricentro más parecido a $\tilde{\mu}$.

### 4.3.3. Conclusiones

En vista a los resultados obtenidos, se concluye que el enfoque de obtener muestreos a partir de la posterior usando una red generativa como prior ha resultado de manera satisfactoria. Además, que la implementación del BWB ha resultado exitoso y que se ha logrado obtener baricentros de manera efectiva.

# Conclusión

Para finalizar este trabajo de tesis, se presentan las conclusiones obtenidas a partir de los resultados obtenidos en los capítulos anteriores. En particular, se resumen los objetivos específicos y generales, y se discute el trabajo futuro que se puede realizar a partir de esta investigación.

A nivel general, el resultado de este trabajo de tesis proporciona una librería en *Python*, escrita utilizando *PyTorch*. Esta librería hace uso de la GPU para el cálculo, tanto de baricentros de Wasserstein de población, como de baricentros de Wasserstein Bayesianos utilizando el SGDW. Dicho esto, el objetivo general de este trabajo de tesis se ha cumplido. Más aún, los resultados se pueden observar y replicar en el repositorio desarrollado en [44].

Pasando a los objetivos específicos, en la Sección 4.1.2 se explican dos maneras de estimar la medida $\Gamma \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$ que se requiere para el cálculo del baricentro de Wasserstein Bayesiano. En particular, se proponen dos maneras de estimar la medida $\Gamma$: la primera es utilizar una red generativa, y la segunda es ocupar el conjunto de datos directamente. De esta manera, se cumple el objetivo específico **OE1**. Por otro lado, en la Sección 4.3.1 se propone una manera de estimar la medida posterior $\Pi_n(\mathrm{d}\mu)$ usando una red generativa como distribución a priori, de manera que se cumple el objetivo específico **OE3**.

Con el fin de cumplir el objetivo específico **OE2**, en la Sección 4.1 se presenta el Algoritmo 4.1, que es una adaptación del Algoritmo 1.1 para el cálculo de baricentros de Wasserstein que no posean densidad con respecto a la medida de Lebesgue. De este modo, se logra implementar el SGDW para una medida $\Gamma$, cumpliendo el objetivo específico **OE2**.

Por último, cumplidos los objetivos específicos **OE2** y **OE3**, se calcula el baricentro de Wasserstein Bayesiano en la Sección 4.3.2 para distintos valores de $n$, cumpliendo el objetivo específico **OE4**.

Adicionalmente, en el transcurso de la tesis se ha desarrollado una manera novedosa de entrenar redes generativas adversarias basadas en la distancia de Wasserstein. Esto, para que el generador posea un codificador, con el objetivo de obtener un proyector sobre la variedad de imágenes deseada. Sin embargo, como este no ha sido el enfoque principal de la tesis, es necesario realizar un estudio más profundo para determinar si este enfoque es efectivo, además de acompañar los resultados visuales con métricas que permitan comparar este método con otros del estado del arte. Se puede observar el desarrollo de este trabajo en el repositorio [46].

Con respecto al descenso del gradiente estocástico, en la Sección 4.2 se ha propuesto una extensión del algoritmo SGDW a una versión proyectada, de manera que el resultado del

baricentro tenga un aspecto más natural. Esta extensión del algoritmo abre otras posibilidades de estudio, pues aún falta calibrar los parámetros efectivos. A pesar de esto, los resultados preliminares resultan prometedores.

# Bibliografía


[1] M. Agueh y G. Carlier, «Barycenters in the Wasserstein space,» *SIAM Journal on Mathematical Analysis*, vol. 43, n.º 2, págs. 904-924, 2011.

[2] J. Amaya, *Optimización No Lineal, Parte IV: Algoritmos para optimización no lineal*, Material Docente de UCursos, notas de clase para MA5701-1, mayo de 2022.

[3] C. Andrieu, N. De Freitas, A. Doucet y M. I. Jordan, «An introduction to MCMC for machine learning,» *Machine learning*, vol. 50, págs. 5-43, 2003.

[4] M. Arjovsky, S. Chintala y L. Bottou, «Wasserstein generative adversarial networks,» en *Proceedings of the 34th International Conference on Machine Learning*, D. Precup e Y. W. Teh, eds., ép. Proceedings of Machine Learning Research, vol. 70, PMLR, ago. de 2017, págs. 214-223. dirección: `https://proceedings.mlr.press/v70/arjovsky17a.html`.

[5] J. Backhoff-Veraguas, J. Fontbona, G. Rios y F. Tobar, «Bayesian learning with Wasserstein barycenters,» *ESAIM: Probability and Statistics*, vol. 26, págs. 436-472, 2022.

[6] J. Backhoff-Veraguas, J. Fontbona, G. Rios y F. Tobar, «Stochastic Gradient Descent for Barycenters in Wasserstein Space,» *arXiv preprint arXiv:2201.04232*, 2023, To appear in Journal of Applied Probability. arXiv: `2201.04232 [math.OC]`.

[7] T. Bayes, «An essay towards solving a problem in the doctrine of chances,» *Biometrika*, vol. 45, págs. 296-315, 1765.

[8] J. Bigot y T. Klein, «Characterization of barycenters in the Wasserstein space by averaging optimal transport maps,» *ESAIM: Probability and Statistics*, vol. 22, págs. 35-57, 2018.

[9] N. Bonneel, M. Van De Panne, S. Paris y W. Heidrich, «Displacement interpolation using Lagrangian mass transport,» en *Proceedings of the 2011 SIGGRAPH Asia conference*, 2011, págs. 1-12.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein et al., «Distributed optimization and statistical learning via the alternating direction method of multipliers,» *Foundations and Trends® in Machine learning*, vol. 3, n.º 1, págs. 1-122, 2011.

[11] Y. Brenier, «Polar factorization and monotone rearrangement of vector-valued functions,» *Communications on pure and applied mathematics*, vol. 44, n.º 4, págs. 375-417, 1991.

[12] M. M. Breunig, H.-P. Kriegel, R. T. Ng y J. Sander, «LOF: identifying density-based local outliers,» en *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, págs. 93-104.

[13] S. Brooks, A. Gelman, G. Jones y X.-L. Meng, *Handbook of markov chain monte carlo*. CRC press, 2011.

[14] O. Calin, *Deep learning architectures*. Springer, 2020.

[15] R. J. Campello, D. Moulavi y J. Sander, «Density-based clustering based on hierarchical density estimates,» en *Pacific-Asia conference on knowledge discovery and data mining*, Springer, 2013, págs. 160-172.

[16] A. D. Cobb, A. G. Baydin, A. Markham y S. J. Roberts, «Introducing an Explicit Symplectic Integration Scheme for Riemannian Manifold Hamiltonian Monte Carlo,» *arXiv preprint arXiv:1910.06243*, 2019.

[17] M. Cuturi, «Sinkhorn distances: Lightspeed computation of optimal transport,» *Advances in neural information processing systems*, vol. 26, 2013.

[18] M. Cuturi y G. Peyré, «A smoothed dual approach for variational Wasserstein problems,» *SIAM Journal on Imaging Sciences*, vol. 9, n.º 1, págs. 320-343, 2016.

[19] M. Cuturi y J. Solomon, «A primer on optimal transport,» en *Tutorial of 31st Conference on Neural Information Processing Systems*, 2017.

[20] G. Cybenko, «Approximation by superpositions of a sigmoidal function,» *Mathematics of control, signals and systems*, vol. 2, n.º 4, págs. 303-314, 1989.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al., «A density-based algorithm for discovering clusters in large spatial databases with noise,» en *kdd*, vol. 96, 1996, págs. 226-231.

[22] R. Flamary, N. Courty, A. Gramfort et al., «POT: Python Optimal Transport,» *Journal of Machine Learning Research*, vol. 22, n.º 78, págs. 1-8, 2021. dirección: `http://jmlr.org/papers/v22/20-451.html`.

[23] D. Foreman-Mackey, D. W. Hogg, D. Lang y J. Goodman, «¡tt¿emcee¡/tt¿: The MCMC Hammer,» *Publications of the Astronomical Society of the Pacific*, vol. 125, n.º 925, págs. 306-312, mar. de 2013, ISSN: 1538-3873. DOI: `10.1086/670067`. dirección: `http://dx.doi.org/10.1086/670067`.

[24] M. Fréchet, «Les éléments aléatoires de nature quelconque dans un espace distancié,» en *Annales de l'institut Henri Poincaré*, vol. 10, 1948, págs. 215-310.

[25] «Generative adversarial network - Wikipedia — en.wikipedia.org.» [Accedido 09-05-2024]. (), dirección: `https://en.wikipedia.org/wiki/Generative_adversarial_network`.

[26] I. Goodfellow, Y. Bengio y A. Courville, *Deep learning*. MIT press, 2016, `http://www.deeplearningbook.org`.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., «Generative adversarial nets,» *Advances in neural information processing systems*, vol. 27, 2014.

[28] J. Goodman y J. Weare, «Ensemble samplers with affine invariance,» *Communications in applied mathematics and computational science*, vol. 5, n.º 1, págs. 65-80, 2010.

[29] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf y A. Smola, «A kernel method for the two-sample-problem,» *Advances in neural information processing systems*, vol. 19, 2006.

[30] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf y A. Smola, «A kernel two-sample test,» *The Journal of Machine Learning Research*, vol. 13, n.º 1, págs. 723-773, 2012.

[31] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin y A. C. Courville, «Improved training of wasserstein gans,» *Advances in neural information processing systems*, vol. 30, 2017.

[32] K. He, X. Zhang, S. Ren y J. Sun, «Deep residual learning for image recognition,» en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, págs. 770-778. eprint: 1512.03385.

[33] M. D. Hoffman, A. Gelman et al., «The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.,» *J. Mach. Learn. Res.*, vol. 15, n.º 1, págs. 1593-1623, 2014.

[34] K. Hornik, «Approximation capabilities of multilayer feedforward networks,» *Neural networks*, vol. 4, n.º 2, págs. 251-257, 1991.

[35] H. Janati, M. Cuturi y A. Gramfort, «Debiased sinkhorn barycenters,» en *International Conference on Machine Learning*, PMLR, 2020, págs. 4692-4701.

[36] J. Jongejan, H. Rowley, T. Kawashima, J. Kim y N. Fox-Gieg, «The quick, draw!-ai experiment,» *Mount View, CA, accessed Feb*, vol. 17, n.º 2018, pág. 4, 2016.

[37] L. V. Kantorovich, «On the translocation of masses,» en *Dokl. Akad. Nauk. USSR (NS)*, vol. 37, 1942, págs. 199-201.

[38] M. Kusner, Y. Sun, N. Kolkin y K. Weinberger, «From word embeddings to document distances,» en *International conference on machine learning*, PMLR, 2015, págs. 957-966.

[39] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *nature*, vol. 521, n.º 7553, págs. 436-444, 2015.

[40] J. Lellmann, D. A. Lorenz, C. Schonlieb y T. Valkonen, «Imaging with Kantorovich–Rubinstein Discrepancy,» *SIAM Journal on Imaging Sciences*, vol. 7, n.º 4, págs. 2833-2859, 2014.

[41] R. J. McCann, «A convexity principle for interacting gases,» *Advances in mathematics*, vol. 128, n.º 1, págs. 153-179, 1997.

[42] L. McInnes, J. Healy, N. Saul y L. Grossberger, «UMAP: Uniform Manifold Approximation and Projection,» *The Journal of Open Source Software*, vol. 3, n.º 29, pág. 861, 2018.

[43] G. Monge, «Mémoire sur la théorie des déblais et des remblais,» *Mem. Math. Phys. Acad. Royale Sci.*, págs. 666-704, 1781.

[44] F. Muñoz, *Bayesian-Learning-with-Wasserstein-Barycenters*, dic. de 2022. dirección: https://github.com/framunoz/Bayesian-Learning-with-Wasserstein-Barycenters.

[45] F. Muñoz, *quick-torch*, ver. 1.0.4, oct. de 2023. dirección: https://github.com/framunoz/quick-torch.

[46] F. Muñoz, *wgan-gp*, ago. de 2023. dirección: https://github.com/framunoz/wgan-gp.

[47] O. Nikodym, «Sur une généralisation des intégrales de MJ Radon,» *Fundamenta Mathematicae*, vol. 15, n.º 1, págs. 131-179, 1930.

[48] «Overview of GAN Structure — Machine Learning — Google for Developers — developers.google.com.» [Accedido 10-05-2024]. (), dirección: https://developers.google.com/machine-learning/gan/gan_structure.

[49] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone y J. C. Phillips, «GPU computing,» *Proceedings of the IEEE*, vol. 96, n.º 5, págs. 879-899, 2008. eprint: 1408.6923.

[50] V. M. Panaretos e Y. Zemel, *An invitation to statistics in Wasserstein space*. Springer Nature, 2020.

[51] A. Paszke, S. Gross, F. Massa et al., «PyTorch: An Imperative Style, High-Performance Deep Learning Library,» en *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, págs. 8024-8035. dirección: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[52] D. Patel y A. A. Oberai, «Bayesian inference with generative adversarial network priors,» *arXiv preprint arXiv:1907.09987*, 2019.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort et al., «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.

[54] S. Peleg, M. Werman y H. Rom, «A unified approach to the change of resolution: Space and gray-level,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, n.º 7, págs. 739-742, 1989. DOI: 10.1109/34.192468.

[55] G. Peyré y M. Cuturi, «Computational Optimal Transport,» *Foundations and Trends in Machine Learning*, vol. 11, n.º 5-6, págs. 355-607, 2019. DOI: 10.48550/ARXIV.1803.00567.

[56] G. Ríos Díaz, «Contributions to bayesian machine learning via transport maps,» Tesis doct., Universidad de Chile, 2020. dirección: https://repositorio.uchile.cl/handle/2250/173819.

[57] Y. Rubner, C. Tomasi y L. J. Guibas, «A metric for distributions with applications to image databases,» en *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, IEEE, 1998, págs. 59-66.

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

[59] C. Santana. «Creando Caras Artificiales Con Gans mejoradas: Data Coffee #4.» [Accedido 10-01-2024]. (nov. de 2017), dirección: https://youtu.be/bMDpMRB7CEs?si=JtMHA9kZ3uKrKl4T&t=272.

[60] D. Simon y A. Aberdam, *Barycenters of Natural Images Constrained Wasserstein Barycenters for Image Morphing*, https://github.com/drorsimon/image_barycenters, [Accedido 21-06-2024], jun. de 2020.

[61] D. Simon y A. Aberdam, «Barycenters of natural images constrained wasserstein barycenters for image morphing,» en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, págs. 7910-7919.

[62] J. Solomon, F. De Goes, G. Peyré et al., «Convolutional wasserstein distances: Efficient optimal transportation on geometric domains,» *ACM Transactions on Graphics (ToG)*, vol. 34, n.º 4, págs. 1-11, 2015.

[63] G. Tartavel, G. Peyré e Y. Gousseau, «Wasserstein loss for image synthesis and restoration,» *SIAM Journal on Imaging Sciences*, vol. 9, n.º 4, págs. 1726-1755, 2016.

[64] I. Tolstikhin, O. Bousquet, S. Gelly y B. Schoelkopf, «Wasserstein auto-encoders,» *arXiv preprint arXiv:1711.01558*, 2017. arXiv: 1711.01558 [stat.ML].

[65] C. Villani, *Optimal Transport: Old and New*. Springer Berlin Heidelberg, 2009, vol. 338, ISBN: 978-3-540-71049-3. DOI: 10.1007/978-3-540-71050-9.

[66] M. Welling e Y. W. Teh, «Bayesian learning via stochastic gradient Langevin dynamics,» en *Proceedings of the 28th international conference on machine learning (ICML-11)*, Citeseer, 2011, págs. 681-688.

[67] J.-Y. Zhu, P. Krähenbühl, E. Shechtman y A. A. Efros, «Generative visual manipulation on the natural image manifold,» en *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, Springer, 2016, págs. 597-613.

# Anexo A

# Demostraciones Adicionales de los Teoremas de la GAN

La siguiente demostración se basa en [25].

Demostración del Teorema 2.2.1. Dado $X \sim \mathbb{P}_X$, por la desigualdad de Jensen se tiene que:

$$\mathbb{E}_{Y\sim\mathbb{P}_D(\mathrm{d}y|X)}\left[\ln Y\right] \leq \ln\left(\mathbb{E}_{Y\sim\mathbb{P}_D(\mathrm{d}y|X)}\left[Y\right]\right). \tag{A.1}$$

Análogamente, dado $\tilde{X} \sim \mathbb{P}_G$, se tiene que:

$$\mathbb{E}_{Y\sim\mathbb{P}_D(\mathrm{d}y|\tilde{X})}\left[\ln\left(1-Y\right)\right] \leq \ln\left(\mathbb{E}_{Y\sim\mathbb{P}_D(\mathrm{d}y|\tilde{X})}\left[1-Y\right]\right). \tag{A.2}$$

Por el lema de Doob [58, ver Cor. 9.4.11], se sabe que existe una función $D\colon \mathcal{X} \to [0,1]$ tal que $\mathbb{E}_{Y\sim\mathbb{P}_D(\mathrm{d}y|X)}\left[Y\right] = D(X)$, es decir, que el discriminador toma una forma determinista $\mathbb{P}_D\left(\mathrm{d}y \mid x\right) = \delta_{D(x)}(\mathrm{d}y)$. Utilizando esta propiedad en las desigualdades (A.1) y (A.2) y sumando, se tiene que:

$$V(\mathbb{P}_G, \mathbb{P}_D) \leq \mathbb{E}_{X\sim\mathbb{P}_X}\Big[\ln D(X)\Big] + \mathbb{E}_{\tilde{X}\sim\mathbb{P}_G}\Big[\ln\left(1-D(\tilde{X})\right)\Big], \tag{A.3}$$

y dado que la parte derecha de la desigualdad es una cota superior de la función valor, s.p.g. se puede asumir que el discriminador toma una forma determinista en el óptimo. En tal caso, la función valor toma la forma del lado derecho de la desigualdad anterior.

Tomando $\mathbb{P} = \mathbb{P}_X + \mathbb{P}_G$, es claro que $\mathbb{P}_X \ll \mathbb{P}$ y $\mathbb{P}_G \ll \mathbb{P}$, y por tanto, existen las derivadas de Radon-Nikodym:

$$\rho_X = \frac{\mathrm{d}\,\mathbb{P}_X}{\mathrm{d}\,\mathbb{P}}, \qquad\qquad \rho_G = \frac{\mathrm{d}\,\mathbb{P}_G}{\mathrm{d}\,\mathbb{P}},$$

donde claramente $\rho_X + \rho_G = 1$. En tal caso, la función valor toma la siguiente forma:

$$V(\mathbb{P}_G, \mathbb{P}_D) = \int_{\mathcal{X}} \left[\rho_X(x)\ln D(x) + \rho_G(x)\ln\left(1-D(x)\right)\right]\ \mathbb{P}(\mathrm{d}x). \tag{A.4}$$

Notemos que la función $y \mapsto a \ln y + b \ln(1-y)$ alcanza su máximo en $y^* = \frac{a}{a+b}$, y por tanto, la función $D^*$ que maximiza la función valor es:

$$D^* = \frac{\rho_X}{\rho_X + \rho_G} = \frac{\mathrm{d}\,\mathbb{P}_X}{\mathrm{d}(\mathbb{P}_X + \mathbb{P}_G)}. \tag{A.5}$$

Por otro lado, si evaluamos $D^*$ en la función valor (A.4), se tiene que:

$$\begin{aligned} V(\mathbb{P}_G, \mathbb{P}_D^*) &= \mathbb{E}_{X\sim\mathbb{P}_X}\left[\ln \frac{\mathrm{d}\,\mathbb{P}_X}{\mathrm{d}\,\mathbb{P}}\right] + \mathbb{E}_{\tilde{X}\sim\mathbb{P}_G}\left[\ln \frac{\mathrm{d}\,\mathbb{P}_G}{\mathrm{d}\,\mathbb{P}}\right] \\ &= \mathrm{KL}\left(\mathbb{P}_X \mid \frac{\mathbb{P}_X + \mathbb{P}_G}{2}\right) - \ln 2 + \mathrm{KL}\left(\mathbb{P}_G \mid \frac{\mathbb{P}_X + \mathbb{P}_G}{2}\right) - \ln 2 \\ &= \mathrm{JS}\left(\mathbb{P}_X, \mathbb{P}_G\right) - 2\ln 2, \end{aligned}$$

lo que concluye la demostración. □

Demostración del Teorema 2.2.2. Por el Teorema anterior, se tiene que:

$$\begin{aligned} \min_{\mathbb{P}_G} \max_{\mathbb{P}_D} V(\mathbb{P}_G, \mathbb{P}_D) &= \min_{\mathbb{P}_G} V(\mathbb{P}_G, \mathbb{P}_D^*) \\ &= \min_{\mathbb{P}_G} C(\mathbb{P}_G) \\ &= \min_{\mathbb{P}_G} \mathrm{JS}\left(\mathbb{P}_X, \mathbb{P}_G\right) - 2\ln 2. \end{aligned}$$

Como la divergencia de Jensen-Shannon es estrictamente positiva si $\mathbb{P}_G \neq \mathbb{P}_X$, y nula si $\mathbb{P}_G = \mathbb{P}_X$, entonces se tiene que el mínimo global de la función $C(\mathbb{P}_G)$ se alcanza en $\mathbb{P}_G^* = \mathbb{P}_X$, y el valor mínimo es $C(\mathbb{P}_X) = -\ln 4$. Esto concluye la demostración. □

# Anexo B

# Complementos del SGDW

En este anexo se presentan algunos detalles adicionales de la Sección 4.1 del Capítulo 4. A decir, algunos parámetros adicionales, e imágenes de las iteraciones del algoritmo.

## B.1. Parámetros Adicionales

En el Algoritmo 4.1 se menciona que este debe de recibir como entrada los parámetros $(\eta_k)_k \in [0,1]^{\mathbb{N}}$ y $(S_k)_k \in \mathbb{N}_*^{\mathbb{N}}$. Sin embargo, no se menciona la manera de cálcular el baricentro en el paso 6. del algoritmo. En este trabajo, se utiliza una estimación del baricentro. Método el cuál también posee sus propios parámetros.

En los experimentos, se utiliza el método de Baricentros de Wasserstein Convolucionales Insesgados [35], implementada en la librería de POT [22]. En particular, se utiliza la función `convolutional_barycenter2d_debiased` del módulo `ot.bregman` de POT con los siguientes parámetros:

- `reg` fijado a $10^{-2}$.
- `numItermax` fijado a $10,000$.
- `stopThr` fijado a $10^{-3}$.

Estos parámetros han sido fijados de manera heurística, y no se ha realizado un estudio de sensibilidad de los mismos.

## B.2. Iteraciones del Algoritmo

Adicional a los baricentros presentados en la Figura 4.3, se han registrado las primeras y últimas iteraciones del cálculo de los baricentros. En partícular, se presenta la primera

medida del muestreo de $\Gamma$, que corresponde a la línea 4. del Algoritmo 4.1; y seguidamente se presenta el paso del algoritmo, que corrresponde a la línea 6. de dicho algoritmo.

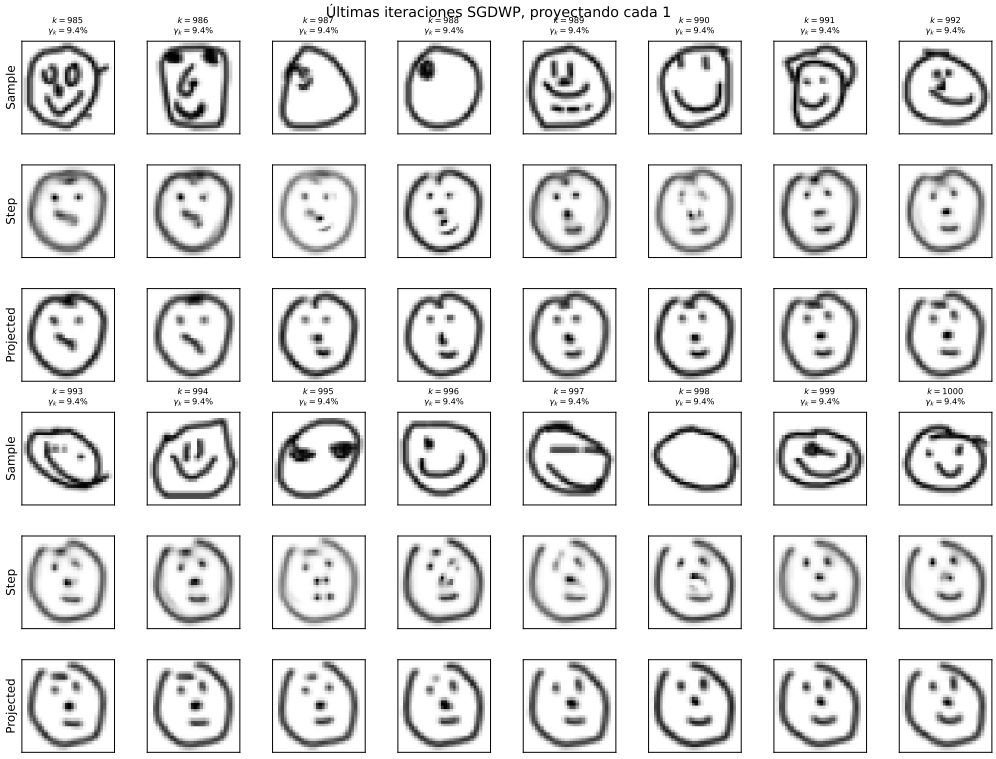
# Anexo C

# Complementos del SGDWP

## C.1. Iteraciones Adicionales

A continuación se presentan las primeras y últimas iteraciones que pasaron los baricentros proyectados de la Sección 4.2. En particular, los baricentros de las Figuras 4.9. Recordar que en este caso se utilizó los parámetros $n_P \in \{1, 3, 5, 10\}$.


(a) Primeras iteraciones del baricentro de la Fig. 4.9a



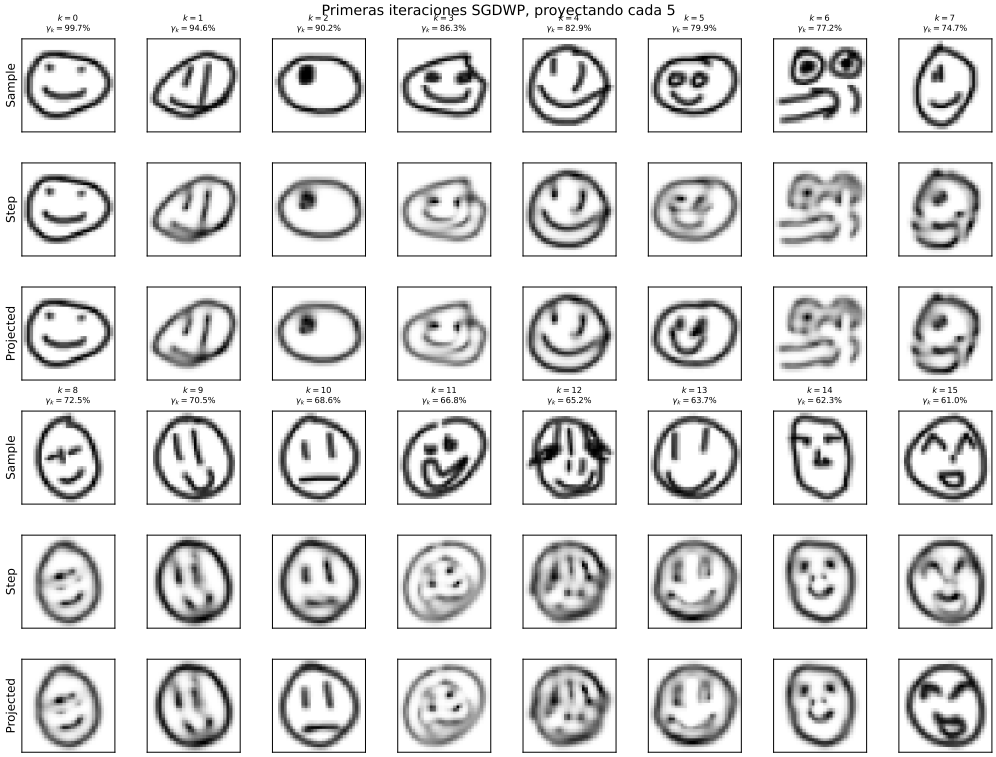

(b) Últimas iteraciones del baricentro de la Fig. 4.9a

Figura C.1: Iteraciones del cálculo del baricentro de la Fig. 4.9a.


(a) Primeras iteraciones del baricentro de la Fig. 4.9b


(b) Últimas iteraciones del baricentro de la Fig. 4.9b

Figura C.2: Iteraciones del cálculo del baricentro de la Fig. 4.9b.

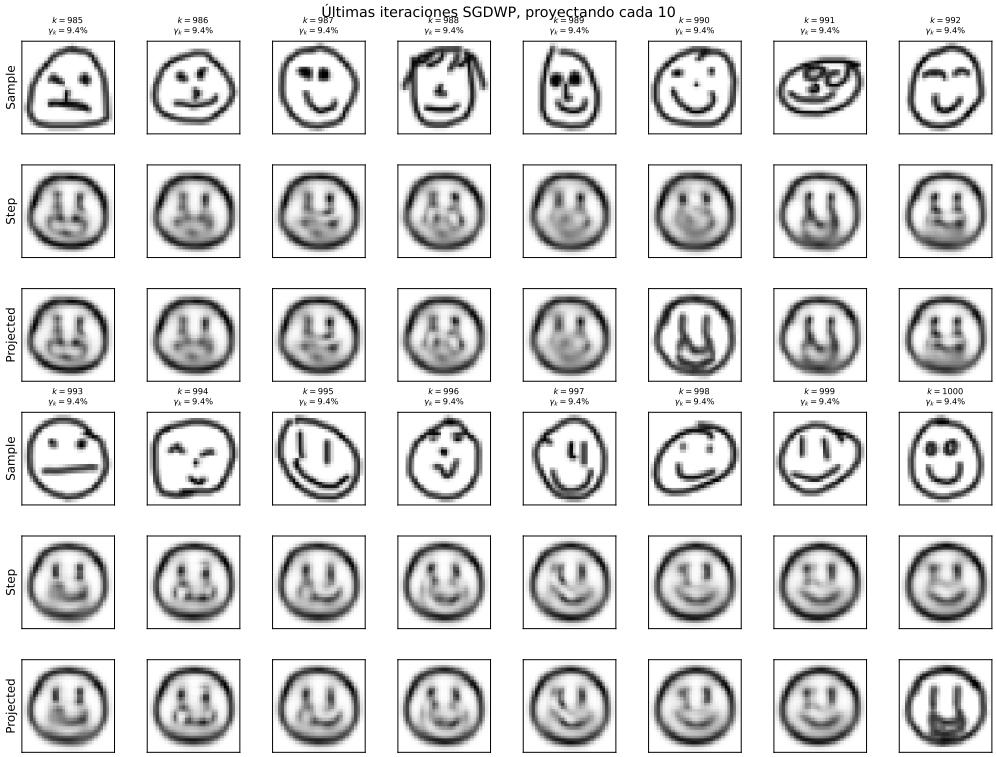



(a) Primeras iteraciones del baricentro de la Fig. 4.9c


(b) Últimas iteraciones del baricentro de la Fig. 4.9c

Figura C.3: Iteraciones del cálculo del baricentro de la Fig. 4.9c.


(a) Primeras iteraciones del baricentro de la Fig. 4.9d




(b) Últimas iteraciones del baricentro de la Fig. 4.9d

Figura C.4: Iteraciones del cálculo del baricentro de la Fig. 4.9d.