Universidad Nacional Autónoma De México

Centro de Física Aplicada y Tecnología Avanzada
Licenciatura en Tecnología

Diseño de un modelo generativo para muestras de comunidades microbianas complejas

Tesis

Que para obtener el título de:

Licenciado en Tecnología

Presenta:

OSCAR LOPÉZ ACEVEDO

Tutor:

DR. MARCO TULIO ANGULO BALLESTEROS

CONACyT - Instituto de Matemáticas
Universidad Nacional Autónoma de México

Juriquilla, Querétaro, México, 2020

Resumen Ejecutivo

Los microbios tienden a formar comunidades complejas que tienen un papel fundamental en la integridad de muchos ecosistemas del planeta. Por ejemplo, en el cuerpo humano, el microbioma intestinal tiene un rol principal en nuestra fisiología, enfermedades, e incluso nuestro comportamiento. En el suelo, los microbiomas tienen un papel fundamental en diversos procesos biogeoquímicos, como la producción y descomposición de nitrógeno. Es por tanto fundamental mejorar nuestro entendimiento de estas comunidades microbianas complejas.

Una de las formas más poderosas disponibles para entender estas comunidades microbianas complejas son los estudios metagenómicos. Estos estudios utilizan técnicas de secuenciación genética para inferir "muestras" que contienen la abundancia relativa de cada especie microbiana (o taxón) de la comunidad. Sin embargo, realizar estos estudios es económicamente costoso, lo que usualmente limita la cantidad de muestras que se pueden obtener de una comunidad microbiana.

Para resolver dicha limitación, en esta tesis construimos un algoritmo basado en Redes Neuronales Profundas que aprende a "generar" muestras de una comunidad microbiana dada. Una vez que el algoritmo es entrenado, las muestras que genera son estadísticamente muy cercanas a las muestras reales. Con más precisión, construimos una nueva arquitectura de Red Adversaria Generativa (GAN, por sus siglas en inglés) basada en la distancia de Wasserstein con penalización de gradiente. Buscamos que esta GAN tenga un aprendizaje eficiente utilizando un número limitado de muestras de entrenamiento.

Evaluamos sistemáticamente la habilidad de nuestra arquitectura para aprender a generar muestras de abundancia relativa usando datos sintéticos, comparando su desempeño con un modelo Dirichlet. Luego aplicamos nuestra arquitectura a datos de microbioma humano.

En su conjunto, los resultados de esta tesis muestran como las técnicas de *Aprendizaje Profundo* con redes neuronales pueden ser muy útiles para mejorar nuestro entendimiento de comunidades microbianas complejas.

Índice general

Capítulo 1

Introducción

Resumen

La vida microbiana es asombrosamente diversa, los microbios cubren todos los ecosistemas. Los microbios son vitales para todos los ecosistemas y son particularmente cruciales en zonas sin luz (es decir, donde la fotosíntesis no es el medio principal para recolectar energía). Los microbios participan en una serie de procesos ecológicos fundamentales, que incluyen la producción, la descomposición y la fijación del nitrógeno. Además, los microbios cumplen un papel principal en la regulación de los ciclos biogeoquímicos en prácticamente todos los entornos de nuestro planeta.

La metagenómica es el estudio del metagenoma, el genoma colectivo de microorganismos de una muestra ambiental, para proporcionar información sobre la diversidad microbiana y la ecología de un entornoz específico.

De forma general, existen dos enfoques para modelar datos, el enfoque discriminatorio y el enfoque generativo. Los modelos discriminatorios modelan la distribución de probabilidad condicional $P(y|x)$, mientras que los modelos generativos modelan la distribución conjunta de probabilidad $P(X = x, Y = y)$. Aunque en muchas ocasiones es más simple y suficiente utilizar un modelo discriminatorio, los modelos generativo (*probabilidad conjunta*) son más generales, incluso en problemas distintos a la clasificación.

Una familia de modelos generativos que han demostrado ser útiles en la práctica son las redes generativas adversarias (GAN, por sus siglas en inglés). Se han sido utilizadas principalmente para problemas de visión por computadora y generación de texto.

1.1. Motivación: importancia de las comunidades microbianas

Los microbios que viven en la superficie y en el interior de los humanos superan en número a las células somáticas y germinales aproximadamente en un factor de diez [29]. A las comunidades de microbios se les conoce como microbiota o microbioma. Colectivamente, los genomas de estos simbiontes microbianos proporcionan características que los humanos no necesitan evolucionar por sí mismos [29]. Si consideramos a los humanos como un conjunto de células microbianas y humanas -donde los microbios tienen un papel crucial en nuestro metabilismo,entonces podemos concebir a los humanos como un supraorganismo.[1] [29]. Además de la importancia de los microbios para los humanos, éstos tienen un papel primario en los ciclos biogeoquímicos[2] de los ecosistemas.

1.1.1. El rol que juegan las comunidades microbianas en la salud humana y en los ecosistemas de nuestro planeta

A pesar de que la mayoría de los microbios no son visibles para los humanos, estos se encuentran presentes en todo tipo de medio ambiente, incluso en aquellos extremos donde otras formas de vida sucumben. Los microbios tienden a formar comunidades complejas que juegan un papel esencial en los ecosistemas de nuestro planeta, regulando los ciclos biogeoquímicos de los ecosistemas, siendo el ciclo del carbón y el ciclo del nitrógeno. Los microbios tienen varias funciones, entre ellas:

[1] Un **supraorganismo** se refiere a una colección de individuos que se comportan como una sola unidad con una función mejorada Glendinning y Free [5]

[2] Un **ciclo biogeoquímico** es una vía por la cual un elemento químico (como el carbono o el nitrógeno) circula y es reciclado por un ecosistema.

1. **Los microbios y la atmósfera:** Los microbios regulan la atmósfera [2]. El carbón es el elemento más abundante en los seres vivos, al mismo tiempo la mayor fuente de carbono está en la atmósfera como CO_2. Sin embargo, la mayoría de los animales y bacterias no pueden hacer uso de CO_2. Las plantas y algunas bacterias a través de la fotosíntesis convierten el CO_2 en glucosa y oxígeno. Las bacterias son responsables de aproximadamente la mitad de los procesos de fotosíntesis.

2. **Los microbios y control de derrame de petróleo:** Otro ejemplo de uso de microorganismos es el limpiar derrames de gasolina [2]. Cada especie de microbio tiene un metabolismo, los cuales utilizan para transformar los componentes de la gasolina hasta dejar CO_2 y H_2O

3. **Los microbios y su importancia para el desarrollo de algunas plantas:** Los microbios también juegan un papel crucial en el desarrollo de las plantas [2]. Las bacterias ayudan a fijar en el suelo el nitrógeno que se encuentra en la atmósfera, el cual no puede ser usado por las plantas en forma de nitrógeno molecular. También las bacterias ayudan a reciclar nutrientes de plantas y animales muertos.

4. **Los microbios y la salud humana:** En los humanos, las comunidades microbianas hospedadas en nuestro cuerpo tienen un papel central en nuestra nutrición, resistencia y desarrollo de enfermedades, e inclusive nuestro comportamiento. Turnbaugh y col. [29] destaca que existen alrededor de diez veces más microorganismos en el humano que células. Los microorganismos impactan varios aspectos de la salud humana. Por ejemplo, el microbioma intestinal tiene un impacto muy profundo en nuestra salud y nutrición. Especialmente, la población más grande de microorganismos en los humanos se encuentra en los intestinos. La búsqueda de relaciones entre diferencias en microbiomas y enfermedades es un área de investigación activa Pepper y Rosenfeld [24]. Por ejemplo, [13] estudia los efectos del microbioma intestinal en el sistema inmunológico en los humanos.

1.1.2. ¿Qué es la metagenómica?

La metagenómica es el estudio del metagenoma, el genoma colectivo de microorganismos de una muestra ambiental, para proporcionar información sobre la diversidad microbiana y la ecología de un entorno específico. La metagenómica de *shotgun* se refiere a fragmentar ADN extraído de la muestra ambiental y la secuenciación de los pequeños fragmentos. La ventaja de los métodos de la metagenómica es que evaden a dificultad de cultivo y la diversidad genómica de la mayoría de los microbios, que son los mayores obstáculos para lograr entender las comunidades microbianas complejas en la clínica o el ambiente.

La metagenómica busca comprender la biología a nivel colectivo, trascendiendo el organismo individual para enfocarse en los genes de la comunidad y cómo los genes pueden influir en las actividades de los demás para cumplir funciones colectivas. Además, reconoce la necesidad de desarrollar métodos computacionales que maximicen la comprensión de la composición genética y las actividades de las comunidades tan complejas que solo pueden ser muestreadas, nunca completamente caracterizadas.

Todos los estudios de metagenómica toman el mismo primer paso: el ADN se extrae directamente de todos los microbios que viven en un ambiente particular. La muestra mixta de ADN puede analizarse directamente o clonarse en una forma mantenible en bacterias de laboratorio, creando una biblioteca que contiene los genomas de todos los microbios encontrados en ese entorno [2].

La palabra clon puede tener varios significados diferentes en biología. En el contexto de esta sección, la palabra clon se usa para describir un proceso mediante el cual los fragmentos de ADN aislados de una comunidad microbiana se insertan en piezas circulares de ADN llamadas plásmidos. Las bacterias de laboratorio pueden manipularse para absorber todos los plásmidos; Cuando las bacterias se dividen posteriormente, replican el plásmido junto con su ADN genómico. Cuando una extensa colección de plásmidos que contienen todos los fragmentos de ADN de una comunidad determinada se clona en un cultivo bacteriano, la colección resultante de bacterias se denomina biblioteca - un depósito vivo de todo el ADN de una comunidad microbiana.

La biblioteca se puede estudiar de varias maneras, basándose principalmente en el análisis de la secuencia de nucleótidos del ADN clonado o en la determinación de lo que pueden hacer los genes clonados cuando se

expresan como proteínas. Es esencial reconocer que la biblioteca no está organizada en volúmenes limpios, cada uno con el genoma de un miembro de la comunidad. En cambio, consta de millones de clones, cada uno con un fragmento aleatorio de ADN. Una biblioteca de metagenómica es como miles de rompecabezas mezclados en una sola caja: volver a armar los rompecabezas es uno de los desafíos más importantes de la metagenómica. El enfoque de la metagenómica ahora es posible debido a la disponibilidad de secuenciación de ADN económica y masiva [3] y las capacidades computacionales avanzadas necesarias para dar sentido a los millones de secuencias aleatorias contenidas en las bibliotecas.

[3]Secuenciación masiva también se conoce como **Next Generation Sequencing (NGS)** en inglés, o **High-throughput Sequencing (HTS**)

1.1.3. Herramientas metagenómicas para el estudiar la composición y abundancia de comunidades microbianas complejas.

Una definición común de los datos composicionales es que *son observaciones multivariables con valores positivos cuya suma es una constante, usualmente 1 o 100 %.*

$$S^D = \left\{ \mathbf{x} = [x_1, x_2, \ldots, x_D] \in \mathbb{R}^D | x_i > 0, i = 1, 2, \ldots, D; \sum_{i=1}^{D} x_i = 1 \right\} \tag{1.1}$$

Otra definición es que los datos composicionales *son vectores con componentes estrictamente positivos que llevan información relativa* [23]. Cuando esto se cumple los datos le llamamos datos composicionales [4].

Un simplex es la generalización de un tetraedro a N dimensiones. Los datos composicionales pertenecen a un simplex y la ecuación 1.1 es un modelo matemático de éste.

Finalmente, hay que considerar en qué tipo de información estamos interesados, ya que un mismo conjunto de datos puede ser composicional o no [4]. Por ejemplo, dado un conjunto de datos geoquímicos, uno podría estar interesado en verificar que el valor absoluto de un elemento químico no exceda un límite. Por otro lado, si se realiza un proceso químico puede ser más relevante la información relativa del mismo conjunto de datos geoquímicos.

En un nicho ecológico, pueden coexistir muchas especies diferentes, y su abundancia absoluta puede ser significativa. La abundancia de una especie puede no influir en la abundancia de otra. Sin embargo, la independencia de las especies no puede garantizarse en experimentos de secuenciación masiva porque los instrumentos de secuenciación pueden entregar lecturas solo hasta la capacidad del instrumento. Por lo tanto, es apropiado pensar que estos instrumentos contienen un número fijo de ranuras que deben llenarse. El recuento de lectura total obtenido de una máquina de secuenciación masiva es de tamaño fijo, y una muestra aleatoria de la abundancia relativa de las moléculas en el ecosistema subyacente [6].

Además, el recuento no puede relacionarse con el número absoluto de moléculas en la muestra de entrada, como se muestra en la **??**. Esto reconoce implícitamente cuando los conjuntos de datos de microbiomas se convierten en valores de abundancia relativa, o recuentos normalizados [16] [32] antes del análisis. Por lo tanto, el número de lecturas obtenidas es irrelevante y solo contiene información sobre la precisión de la estimación [3]. Los datos que se describen naturalmente como proporciones o probabilidades, o con una suma constante o irrelevante, se denominan datos composicionales. Los datos de composición contienen información sobre las relaciones entre las partes.

1.1.4. Limitaciones para obtener muestras de comunidades microbianas

Obtener muestras de comunidades microbianas es un proceso complicado que involucra conseguir muestras de humanos que deben ser valorados como “saludables” o “enfermos” por médicos. Además, el análisis para determinar qué especies están presentes y su abundancia es también costoso debido a que requiere secuenciar muestras de heces humanas.

Los costos para calcular abundancia de especie en microbioma varían mucho dependiendo de los laboratorios. Sin embargo, podemos estimar la magnitud de costos en base al proceso de secuenciación. En el método descrito en Qin y col. [25], primero se extrae ADN de heces usando un *Qiagen QIAamp DNA Stool mini kit*, con precio de 269 USD por un kit de 50 preparaciones. Después, se mide la concentración de ADN por espectrofotometría de UV visible de espectro completo. Por ejemplo, los precios de los espectrómetros NanoDrop™ de Thermo Scientific™ tienen costos que varían entre 5000 USD y 10000 USD. Después, se mide el tamaño del ADN por electroforesis en gel de agarosa. Luego, se preparan las librerías de ADN, para lo cual se utiliza otro kit, por ejemplo un *TruSeq Nano DNA LT Sample Preparation Kit Set A* (#FC-121-3001) con precio de €721 . Finalmente, se usan un *TruSeq PE Cluster Kit* (#PE-401-3001) €5,936 y un *TruSeq SBS Kit* (#FC-401-3002) €2,257 por el kit de 50 ciclos se utilizaron para realizar la generación de agrupamientos, hibridación de plantillas, amplificación isotérmica,

linealización, bloqueo y desnaturalización e hibridación de los cebadores de secuenciación. **En resumen, es un proceso que requiere de varios consumibles, equipo y personal especializado con habilidades técnicas en metagenómica.**. Realizando una sencilla consulta web los costos de secuenciación, encontramos que realizar 24 librerías metagenómicas, cuestan en promedio 2350 USD [4]. Otro proveedor de servicios de secuenciación de siguiente generación cobra 900 USD por 10 millones de secuencias con lecturas de 2x250bp [5].

1.1.5. Potenciales ventajas de modelos generativos para crear "muestras sintéticas"

La dificultad de adquirir muestras representativas de un microbioma, aunado al costo de secuenciación metagenómica, nos motivan a crear un método computacional que pueda generar "muestras sintéticas" que son estadísticamente indistinguibles de muestras reales. Para ello, proponemos utilizar modelos generativos basados en GANs.

1.2. Modelos generativos utilizando Redes Generativas Adversarias (GAN, por sus siglas en inglés)

1.2.1. ¿Qué es un modelo generativo?

Imaginemos que tenemos un problema de clasificacion. Para resolver el problema, debemos encontar la probabilidad a posteriori $p(Y|X)$, donde X y Y son los datos y las variables aleatorios respectivamente. Los posibles modelos se pueden dividr en generativos o discrimnativos.

- **Discriminativo**: este tipo de modelo calcula de forma directa calcula la probabilidad a posteriori $p(Y|X)$.
- **Generativo**: este tipo de modelo calcula la probabilidad conunta $p(x, y)$, la cual se relaciona con $p(Y|X)$ mediante el teorema de Bayes.

En algunos casos, el modelo geneativo estima la distribución p_{modelo} explícitamente, como se muestra en la figura 1.1. En otros casos, el modelo generativo estima la distribución de forma implícita y únicamente puede generar muestras de la distribución p_{modelo} [8], como se muestra en la figura 1.2 . Algunos modelos generativos pueden hacer ambas cosas. Las GAN se centran en la estimación implícita de la distribución de probabilidad, por lo que sirven para generar muestras la distribución estimada.

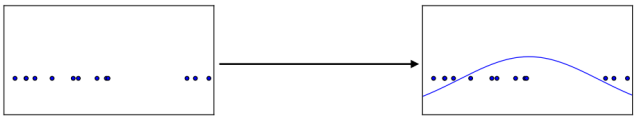

Figura 1.1: Modelos generativos realizan estimaciones de densidad.(Figura adaptada de Goodfellow [8]). Estos modelos toman ejemplos de un conjunto de datos de entrenamiento extraídos de una distribución de datos desconocida que generan datos p_{datos}, y devuelven una estimación de esa distribución p_{modelo}. La estimación p_{modelo} puede evaluarse para un valor particular de $\mathbf{x}$ para obtener una estimación $p_{modelo}(x)$ de densidad real $p_{modelo}(x)$. Esta figura ilustra el proceso para una colección de muestras de datos unidimensionales y un modelo Gaussiano (Goodfellow [8].)

[4] Consulta realizada el 24 de Octubre 2018 en https://phylogenomics.me/sequencing-costs/
[5] Consulta realizada el 24 de Octubre 2018 en http://www.mrdnalab.com/sequencing-service.html

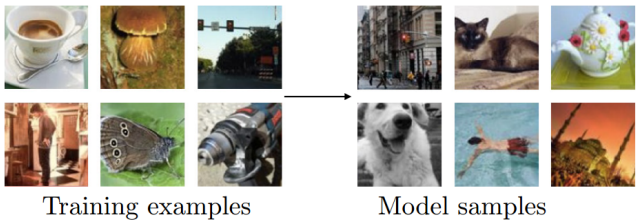

Figura 1.2: Muestras de datos utilizados para entrenar una GAN y datos creados por el generador de la GAN(Figura adaptada de Goodfellow [8]). Algunos modelos generativos pueden generar muestras a partir de la distribución del modelo. En esta ilustración del proceso, mostramos muestras del conjunto de datos ImageNet Org [21]. Un modelo generativo ideal podría ser capaz de entrenar con ejemplos como se muestra a la izquierda y luego crear más ejemplos de la misma distribución que se muestra a la derecha.

1.2.2. Historia de los modelos generativos y uso de las GAN

Las *Redes Adversarias Generativas* (GANs) son un algoritmo basado en un juego min-max. La arquitectura original de una GAN contiene dos perceptrones multicapa [7]. El primero se denota como *discriminador* y el otro como *generador*, vea la figura 1.3. La meta del *discriminador* es clasificar si una muestra de datos viene de la distribución de datos reales ó si fue creada por el *generador*. En contraste, la meta del *generador* es crear muestras artificiales que el discriminador no pueda distinguir de los datos reales. Este conflicto entre generador y discriminador motiva el nombre "redes adversarias".

El entrenamiento de una GAN consiste entonces en ajustar simultáneamente los pesos del discriminador para que mejore su habilidad de clasificación, al igual que ajustar los pesos del generador para crear muestras que "engañen" al discriminador. Este procedimiento puede realizarse a través de minimizar una función de costo adecuada utilizando el algoritmo *gradient descent*, u otro algoritmo de optimización.

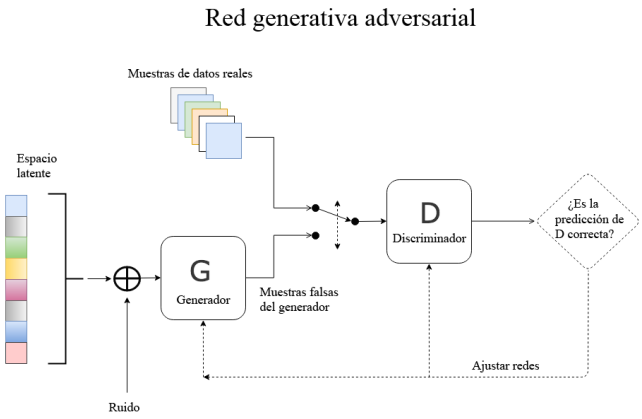

Figura 1.3: Diagrama de arquitectura de Redes Adversarias Generativas. De Generative Adversarial Networks – Hot Topic in Machine Learning, Al Gharakhanian, 2017,https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html

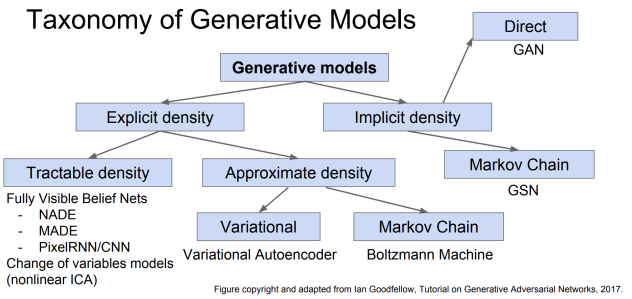

Figura 1.4: Diagrama de la taxonomía de los modelos generativos. Existen diferentes tipos de modelos generativos, sin embargo, las GAN han probado ser superiores en la práctica, creando muestras más realistas [8]

Existen diferentes familias de modelos generativos, vea figura 1.4. Las Redes Adversarias Generativas, tienen las siguientes características comparadas con otros modelos generativos [8]:

- Pueden generar muestras en paralelo.
- El diseño de la función del generador tiene muy pocas restricciones.
- No se necesitan cadenas de Markov.

- Se consideran subjetivamente como generadores de mejores muestras que otros métodos.
- Entrenarlas requiere encontrar el equilibrio de Nash de un juego, que es un problema más difícil que optimizar una función objetivo.

1.2.3. Aplicaciones existentes

A primera impresión, las GANs dan la sensación de ser un objeto de entretenimiento, produciendo imágenes curiosas. Además, las GANs se limitan a generar muestras de una distribución de probabilidad en lugar de estimar explícitamente la función de densidad de probabilidad.

Una razón por la cual los modelos generativos son útiles, y en particular las GAN, es que son una forma de representar y manipular distribuciones de probabilidad en altas dimensiones [8]. Las distribuciones de probabilidad en altas dimensiones son importantes en matemáticas aplicadas e ingeniería.

En las siguientes subsecciones mostramos algunas aplicaciones de generación de imágenes, en diferentes contextos, utilizando GANs

1.2.3.1. Generación de rostros humanos

En 2017, Karras y col. [14], figura 1.5, mostró que las GANs pueden generar fotografías realistas de humanos.

Figura 1.5: Rostros de humanos ficticios creados por GAN de Karras y col. [14]. Ésta GAN expone la capacidad de éstas para crear datos complejos, en particular imágenes

1.2.3.2. Rellenado de agujeros en imágenes

El llenado de agujeros en imágenes es una tarea desafiante donde las grandes regiones faltantes se deben rellenar según los datos visuales disponibles, observe figura 1.6. La GAN de Yeh y col. [33] logra reconstruir imágenes que tienen un agujero negro.

1.2.3.3. Traducción de texto a imagen

Zhang y col. [34] muestra el uso de su GAN para generar fotografías de apariencia realista a partir de descripciones textuales de objetos simples como pájaros y flores, observe figura 1.7.

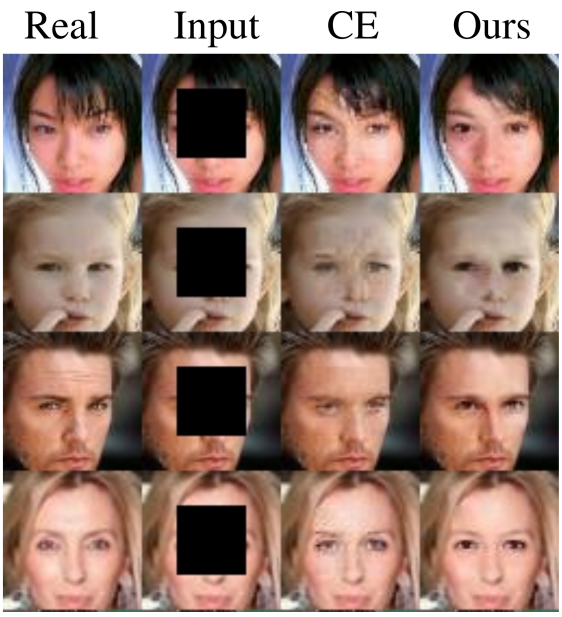

Figura 1.6: Comparación de imágenes reconstruidas usando la GAN de Yeh y col. [33]. Las imágenes resultantes son mejores que las obtenidas con *Context Encoder* (CE) de Pathak y col. [22]

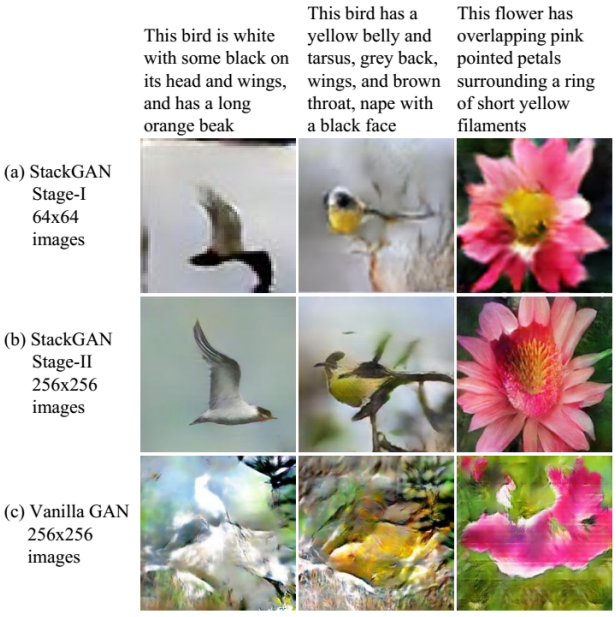

Figura 1.7: Muestras de imágenes generadas a partir de descripciones textuales usando la GAN de Zhang y col. [34]

1.2.3.4. Traducción semántica de imagen a foto

Wang y col. [31] utiliza una GAN para generar imágenes fotorrealistas de alta resolución a partir de mapas de etiquetas semánticas, observe figura 1.8.

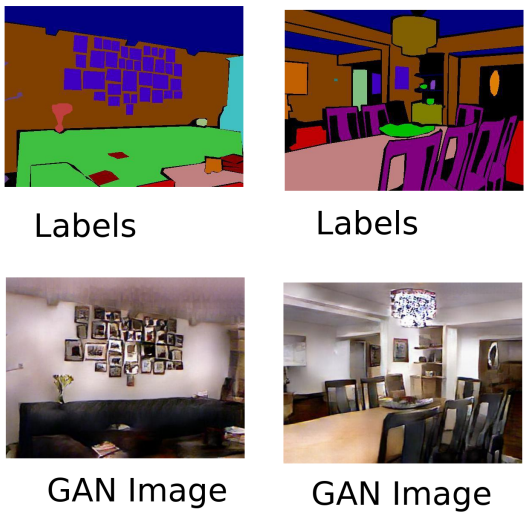

Figura 1.8: Ejemplos de imágenes generadas a de imágenes con etiquetas semánticas usando la GAN de Wang y col. [31]

1.2.3.5. Súper resolución

La súper resolución es el proceso de ampliación y/o mejora de los detalles dentro de una imagen, observe figura 1.9. Ledig y col. [15] propone una GAN para resolver esta tarea.

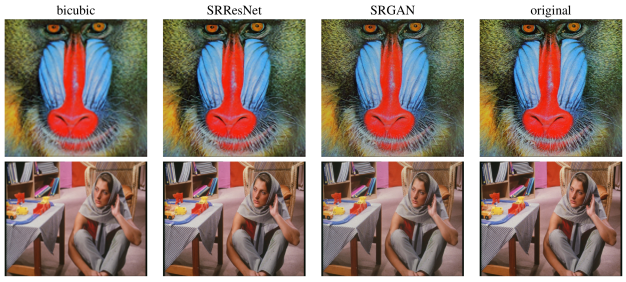

Figura 1.9: Comparación de diferentes métodos para ampliar imágenes. SRGAN corresponde a la GAN de Ledig y col. [15], la cual se aproximas mas a la imagen original.

1.2.3.6. Traducción de imagen a imagen

La traducción de imagen a imagen, figura 1.10, es una clase de problemas de visión y gráficos donde el objetivo es aprender el mapeo entre una imagen de entrada y una imagen de salida utilizando un conjunto de datos de entrenamiento de pares de imágenes alineados.

Isola y col. [12] propone una GAN que puede realizar diversas tareas de traducción de imagen a imagen. Algunos ejemplos de dichas tareas son:

- Traducción de imágenes semánticas a fotografías de paisajes urbanos y edificios.
- Traducción de fotografías satelitales a Google Maps.

- Traducción de fotos de día a noche.
- Traducción de fotografías en blanco y negro a color.
- Traducción de bocetos a fotografías en color.

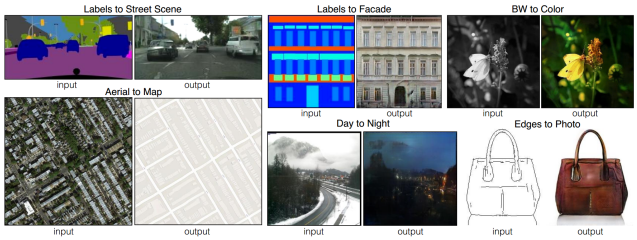

Figura 1.10: La traducción de imagen a imagen. Ejemplos de las diversas tareas de traducción de imagen a imagen de la GAN de Isola y col. [12]

1.3. Limitaciones de las GANs existentes y contribuciones principales de la tesis

1.3.1. Limitaciones de las GANs para generar muestras de microbioma

Las GANs descritas en Goodfellow y col. [7] y Arjovsky, Chintala y Bottou [1] están diseñadas para generar imágenes, no distribuciones de probabilidad. Por tanto, las arquitecturas existentes de GANs no puede inmediatamente aplicarse para generar muestras de microbiomas.

Una de las dificultades de los modelos generativos, y en particular de las GANs, es evaluar las muestras que producen. Lo que deseamos de un buen modelo generativo es que produzca muestras bien definidas, es decir, que las muestras no sean ambiguas, además que las muestras tengan diversidad. Originalmente, las GANs surgieron en el contexto de crear imágenes y en base la opinión humana determinaba la calidad de las muestras. En 2016, Salimans y col. [26] propone una métrica para evaluar la calidad de las imágenes creadas por las GANs sin utilizar el juicio de un humano. La métrica que proponen se llama *Inception Score*, donde propone utilizar una red neuronal clasificadora para obtener una distribución de probabilidad de clases, y construir una distribución de probabilidad marginal, después, calculan la divergencia Kullback-Leibler entre ambas distribuciones. La definición formal se puede ver en Salimans y col. [26], donde exhiben que *Inception Score* se correlaciona con el juicio de los humanos acerca del realismo de la fotos.

Una forma de examinar el aprendizaje de datos composicionales de microbioma la GAN es con una metodología que detallamos en la sección 6.1.

1.3.2. Contribuciones principales de la tesis

Las contribuciones principales de la tesis son las siguientes:

- Diseñar una arquitectura de GAN para aprender eficientemente datos composicionales de microbioma.
- Desarollar una metodología para evaluar el aprendizaje de la GAN aplicada a los datos composicionales de microbioma. La relevancia de tener dicha metodología se detalla en la sección 1.3.1
- Validar numéricamente la GAN, ademas analizar la GAN sobre una muestra de datos experimentales de microbioma.

Capítulo 2

Marco Teórico

Resumen

Las GANs son un modelo generativo que usualmente utiliza redes neuronales como sus componentes. El tipo de red neuronal usado en la GAN original de Goodfellow y col. [7], y uno usado de forma general, es el perceptrón multicapa. El estudio de las redes neuronales ha ganado mayor relevancia en los últimos años debido a que han mostrado ser útiles en problemas de reconocimiento de patrones. Un ejemplo es ResNet, una red neuronal ganadora de la competencia de identificación de imágenes ILSVRC 2015. Dicha competencia se realiza sobre un conjunto de datos llamados ImageNet que consiste en 14,197,122 imágenes etiquetadas [21]. El *Deep Learning*, el cual estudia redes neuronales de múltiples capas, es un subconjunto del *Machine Learning*. Varios métodos de *Machine Learning*, entre ellos el entrenamiento de redes neuronales se basa en la optimización matemática. Una familia de algoritmos de optimización más relevantes para el entrenamiento de redes neuronales es el descenso de gradiente estocástico y sus variantes. La GAN de Wasserstein, también conocida como WGAN, es una GAN propuesta por Arjovsky, Chintala y Bottou [1] la cual utiliza la distancia de Wasserstein o mueve tierra como función de costo. La WGAN ha mostrado tener un entrenamiento más estable y generar mejores muestras.

Las medidas de distancia o similitud son esenciales para resolver muchos problemas de reconocimiento de patrones, como la clasificación y la agrupación. Varias medidas de distancia 0 similitud están disponibles en la literatura para comparar dos distribuciones de datos. Como sugieren los nombres, una similitud mide qué tan cerca están dos distribuciones [17].

Los datos con los que estamos trabajando son abundancias relativas de las especies en un microbioma, por lo que son datos composicionales. Las muestras de datos de una distribución Dirichlet son un caso de *datos composicionales*, es decir que cumplen $\sum_{i=0}^{K} x_i = 1$, al igual que los datos de abundancias relativas. Por lo tanto, usamos un conjunto de datos de muestras de una distribución de Dirichlet.

Finalmente, revisamos la función Softmax, también llamada función exponencial normalizada. La función Softmax se usa en varios métodos de clasificación multiclase, tales como regresión logística multinomial, análisis discriminante lineal multiclase, clasificadores Bayesianos ingenuos y redes neuronales artificiales.

2.1. *Machine Learning*

El **Machine Learning** o aprendizaje automático puede definirse en términos generales como un conjunto de métodos computacionales que utilizan la experiencia para mejorar el rendimiento o para hacer predicciones precisas. Aquí, la experiencia se refiere a la información pasada disponible, que generalmente toma la forma de datos electrónicos recopilados y puestos a disposición para su análisis. Estos datos pueden ser en forma de conjuntos de entrenamiento digitalizados y etiquetados por humanos, u otros tipos de información obtenida a través de la interacción con el medio ambiente. En todos los casos, su calidad y tamaño son cruciales para el éxito de las predicciones [18].

2.1.1. Problemas en donde se usa *Machine Learning*

El *Machine Learning* permite resolver diferentes clases de tareas [18], entre ellas:

- **Clasificación de texto**: problemas como determinar el tema de un texto, definir si el contenido de un texto es *spam*.

- **Natural language processing (NLP)** : conocido en español procesamiento del lenguaje natural. Examina problemas como etiquetado de voz, análisis libre de contexto, reconocimiento de la entidad nombrada,
- **Procesamiento de voz** : Se ocupa de los problemas de reconocimiento de voz, síntesis de voz, verificación del hablante, identificación del hablante, así como problemas secundarios como modelado del lenguaje y modelado acústico.
- **Visión artificial** : Esto incluye reconocimiento de objetos, identificación de objetos, detección de rostros, reconocimiento óptico de caracteres (OCR), recuperación de imágenes basada en contenido, o estimación de pose.
- **Otros** : Muchos otros problemas, como la detección de fraudes para tarjetas de crédito, compañías telefónicas o de seguros, intrusión en la red, aprender a jugar juegos como el ajedrez, el Go, el control sin asistencia de vehículos como robots o automóviles, el diagnóstico médico, el diseño de sistemas de recomendación. Los motores de búsqueda o sistemas de extracción de información se abordan utilizando técnicas de aprendizaje automático.

2.1.2. Tareas para realizar con *Machine Learning*

Algunas de las tareas más comunes del *Machine Learning* son [9],[18]

- **Clasificación**: El algoritmo específica a cuál de las k categorías pertenece la entrada. Para ello, frecuentemente el algoritmo produce una función $f : \mathbb{R}^n \to \{1, ..., k\}$. Cuando $y = f(x)$, el modelo asigna un vector de entrada $\mathbf{x}$ a una categoría con una etiqueta numérica $\mathbf{y}$. Existen otras variantes de tareas de clasificación, por ejemplo, donde f tiene como resultado una distribución de probabilidad sobre las clases.
- **Regresión**: Éste problema se trata de predecir un valor numérico para una entrada dada. En el caso univariable, el algoritmo de aprendizaje produce una función $f : \mathbb{R}^n \to \mathbb{R}$. Un ejemplo de regresión es predecir el valor de acciones bursátiles
- **Clustering**: Este es el problema de particionar un conjunto de elementos en subconjuntos homogéneos. La agrupación a menudo se usa para analizar conjuntos de datos muy grandes. Por ejemplo, en el contexto del análisis de redes sociales, los algoritmos de agrupamiento intentan identificar comunidades naturales dentro de grandes grupos de personas.
- **Reducción de la dimensionalidad o aprendizaje de variedades**: Este problema consiste en transformar una representación inicial de los elementos en una representación de menor dimensión mientras se conservan algunas propiedades de la representación inicial. Un ejemplo común de reducción de dimensionalidad es el preprocesamiento de imágenes digitales en tareas de visión por computadora.
- **Detección de anomalías**: En este tipo de tarea, el programa de computadora examina un conjunto de eventos u objetos y señala que algunos de ellos son inusuales o atípicos. Un ejemplo de una tarea de detección de anomalías es la detección de fraude con tarjeta de crédito. Al modelar sus hábitos de compra, una compañía de tarjetas de crédito puede detectar el mal uso de sus tarjetas. Si un ladrón roba su tarjeta de crédito o información de la tarjeta de crédito, las compras del ladrón a menudo provendrán de una distribución de probabilidad diferente sobre los tipos de compra que la suya. La compañía de la tarjeta de crédito puede prevenir el fraude al suspender una cuenta tan pronto como esa tarjeta haya sido utilizada para una compra anómala.

2.1.3. Escenarios de aprendizaje

Los escenarios de aprendizaje difieren en los tipos de datos de entrenamiento disponibles para el alumno, el orden y el método por el cual se reciben los datos de entrenamiento y los datos de prueba utilizados para evaluar el algoritmo de aprendizaje [18].

- **Aprendizaje supervisado**: El alumno recibe un conjunto de ejemplos etiquetados como datos de entrenamiento y hace predicciones para puntos desconocidos. Este es el escenario más común asociado con problemas de clasificación, regresión y clasificación.
- **Aprendizaje no supervisado**: el alumno recibe exclusivamente datos de entrenamiento no etiquetados y realiza predicciones para todos los puntos desconocidos. Dado que, en general, no hay ejemplos etiquetados disponibles en ese entorno, puede ser difícil evaluar cuantitativamente el rendimiento de un alumno. La agrupación y la reducción de la dimensionalidad son un ejemplo de problemas de aprendizaje no supervisados.
- **Aprendizaje semi-supervisado**: El alumno recibe una muestra de entrenamiento que consta de datos etiquetados y no etiquetados, y hace predicciones para puntos desconocidos. El aprendizaje semi-supervisado es común en entornos donde los datos no etiquetados son fácilmente accesibles pero las etiquetas son caras de obtener. Varios tipos de problemas que surgen en las aplicaciones, incluidas las tareas de clasificación, regresión o clasificación, pueden enmarcarse como instancias de aprendizaje semi-supervisado. La esperanza es que la distribución de datos no etiquetados accesibles para el alumno pueda ayudarlo a lograr un mejor rendimiento que en el entorno supervisado. El análisis de las condiciones bajo las cuales esto puede realizarse es el tema de muchas investigaciones modernas de aprendizaje automático teórico y aplicado.
- **Aprendizaje por refuerzo**: Las fases de entrenamiento y evaluación también se entremezclan en el aprendizaje por refuerzo. Para recopilar información, el alumno interacciona activamente con el entorno y, en algunos casos, afecta el entorno y recibe una recompensa inmediata por cada acción. El objetivo del alumno es maximizar su recompensa en un curso de acciones e iteraciones con el entorno. Sin embargo, el entorno no proporciona retroalimentación de recompensa a largo plazo, y el alumno se enfrenta al dilema de exploración y explotación, ya que debe elegir entre explorar acciones desconocidas para obtener más información o explotar la información ya recopilada.

2.2. *Deep Learning*

El *Deep Learning* ha sido efectivo en situaciones donde los datos etiquetados son abundantes. Un ejemplo es ResNet, una red neuronal ganadora de la competencia de identificación de imágenes ILSVRC 2015. Dicha competencia se realiza sobre un conjunto de datos llamados ImageNet que consiste en 14,197,122 imágenes etiquetadas [21]. ResNet logró alcanzar un error de clasificación de 3.57 % [11], error que es menor al error humano en el mismo conjunto de datos. **Sin embargo, tener un conjunto de datos grande es la excepción en la mayoría de situaciones** . En particular, esto sucede en muestras del microbioma humano.

Dentro del *Deep Learning*, los modelos generativos tienen el potencial de maximizar la información disponible en el conjunto de muestras, permitiendo utilizar estas muestras para generar muestras sintéticas que son estadísticamente indistinguibles de las muestras verdaderas.

2.2.1. Perceptron Multicapa

Los perceptrones multicapa (MLP por sus siglas en inglés) son los modelos clásicos de aprendizaje profundo. El objetivo de un perceptrón multicapa es aproximar alguna función f^*. Por ejemplo, para un clasificador, $y = f^*(x)$ asigna una entrada $\boldsymbol{x}$ a una categoría $\boldsymbol{y}$. Una red neuronal *feedforward* define una asignación $y = f(x; \theta)$ y aprende el valor de los parámetros θ que dan como resultado la mejor aproximación de la función [9].

Estos modelos se denominan *feedforward* porque la información fluye a través de la función que se evalúa desde $\boldsymbol{x}$, a través de los cálculos intermedios utilizados para definir $\boldsymbol{f}$, y finalmente hasta la salida $\boldsymbol{y}$. No hay conexiones de retroalimentación en las que las salidas del modelo se realimentan. Cuando las redes neuronales *feedforward* se extienden para incluir conexiones de retroalimentación, se denominan redes neuronales recurrentes (RNN por sus siglas en inglés).

Las redes perceptrón multicapa se denominan redes porque, en general, se representan componiendo muchas funciones diferentes. El modelo se asocia con un gráfico acíclico dirigido que describe cómo se componen las funciones juntas. Por ejemplo, podríamos tener tres funciones $f^{(1)}$, $f^{(2)}$ y $f^{(3)}$ conectadas en una cadena, para

formar $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. Estas estructuras de cadena son las estructuras más utilizadas de las redes neuronales. En este caso, $f^{(1)}$ se llama la primera capa de la red, $f^{(2)}$ se llama la segunda capa, y así sucesivamente [9].

La longitud total de la cadena da la **profundidad** del modelo. El nombre "*Deep Learning*" surgió de esta terminología. La capa final de una red de avance se llama **capa de salida**. Durante el entrenamiento de la red neuronal, llevamos a $f(x)$ para que coincida con $f^*(x)$. Los datos de entrenamiento nos proporcionan ejemplos ruidosos y aproximados de $f^*(x)$ evaluados en diferentes puntos de entrenamiento. Cada ejemplo $\boldsymbol{x}$ va acompañado de una etiqueta $y \approx f^*(x)$. Los ejemplos de entrenamiento especifican directamente lo que debe hacer la capa de salida en cada punto $\boldsymbol{x}$; debe producir un valor cercano a $\boldsymbol{y}$. Los datos de entrenamiento no especifican directamente el comportamiento de las otras capas. El algoritmo de aprendizaje debe decidir cómo usar esas capas para producir la salida deseada, pero los datos de entrenamiento no dicen qué debe hacer cada capa individual. En cambio, el algoritmo de aprendizaje debe decidir cómo usar estas capas para implementar mejor una aproximación de f^*. Como los datos de entrenamiento no muestran el resultado deseado para cada una de estas capas, se denominan **capas ocultas**. [9]

Finalmente, estas redes se denominan neurales porque originalmente se inspiran en las neuronas INSERTE CITA. Cada capa oculta de la red suele tener un valor vectorial. La dimensionalidad de estas capas ocultas determina el **ancho** del modelo. Cada elemento del vector puede interpretarse como un papel análogo a una neurona. En lugar de pensar que la capa representa una sola función de vector a vector, también podemos pensar que la capa consiste en muchas **unidades** que actúan en paralelo, cada una representando una función de vector a escalar. Cada unidad se asemeja a una neurona en el sentido de que recibe información de muchas otras unidades y calcula su propio valor de activación. La idea de usar muchas capas de representaciones con valores vectoriales se extrae de la neurociencia. La elección de las funciones $f^{(i)}(x)$ utilizadas para calcular estas representaciones también se guía libremente por las observaciones neurocientíficas sobre las funciones que calculan las neuronas biológicas. Sin embargo, la investigación moderna de redes neuronales se guía por muchas disciplinas matemáticas y de ingeniería, y el objetivo de las redes neuronales no es modelar perfectamente el cerebro. Es mejor pensar en las redes de avance como máquinas de aproximación de funciones que están diseñadas para lograr una generalización estadística, en ocasiones extrayendo algunas ideas de lo que sabemos sobre el cerebro, en lugar de modelos de función cerebral. [9]

2.2.2. Entrenamiento de redes neuronales

2.2.2.1. Optimización y entrenamiento de redes neuronales

El entrenamiento de redes neuronales involucra un problema de optimización. La diferencia de un algoritmo de optimización utilizado para entrenar redes neuronales y uno usado en un problema de optimización tradicional es que en la red neuronal se utiliza de forma indirecta. Usualmente nos interesa el rendimiento de alguna métrica P definida respecto al conjunto de datos de prueba (test set), la cual puede ser intractable. Entonces, optimizamos a P de forma indirecta. Reducimos una función de costo $J(\theta)$ esperando que el valor de P mejore. En un problema de optimización puro, el objetivo sería únicamente minimizar directamente la función J [9].

Usualmente, la función de costo puede ser escrita como un promedio sobre el conjunto de datos de entrenamiento:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\mathrm{y})\sim\hat{p}_{\mathrm{datos}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y) \tag{2.1}$$

Donde:

- $\mathbb{E}_{(\boldsymbol{x},\mathrm{y})}$ es el promedio sobre los datos de entrenamiento.
- L es la función de pérdida por punto.
- $f(\boldsymbol{x};\boldsymbol{\theta})$ es el valor de salida predicho con entrada $\boldsymbol{x}$.
- $\hat{p}_{\mathrm{datos}}$ es la distribución empírica.

Una nota importante es que en el entrenamiento de redes neuronales utilizamos a $\hat{p}$, la distribución empírica de los datos, porque desconocemos la distribución p que genera los datos. Entonces, en la práctica minimizamos la siguiente ecuación:

$$\mathbb{E}_{\boldsymbol{x},\mathrm{y}\sim\hat{p}_{\mathrm{datos}}(\boldsymbol{x},y)}[L(f(\boldsymbol{x};\boldsymbol{\theta}),y)] = \frac{1}{m}\sum_{i=1}^{m} L\left(f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right),y^{(i)}\right) \tag{2.2}$$

Arriba, m es el número de muestras de entrenamiento, θ son los parámetros que definen la red neuronal, $x^{(i)}$ es la i-ésima muestra de entrenamiento, $y^{(i)}$ es la i-ésima etiqueta de la muestra de entrenamiento. En la práctica, no es común optimizar la función de costo empírica de forma directa, porque es propenso a un problema sobre ajuste, conocido como *overfitting* en inglés. Los algoritmos modernos de optimización usados en redes neuronales se basan en gradiente descendiente [9].

También, en varias ocasiones la función de pérdida que conforma la función de costo puede no ser optimizada de forma eficiente, entonces se utiliza una función de pérdida sustituta.

Calcular la función de costo a optimizar utiliza promedios sobre todo el conjunto de datos de entrenamiento, lo cual es computacionalmente caro. Entonces, en la práctica utilizamos un pequeño subconjunto de los datos de entrenamiento escogidos de forma aleatoria [9].

Una motivación adicional para utilizar un subconjunto de datos es el error estándar de la media,el cual está dado para n muestras por $\frac{\sigma}{\sqrt{n}}$. Lo cual sugiere que si bien incrementar el número de muestras mejora la estimación de la media, la mejora no varía de forma lineal, mientras el costo computacional incrementa significativamente. Una razón adicional para utilizar un subconjunto con pocas muestras es la redundancia de los datos [9].

Los subconjuntos de muestras previamente mencionados se llaman **minibatch**. El número de muestras en el *minibatch*, también llamado tamaño de *minibatch*, depende de varios factores:

- Un mayor número de muestras incrementa la precisión del gradiente, pero tiene un retorno sobre la inversión no lineal, en términos de computo.
- Los dispositivos de cómputo de múltiples procesadores son subutilizadas con *minibatches* muy pequeños.
- Debido a que los minibatches suelen procesarse en paralelo, un limitante es la memoria del dispositivo de cómputo.

Un detalle crucial de los *minibatch* es que sean independientes. Por lo tanto, las muestras de cada *minibatch* se escogen de forma aleatoria. En ocasiones, los datasets o conjuntos de datos, están organizados de forma que dos muestras consecutivas son parecidas, ya sea por el método que se obtuvo la muestra, la persona que tomó la medición, etc.

Estrictamente hablando, la estimación sin sesgo[1] de la función de costo - ecuación (2.2)), requiere que las muestras de entrenamiento no sean repetidas. Aunque, en la práctica, se repiten varias veces las muestras en el entrenamiento. Cuando una red neuronal es entrenada con todas muestras de entrenamiento una vez, se dice que se ha entrenado una época, o **epoch** en inglés. Únicamente la primera época calcula el gradiente de la función de costo *sin sesgo*, sin embargo, utilizar más épocas reduce el error de entrenamiento, aunque incrementa en menor cantidad la diferencia entre el error de entrenamiento y el error de prueba [9].

Minimizar la función de costo puede tener varias dificultades, entre ellas: que esté mal definida, que la función tenga varias zonas planas donde el gradiente es nulo, puntos de silla, que tenga un gradiente inestable[2], etc. Además, los algoritmos suelen tomar trayectorias subóptimas en el espacio de parámetros. Estos problemas pueden evitarse con una inicialización de los parámetros apropiada [9]. La inicialización de parámetros de una red neuronal es actualmente un problema abierto de investigación.

[1] El sesgo de un estimador es la diferencia entre el valor esperado del estimador y el parámetro que estima.

[2] El problema de desvanecimiento de gradiente ocurre cuando el gradiente de los parámetros es muy pequeño, lo cual dificulta emplear algoritmos de optimización como gradiente descendiente. También, existe un problema análogo cuando los gradientes son muy grandes, llamado problema de explosión de gradiente

En la práctica, más allá de minimizar la función de costo, buscamos obtener un error de generalización pequeño. Entonces, utilizamos aproximaciones del gradiente exacto para reducir el valor de la función de costo hasta obtener un error de entrenamiento o generalización pequeño, de forma que podemos utilizar la red neuronal para resolver un problema de forma satisfactoria.

2.2.2.2. Algoritmos para el entrenamiento de redes neuronales

Descenso de gradiente estocástico (GSD por sus siglas en inglés), y sus variantes son los algoritmos más utilizados para el entrenamiento de redes neuronales. Abajo, mostramos el algoritmo para actualizar los parámetros de una red neuronal utilizando el algoritmo de descenso de gradiente estocástico, algoritmo 1:

Algoritmo 1: Actualización del parámetro θ por **descenso de gradiente estocástico (GSD)** en la iteración k del entrenamiento

Entrada: *learning rate* ϵ_k
Entrada: Parametro inicial θ
mientras *criterio de detención no cumplido* **hacer**
  Tomar un minibatch con m muestras del conjunto de datos de entrenamiento $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ con sus correspondientes variables dependientes $\boldsymbol{y}^{(i)}$.
  Calcular la estimación del gradiente: $\hat{\boldsymbol{g}} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \boldsymbol{y}^{(i)}\right)$
  Actualizar el parámetro: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\boldsymbol{g}}$
  $k \leftarrow k + 1$

Cabe destacar que, para calcular los gradientes de una forma computacionalmente eficiente, se utiliza el algoritmo de **backpropagation**.

Cómo descenso de gradiente estocástico funciona tomando un conjunto de muestras de forma aleatoria del conjunto de datos de entrenamiento, la estimación de gradiente tiene ruido [9]. Es por ello por lo que en la práctica en lugar de usar un *learning rate* constante, se utiliza uno que disminuye conforme el entrenamiento procede. Escoger el valor del *learning rate* es más una arte que ciencia, entonces es recomendable graficar curvas de aprendizaje, como el valor de la función de costo y las iteraciones.

Si bien el descenso de gradiente estocástico y sus variantes son populares, el entrenamiento de la red neuronal en ocasiones es lento. Una extensión del algoritmo es usar **momento**, el cual utiliza los gradientes de las iteraciones anteriores. El nombre de momento proviene de una analogía con el momento lineal en mecánica clásica, el método de momento introduce una variable $\boldsymbol{v}$ que hace la tarea de la velocidad, $\boldsymbol{v}$ es la dirección y celeridad con la que los parámetros cambian en el espacio de los parámetros [9]. Un parámetro $\alpha \in [0, 1)$ determina que tan rápido las contribuciones de los gradientes anteriores decrecen. Las reglas para actualizar el valor del parámetro son:

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \boldsymbol{y}^{(i)}\right) \right)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$$

Mientras mayor sea α relativo a θ, los gradientes anteriores afectan en mayor medida al valor actual de θ. El algoritmo 2 debajo muestra el pseudocódigo asociado a este algoritmo de aprendizaje.

Algoritmo 2: Descenso de gradiente estocástico con momento

Entrada: *learning rate* ϵ, parametro de momento α
Entrada: Parametro inicial θ
mientras *criterio de detención no cumplido* **hacer**
  Tomar un minibatch con m muestras del conjunto de datos de entrenamiento $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ con sus correspondientes variables dependientes $\boldsymbol{y}^{(i)}$.
  Calcular la estimación del gradiente: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \boldsymbol{y}^{(i)}\right)$
  Calcular la actualización de la velocidad: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
  Actualizar el parámetro: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$

2.2.2.3. Inicialización de los parámetros de la red neuronal

Los algoritmos para el entrenamiento de redes neuronales son iterativos, por lo que necesitan de una configuración inicial de sus parámetros. Los valores iniciales de los parámetros pueden tener una gran influencia en la convergencia del método de entrenamiento. Es decir, algunas inicializaciones conducen a dificultades numéricas o incluso a fallar por completo. También la rapidez con la que el algoritmo converge depende de los parámetros iniciales.

La inicialización de parámetros sigue siendo un área de investigación activa. Usualmente, inicializamos a una constante escogida por una heurística, mientras que los pesos de la red son inicializados de acuerdo con valores muestreados aleatoriamente de una distribución Gaussiana o uniforme. La escala de la distribución tiene un efecto significativo sobre el entrenamiento de la red y su capacidad para generalizar [9].

2.2.2.4. Algoritmos con *learning rate* adaptativo

El *learning rate* es un parámetro muy importante para los algoritmos de entrenamiento de redes neuronales, por lo que se ha desarrollado algoritmos con *learning rates* que cambian en cada iteración.

Uno de los algoritmos con *learning rate* adaptativo más usados es **RMSprop**, algoritmo 3. RMSprop utiliza un promedio de los gradientes pasados que decae exponencialmente. De forma empírica ha mostrado ser un algoritmo eficiente y práctico para el entrenamiento de redes neuronales [9].

Otro algoritmo con *learning rate* adaptativo es **Adam**, algoritmo 4. Su nombre proviene de la frase "momentos adaptativos". A diferencia de RMSprop, donde se tiene un variable con el promedio decadente del cuadrado de los gradientes, Adam tiene una variable con el promedio decadente de los gradientes. Las estimaciones del primer y segundo momento requieren unas correcciones de sesgo.

Algoritmo 3: Algoritmo RMSprop

Entrada: *learning rate* ϵ, tasa de decaimiento ρ
Entrada: Parametro inicial θ
Entrada: Constante pequeña usada para estabilizar división por números pequeños δ, usualmente 10^{-6}
Inicializar las variables de acumulación $\boldsymbol{r} = 0$
mientras *criterio de detención no cumplido* **hacer**
  Mostrar un minibatch con m muestras del conjunto de datos de entrenamiento $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ con sus correspondientes variables dependientes $\boldsymbol{y}^{(i)}$.
  Calcular la estimación del gradiente: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \boldsymbol{y}^{(i)}\right)$
  Calcular el cuadrado del gradiente acumulado: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$
  Calcular la actualización del parámetro : $\boldsymbol{\Delta\theta} = -\frac{\epsilon}{\sqrt{\delta + r}} \odot \boldsymbol{g}$
  Actualizar el parámetro: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\Delta\theta}$

Algoritmo 4: Algoritmo Adam

Entrada: *learning rate* ϵ, usualmente 0.001
Entrada: Tasas de decaimiento para la estimación de momentos, ρ_1 y $\rho_2 \in [0, 1)$, usualmente 0.9 y 0.999 respectivamente.
Entrada: Constante pequeña usada para estabilizar división por números pequeños δ, usualmente 10^{-8}
Entrada: Parametro inicial θ
Inicializar las variables del primer y segundo momento $\boldsymbol{s} = 0$, $\boldsymbol{r} = 0$
Inicializar el paso de tiempo $t = 0$
mientras *criterio de detención no cumplido* **hacer**
  Tomar un minibatch con m muestras del conjunto de datos de entrenamiento $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ con sus correspondientes variables dependientes $\boldsymbol{y}^{(i)}$.
  Calcular la estimación del gradiente: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \boldsymbol{y}^{(i)}\right)$
  Actualizar estimación sesgada del primer momento: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\, \boldsymbol{g}$
  Actualizar estimación sesgada del segundo momento: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\, \boldsymbol{g} \odot \boldsymbol{g}$
  Corregir sesgo en el primer momento: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$
  Corregir sesgo en el segundo momento: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$
  Calcular actualización del parámetro: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{s}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$
  Actualizar el parámetro: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

2.2.2.5. Resumen del proceso de entrenamiento de una red neuronal

El proceso de entrenamiento de una red neuronal se basa en un algoritmo de optimización. El proceso de entrenamiento se resume en la figura 2.1.

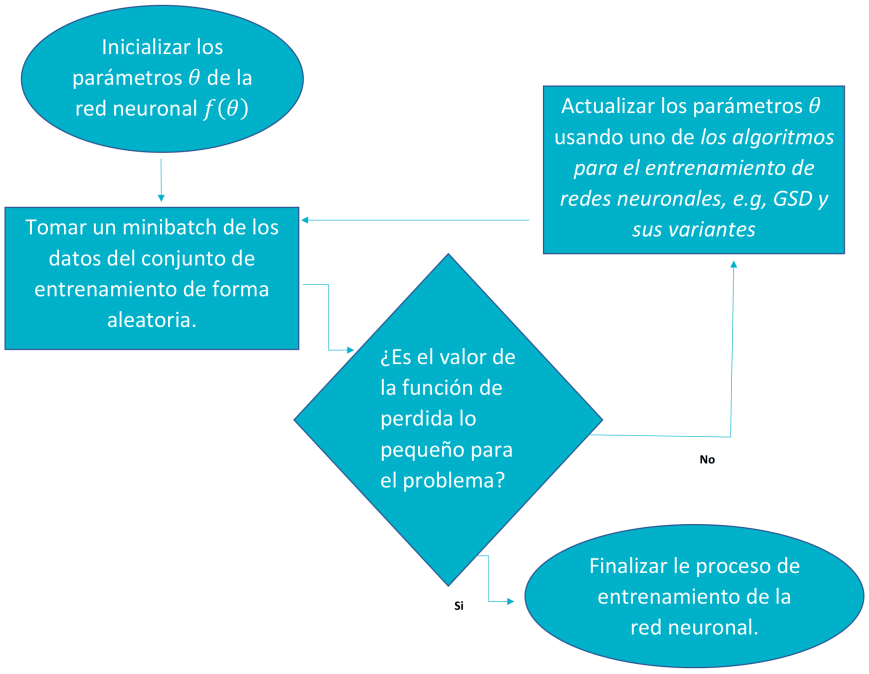

Figura 2.1: Diagrama de flujo del entrenamiento de una red neuronal genérica. El proceso de entrenamiento se basa en los algoritmos de optimización discutidos anteriormente. La inicialización de valores es particularmente importante para un correcto aprendizaje.

2.3. La arquitectura y función de pérdida de la GAN de Wasserstein

Una mejora sobre la GAN original de Goodfellow y col. [7] es utilizar la distancia de Wasserstein como función de costo [1]. Usar dicha distancia genera un gradiente que está mejor condicionado, resultando en mejores muestras creadas por el generador y un proceso de aprendizaje más estable que el de la GAN original.

2.3.1. Las distancias de Wasserstein

Introducimos esta distancia a través de un ejemplo. Considere el problema de transporte de bienes entre productores y consumidores, cuyas respectivas distribuciones espaciales están modelados por medidas de probabilidad. La distancia de Wasserstein representa el *costo mínimo* de transportar $\boldsymbol{x}$ a $\boldsymbol{y}$ para convertir $\mathbb{P}_r$ en $\mathbb{P}_g$.

De forma más matemática, sea (χ, d) un espacio métrico "Polaco"[3] y $p \in [1, \infty]$, la distancia de Wasserstein de orden p entre $\mathbb{P}_r$ y $\mathbb{P}_g$ se define por la fórmula [30].

$$W_p\left(\mathbb{P}_r, \mathbb{P}_g\right) = \left(\inf_{\pi \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \int_{\chi} d(x,y)^p d\pi(x,y)\right)^{\frac{1}{p}} \tag{2.3}$$

Donde:

- $\mathbb{P}_r$ es la distribución de los datos reales.
- $\mathbb{P}_g$ es la distribución de los datos generados por la GAN.
- $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ es el conjunto de distribuciones conjuntas $\pi(x, y)$, cuyas distribuciones marginales son $\mathbb{P}_r$ y $\mathbb{P}_g$ respectivamente [1].
- ínf es el ínfimo de $\Pi(\mathbb{P}_r, \mathbb{P}_g)$. También se le llama máxima cota inferior.

2.3.2. GAN utilizando las distancia de Wasserstein como función de costo

Una mejora sustancial de la GAN original es utilizar la distancia de Wasserstein W_1, también llamada distancia de movimiento de tierra, como función de costo. Esta idea fue propuesta por Arjovsky, Chintala y Bottou [1].

El caso particular W_1 para $p = 1$, la ecuación 2.3 tiene una expresión particular dada por la dualidad de *Kantorovich-Rubinstein* [1]:

$$W_1\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)] \tag{2.4}$$

Donde:

- sup es el supremo del conjunto de funciones 1-Lipschitz. Estas funciones cumplen la relación $\|f\|_L \leq 1$.

[3] Un espacio topológico que es homeomorfo a un espacio métrico completo que tiene un subconjunto denso contable

Sea $\{f_w\}_{w\in\mathcal{W}}$ una familia de funciones parametrizadas, el artículo de Arjovsky, Chintala y Bottou [1] propone resolver el siguiente problema:

$$\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}\left[f_w(x)\right] - \mathbb{E}_{z\sim p(z)}\left[f_w\left(g_\theta(z)\right)\right] \tag{2.5}$$

Donde:

- z es una variable aleatoria.
- $p(z)$ es una función de densidad de probabilidad.
- La función $f(x)$ es estimada por una red neuronal de la GAN. Dicha red neuronal realiza la tarea del discriminador [4].

Arjovsky, Chintala y Bottou [1] propone que para algún $w \in \mathcal{W}$, f_w es el elemento supremo en la ecuación 2.5, resultando en el cálculo de $W_1\left(\mathbb{P}_r, \mathbb{P}_\theta\right)$. Para cumplir la desigualdad de las funciones 1-Lipschitz, se recortan (*clipping* en inglés) los pesos w de f[1].

El algoritmo de la GAN de Wasserstein propuesta por Arjovsky, Chintala y Bottou [1] se muestra en 5. El diagrama de la GAN de Wasserstein penaliza se muestra en la figura 2.2.

Algoritmo 5: Algoritmo de WGAN. Valores por defecto: $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{critico} = 5$

Entrada: *learning rate* α, el parámetro de recorte (gradiente clipping) c, el tamaño del minibatch m, y el número de iteraciones del crítico por iteración del generador $n_{crítico}$
Entrada: Parámetros iniciales del crítico w_0, parámetros iniciales del generador θ_0
mientras *criterio de detención no cumplido* **hacer**
  para $t = 0, \ldots, n_{crítico}$ **hacer**
    Tomar un minibatch de los datos reales: $\left\{x^{(i)}\right\}_{i=1}^{m} \sim \mathbb{P}_r$
    Tomar un minibatch del espacio latente: $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w\left(x^{(i)}\right) - \frac{1}{m}\sum_{i=1}^{m} f_w\left(g_\theta\left(z^{(i)}\right)\right)\right]$
    $w \leftarrow w + \alpha \cdot \mathrm{RMSProp}\left(w, g_w\right)$
    $w \leftarrow \mathrm{clip}(w, -c, c)$
  Tomar un minibatch del espacio latente: $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w\left(g_\theta\left(z^{(i)}\right)\right)$
  $\theta \leftarrow \theta + \alpha \cdot \mathrm{RMSProp}\left(\theta, g_\theta\right)$

[4] Debido a que da un valor numérico en lugar de una probabilidad, le llaman *crítico* en vez de *discriminador*

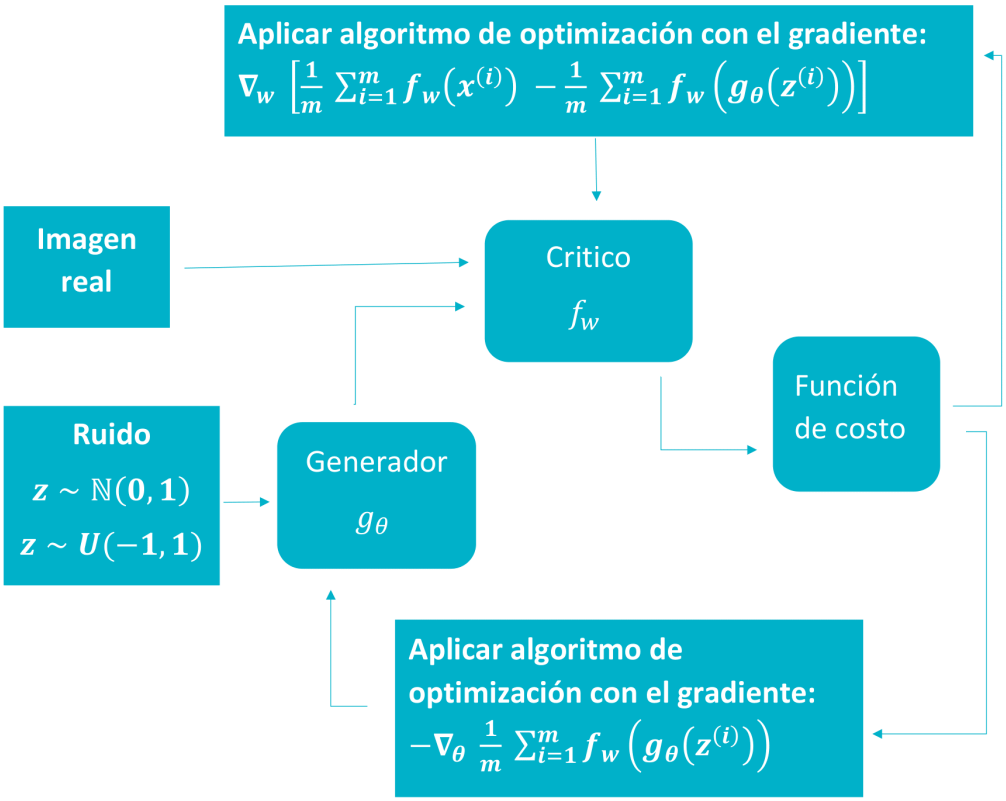

Figura 2.2: Diagrama de la GAN de Wasserstein o WGAN.

2.3.3. Mejora a la GAN de Wasserstein: penalización de gradiente

Una delicadeza de la GAN propuesta por Arjovsky, Chintala y Bottou [1] es la forma en que fuerza la limitación de funciones 1-Lipschitz. Entonces, Gulrajani y col. [10] propone sumar el siguiente término a la función de costo, ecuación 2.4, para penalizar si la norma del gradiente cuando se aleja de la unidad.

$$\lambda \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[(||\nabla_{\hat{x}} f(\hat{x})||_2 - 1)^2\right] \tag{2.6}$$

Donde $\mathbb{P}_{\hat{x}}$ son muestras uniformes de puntos sobre líneas que unen a $\mathbb{P}_r$ con $\mathbb{P}_{\hat{g}}$ Gulrajani y col. [10]. En otras palabras, cada $\hat{x} \in P_{\hat{x}}$ es una combinación lineal de los datos reales y generados con pesos aleatorios.

El algoritmo de la GAN de Wasserstein con penalización de gradiente propuesta por Gulrajani y col. [10] se muestra en 6. El diagrama de la GAN de Wasserstein con penalización de gradiente penaliza se muestra en la figura 2.3.

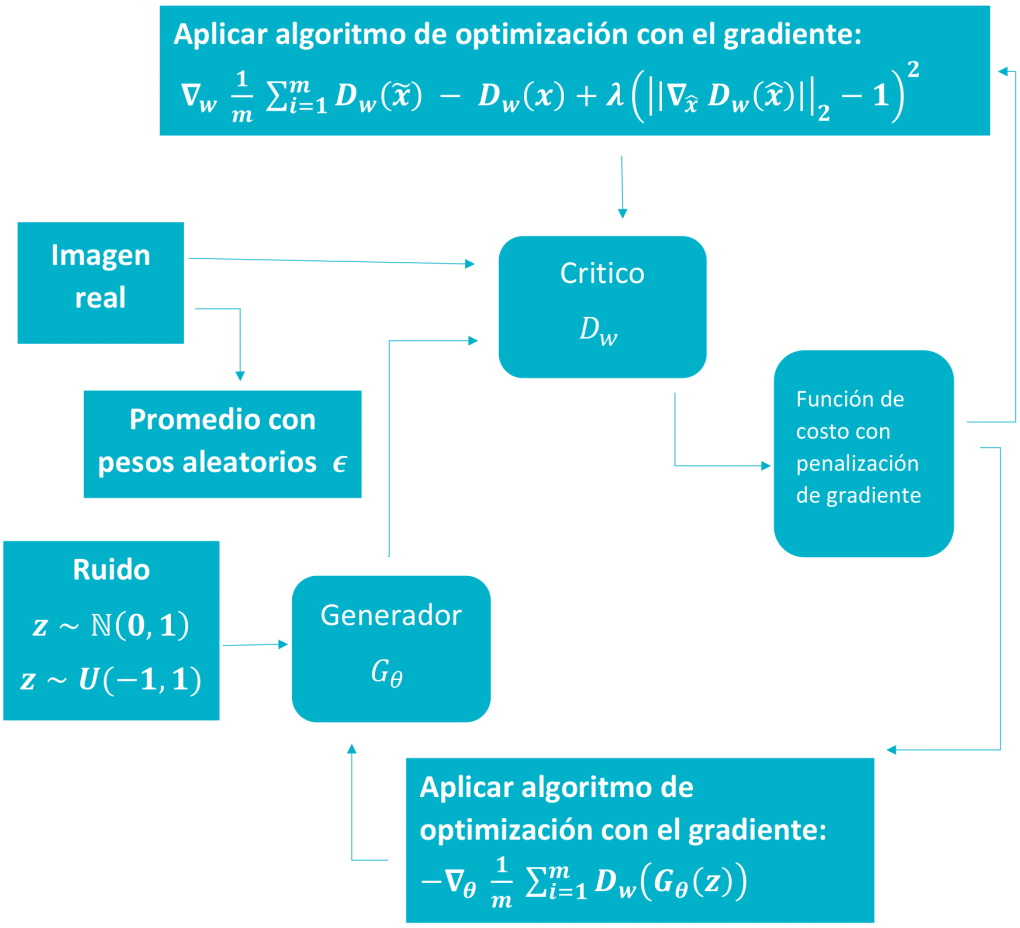

Figura 2.3: Diagrama de la GAN de Wasserstein con penalización de gradiente o WGAN-GP.

Algoritmo 6: Algoritmo de WGAN con penalización de gradiente. Valores por defecto: $\alpha = 0.0001$, $c = 0.01$, $m = 64$, $n_{critico} = 5$, $\lambda = 10$, $\beta_1 = 0$, $\beta_2 = 0.9$

Entrada: Coeficiente de penalización de gradiente: λ, número de iteraciones del crítico por iteración del generador $n_{crítico}$, tamaño del minibatch m, hiper-parámetros del optimizador Adam α, β_1, β_2
Entrada: Parámetros iniciales del crítico w_0, parámetros iniciales del generador θ_0
mientras θ *no ha convergido* **hacer**
  para $t = 0, \ldots, n_{crítico}$ **hacer**
    para $t = 0, \ldots, n_{crítico}$ **hacer**
      Tomar una muestra de los datos reales: $\left\{x^{(i)}\right\}_{i=1}^{m} \sim \mathbb{P}_r$
      Tomar una muestra del espacio latente: $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$
      Muestrear un número aleatorio: $\epsilon \sim U[0, 1]$
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda \left(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1\right)^2$
    $w \leftarrow \text{Adam}\left(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2\right)$
  Tomar un minibatch del espacio latente: $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$
  $\theta \leftarrow \text{Adam}\left(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w\left(G_\theta(\boldsymbol{z})\right), \theta, \alpha, \beta_1, \beta_2\right)$

2.4. Datos sintéticos usados para la validación: distribución Dirichlet

Para generar datos "sintéticos" de la composición de comunidades microbianas utilizamos la distribución de Dirichlet. Las muestras de datos de una distribución Dirichlet son un caso de *datos composicionales*, es decir que cumplen $\sum_{i=0}^{K} x_i = 1$, al igual que los datos de abundancias relativas. Por lo tanto, usamos un conjunto de datos de muestras de una distribución de Dirichlet.

2.5. Descripción de los datos de abundancias de especies en microbiomas

Los datos con los que estamos trabajando son abundancias relativas de las especies en un microbioma, por lo que son datos composicionales. Las abundancias se encuentran usualmente en la literatura como en la figura 2.4. Entonces podemos representar las abundancias como un vector $\hat{v} \in \mathbb{R}^n$ donde n es el número de especies del microbioma.

Entonces, si apilamos m vectores de abundancias obtenemos una matriz de dimensiones $m \times n$.

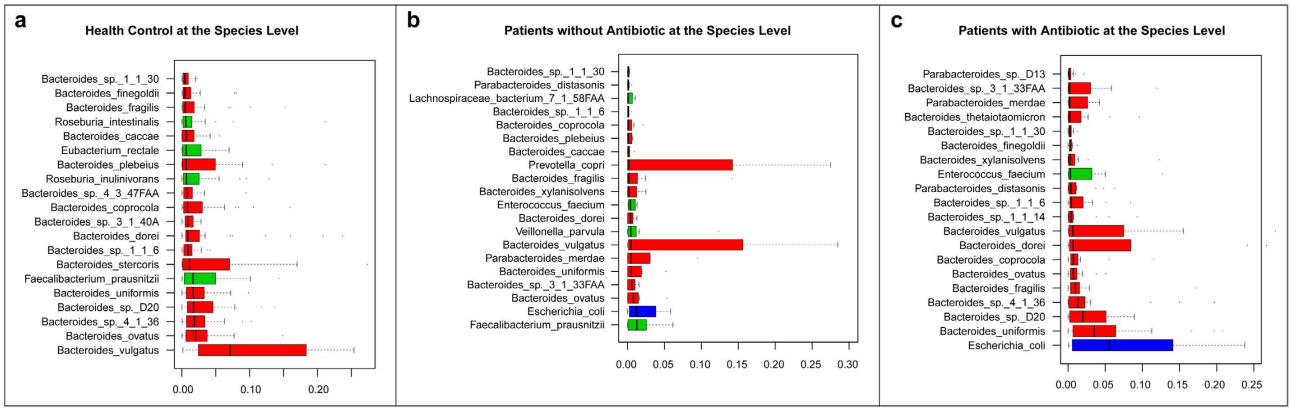

Figura 2.4: Perfiles de abundancias de especies en microbiomas en personas infectadas con H7N9, tomadas del artículo Qin y col. [25]

2.6. Medidas de similitud entre distribuciones de probabilidad: Distancia Jensen-Shannon

Las medidas de distancia o similitud son esenciales para resolver muchos problemas de reconocimiento de patrones, como la clasificación y la agrupación. Varias medidas de distancia 0 similitud están disponibles en la literatura para comparar dos distribuciones de datos. Como sugieren los nombres, una similitud mide qué tan cerca están dos distribuciones [17].

La distancia de Jensen-Shannon entre dos distribuciones de probabilidad discretas P y Q está dada por la ecuación 2.7

$$JSD(P||Q) = \sqrt{\frac{D_{KL}(P||M) + D_{KL}(Q||M)}{2}} \tag{2.7}$$

$$M = \frac{1}{2}(P + Q)$$

Y D_{KL} es la divergencia Kullback-Leibler, la cual es una medida de similitud entre dos distribuciones de probabilidad discreta. La ecuación de la divergencia Kullback-Leibler está dada por la ecuación 2.8

$$D_{KL}(p||q) = -\sum_{i=1}^{N} p(x_i) \log \frac{q(x_i)}{p(x_i)} \tag{2.8}$$

Como ejemplo del uso de la distancia Jensen-Shannon, considere dos conjuntos de datos obtenidos de muestrear un par de distribuciones Gaussianas, G_1 y G_2 con las características de la tabla 2.1.

Gaussiana	μ	σ	p
G_1	0.8	0.25	[0.00145137, 0.00467665, 0.01612638, 0.03128518, 0.04176733, 0.03547804, 0.01935166, 0.00806319, 0.00306401]
G_2	2	1	[0.00391961, 0.00951905, 0.01175882, 0.01455854, 0.0179182 , 0.0179182 , 0.02743725, 0.03023697, 0.02799719]

Tabla 2.1: Datos de las distribuciones Gaussianas.

Con los vectores de probabilidad podemos calcular la distancia Jensen Shannon entre G_1 y G_2, dando como resultado **0.314**. Una visualización de ambas distribuciones se puede observar en la figura 2.5. Las gráficas y datos de la tabla 2.1 se pueden reproducir con esta libreta Jupyter.

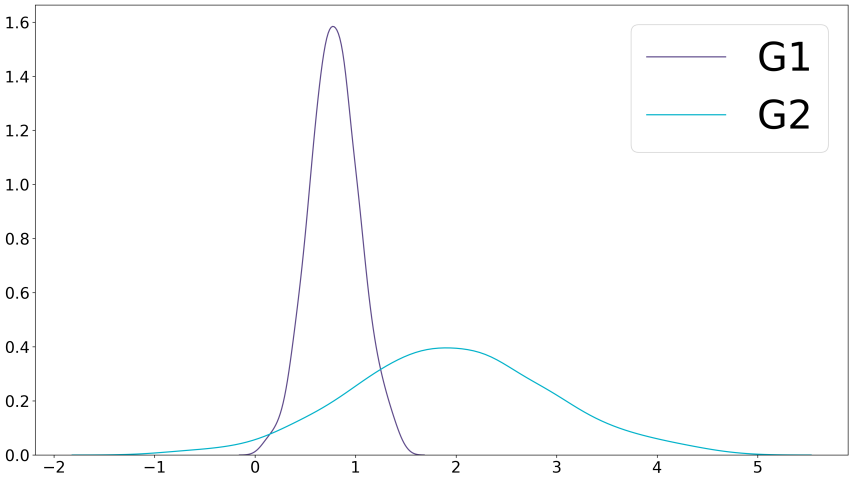

Figura 2.5: Gráfico de G_1 y G_2

2.7. Misceláneo

2.7.1. Funcion Softmax

La ecuación 2.9 es la función Softmax, también llamada función exponencial normalizada. Como el nombre sugiere, una característica de esta función es que su rango es el intervalo (0,1). Debido a que se normaliza y la función exponencial no produce números negativos. Entonces, el resultado de aplicar la función Softmax a un vector $\hat{z}$ de K elementos, es una distribución de probabilidad.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ para } i = 1, \ldots, K \text{ y } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K \tag{2.9}$$

La función Softmax se utiliza en varios métodos de clasificación multiclase.

Capítulo 3

Objetivos e hipótesis

3.1. Objetivos Generales

Diseñar un modelo generativo que, a partir de una colección de muestras con la abundancia de especies de una comunidad microbiana, permita generar «muestras sintéticas» adicionales que son estadísticamente indistinguibles de las originales.

3.1.1. Objetivos Específicos

Para construir el modelo generativo, proponemos utilizar «Redes Generativas Adversarias» (GAN, por sus siglas en inglés), las cuales han sido aplicadas exitosamente en problemas de modelado generativo muy complejos como la generación de rostros humanos 1.5. En particular, los objetivos específicos del trabajo son:

1. Construir la arquitectura de una GAN que permita aprender la distribución de abundancias de especies de una comunidad microbiana basándose en un conjunto de muestras de subcomunidades. Este paso involucra el diseño adecuado de los dos componentes centrales de una GAN: el generador y el discriminador.
2. Realizar una validación del modelo generativo propuesto utilizando muestras sintéticas creadas a partir de modelos matemáticos estándar de comunidades microbianas.
3. Discutir su potencial aplicación a datos experimentales, sus ventajas y sus limitaciones.

3.2. Hipótesis

Si la arquitectura del generador y discriminador es la adecuada, entonces la GAN puede aprender a generar muestras de datos composicionales estadísticamente indistinguibles de las muestras verdaderas. Esto debería ser posible utilizando un menor número de datos de entrenamiento con respecto a otros métodos más clásicos para construir modelos generativos.

Capítulo 4

Metodología

Resumen

Las arquitecturas de GANs existentes se han enfocado principalmente en la generación de imágenes. Para crear una GAN que genere datos composicionales, partimos de una GAN de Wasserstein con penalización de gradiente. Sin embargo, la GAN necesita cambios en la capa de salida para que pueda generar datos composicionales.

Un área abierta de investigación es la evaluación cuantitativa de modelos generativos [8], y por ende las GANs. Para la generación de imágenes, Salimans y col. [26] propone una métrica denominada *inception score*, la cual depende de un clasificador de imágenes. Para evaluar la calidad y proceso de aprendizaje de una distribución de datos composicionales de una GAN, en este capítulo proponemos una metodología análoga.

Como primera prueba de la capacidad de aprender una distribución de datos composicionales de nuestra GAN, entrenamos la GAN con un conjunto de datos de una distribución de Dirichlet de tres y cien especies. Los datos muestreados de una distribución de Dirichlet tienen la característica de que son composicionales. Evaluamos el aprendizaje de la GAN con la metodología introducida en la sección 6.1 y 4.2.2 de este capítulo. Para el caso de datos composicionales con tres especies, utilizamos diagramas ternarios como una forma de visualización de los datos, el uso de diagramas ternarios es detallado en la sección 4.2.4 .

4.1. Modificaciones a la arquitectura de GAN de Wasserstein

La investigación de GANs se ha enfocado principalmente a la generación de imágenes. Para crear una GAN que genere datos composicionales, partimos de una GAN de Wasserstein con penalización de gradiente. Sin embargo, la GAN necesita cambios en la capa de salida para que pueda aprender a generar datos composicionales.

Originalmente, Arjovsky, Chintala y Bottou [1] utilizan dos arquitecturas distintas para la GAN de Wasserstein. La primera es un perceptrón multicapa con 4 capas ocultas de 512 nodos cada una. La segunda llamada DCGAN, una red neuronal convolucional de 4 capas que carece de capas totalmente conectadas y que utiliza capas de agrupamiento (*pooling layer*). Estas dos arquitecturas están diseñadas para funcionar con imágenes y generar imágenes, por lo que requiere modificar la capa de salida para generar datos los composicionales usados para microbiomas.

La GAN que proponemos es capaz de aprender una distribución de datos composicionales en general, no obstante, enfocamos la GAN a generación de datos composicionales de microbioma de suelos.

4.1.1. Modificación en la capa de salida del generador

Partimos de una GAN de Wasserstein con penalización de gradiente, propuesta por Gulrajani y col. [10]. En particular la GAN de Wasserstein con penalización de gradiente que utilizamos es un par de perceptrones multicapa. Los datos composicionales forman una distribución de probabilidad, por lo tanto utilizamos como capa de salida una función de activación *Softmax*. De esta forma, el generador produce datos de una distribución de probabilidad. En la figura 4.6 se muestra un esquema con la arquitectura propuesta de la GAN para generar muestras de microbioma.

4.2. Validación de la GAN

4.2.1. Cuantificando el aprendizaje de GAN

En general, un problema muy difícil es cuantificar la calidad de los datos generados por una GAN. Aquí presentamos la metodología que utilizamos para evaluar y cuantificar el aprendizaje de una distribución de datos composicionales de nuestra GAN. Intuitivamente, comparamos qué tan parecidos son los histogramas de frecuencia de abundancia de cada especie generados por la GAN con respecto a histogramas de frecuencia de abundancia de los datos reales.

Para evaluar el aprendizaje de la GAN utilizamos la distancia Jensen-Shannon introducida en la sección 2.6. Sin embargo, la distancia Jensen-Shannon puede ser reemplazada por otra *medida de similitud* en la ecuación 4.1.

$$S = \exp\left\{ \frac{1}{N} \sum_{k=1}^{N} \mathrm{JSD}(\hat{\mathbb{P}}_k || \mathbb{P}_k) \right\} \tag{4.1}$$

Donde:

- N : número total de especies.
- $\hat{\mathbb{P}}_k$: es la distribución empírica producida por la abundancia de la especie k-ésima utilizando datos reales.
- $\mathbb{P}_k$: es la distribución empírica producida por la abundancia de la especie k-ésima utilizando datos generados por la GAN

Al error descrito por las ecuaciones 4.1 lo llamaremos **error composicional agregado**; agregado porque valora a todas las especies. Lo que hace la ecuación 4.1 es comparar los histogramas producidos por las abundancias de especie de los datos generados por la GAN ($\mathbb{P}_k$) contra los datos reales ($\hat{\mathbb{P}}_k$).

Ilustraremos esta metodología utilizando datos sintéticos generados por una distribución de Dirichlet para tres especies. La figura 4.1 muestra histogramas de abundancia relativa para cada especie de un conjunto de datos que es una combinación de tres distribuciones de Dirichlet, todas de tres especies. Son este tipo de histogramas los datos que utiliza la ecuación 4.1. Para fines ilustrativos, utilizamos 10 divisiones (*bins*). De los histogramas de la figura 4.1, podemos construir la tabla 4.1 de distancias de Jensen-Shannon. Las distancias son promediadas y exponenciadas de acuerdo a la ecuación 4.1.

Especie	p	q	$JSD(p\|\|q)$
1	(0.39, 0.11, 0.08, 0.07, 0.06, 0.06, 0.06, 0.06, 0.1)	(0.41, 0.1 , 0.08, 0.07, 0.06, 0.06, 0.05, 0.06, 0.11)	0.0239
2	(0.5, 0.1 , 0.08, 0.06, 0.05, 0.05, 0.05, 0.05, 0.07)	(0.52, 0.11, 0.08, 0.06, 0.05, 0.05, 0.04, 0.05, 0.06)	0.0235
3	(0.33, 0.11, 0.08, 0.07, 0.06, 0.06, 0.07, 0.07, 0.13)	(0.33, 0.09, 0.07, 0.07, 0.06, 0.06, 0.06, 0.08, 0.16)	0.0239

Tabla 4.1: Ejemplos calculo JSD

Figura 4.1: Histogramas de la abundancia del conjunto de datos K3. (A) Histogramas de las abundancias para la primera especie. La distancia Jensen-Shannon entre el histograma de la primera especie de datos reales contra el histograma de la primera especie de los datos generados es de 0.0239, el cual es confirmado de forma gráfica en los histogramas. (B) Histogramas de abundancia relativa de la segunda especie. La GAN genera muestras donde la distribución de la segunda especie es similar a la distribución de la segunda especie de los datos reales. (C) Histogramas de abundancia relativa para la tercera especie. De igual, la GAN aprende a generar datos de la especie tres, confirmado que la GAN puede aprender datos composicionales multidimensional, vea la tabla 4.1. Otra comprobación de la capacidad de aprendizaje dela GAN incluso en escenarios más complejos es usando diagramas ternarios, un ejemplo detallado de estos diagramas se encuentra la figura 4.5.

4.2.1.1. División de datos

Generamos datos a partir de muestrear datos de una distribución de Dirichlet utilizando los parámetros α del csv del anexo. No se muestra en el documento porque la tabla tiene 100 columnas y 10 filas.

Para cada parámetro α, se generó 60,000 muestras, las cuales se dividieron en 2 conjunto de datos. Uno conjunto de datos con las primeras 50,000 muestras para el entrenamiento de la red y un conjunto de datos de prueba con las 10,000 últimas muestras.

4.2.2. Desempeño del proceso de aprendizaje

Para evaluar el desempeño del proceso de aprendizaje de la GAN de forma adecuada, entrenamos de 5 a 10 veces la misma red neuronal con la misma cantidad de datos. Cada una de estas repeticiones los perceptrones tienen unos pesos iniciales distintos. Después del entrenamiento, calculamos el error estándar de la media del error composicional agregado, el cual se grafica como barras de error encima de la gráfica de error composicional. De forma alternativa también podemos calcular la desviación estándar del error composicional agregado. Un ejemplo de las barras de error encima de una gráfica de error composicional se muestra en la figura 4.2.

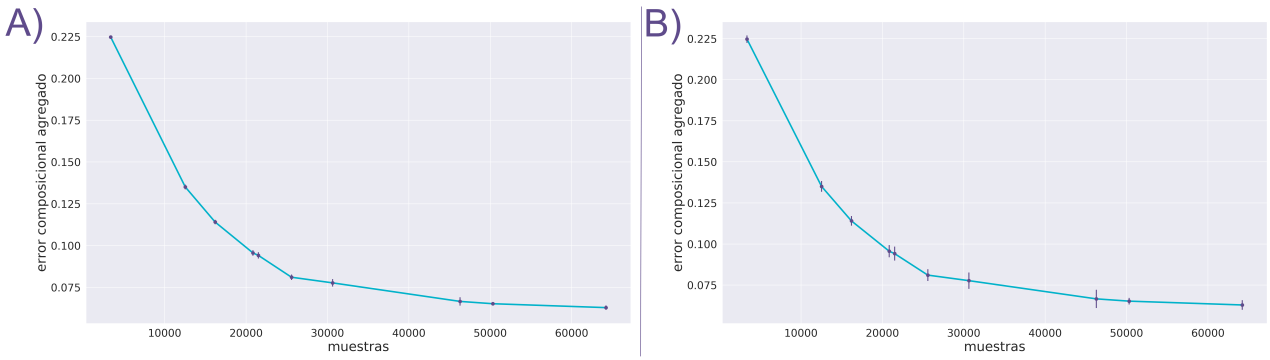

Figura 4.2: Barras de error como visualización de la variabilidad del proceso del aprendizaje de la GAN. **A)** Calculamos el error estándar del error composicional agregado con 5 repeticiones para cada cantidad de muestras. A pesar de usar únicamente 5 repeticiones se obtuvo un error estándar con media de 0.0015. **B)** Calculamos la desviación estándar del error composicional agregado con 5 repeticiones para cada cantidad de muestras encima de la misma gráfica de error composicional agregado.

4.2.3. Entrenamiento de la GAN

El entrenamiento de los perceptrones que forman la GAN se realizó de la forma general descrita en la sección 2.2.2. Como sugiere Arjovsky, Chintala y Bottou [1], Gulrajani y col. [10], se entrenó 5 veces más el crítico que al generador.

Para entrenar la GAN utilizamos un *learning rate* constante de 10^{-5}. Utilizar *learning rates* variables tiene resultados negativos, los cuales se detallan en la sección 5.1.4.

Para evaluar el desempeño del proceso de aprendizaje (descrito a detalle en la sección 4.2.2), tenemos que entrenar varias veces la misma red, lo cual llamamos hacer repeticiones. Como buscamos determinar cómo varía el aprendizaje de la GAN en función de la cantidad de datos, creamos subconjuntos del conjunto de datos de entrenamiento con las cantidades de muestras de la tabla 4.2. Como realizamos 5 repeticiones por punto, como en las figuras 5.2, 5.6, y 5.3, creamos 50 subconjuntos del conjunto de datos del entrenamiento y entrenamos la misma red 50 veces con diferentes conjunto de datos. Encima de gráficas se exhibe la variabilidad del proceso de aprendizaje usando el error estándar de la media del error agregado composicional para cada punto, siendo representado por las barras de error.

De forma específica, hacemos subconjuntos con diferentes cantidades de datos de acuerdo con la tabla 4.2. Por ejemplo, para el subconjunto #1 tomamos de forma aleatoria 4529 muestras de nuestro set de entrenamiento de 50,000 muestras. Hacemos 4 repeticiones para tener 5 puntos del rendimiento de la GAN entrenada con 4529 muestras. Todos estos subconjuntos de 4529 muestras son diferentes y se guarda un registro de los índices por cuestiones de reproducibilidad.

Subconjunto #	Cantidad de datos
1	4529
2	6386
3	9812
4	12738
5	15255
6	23640
7	27881
8	35261
9	43791
10	49478

Tabla 4.2: Tabla de muestras usadas en cada subconjunto. Para cada subconjunto, se hicieron 5 variantes con la misma cantidad de datos, pero diferentes muestras

Un diagrama el proceso de entrenamiento se muestra en la figura 4.3.

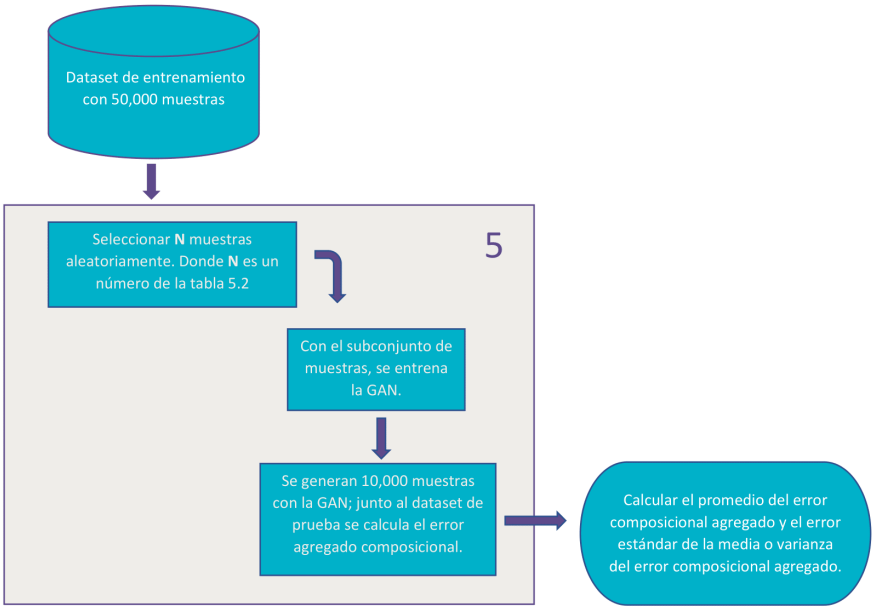

Figura 4.3: Diagrama del procedimiento para calcular la variabilidad del entrenamiento. El proceso permite calcular el error composicional agregado promedio y el error estándar de la media de éste. Este par de números nos permite graficar el error agregado composicional contra muestras como las de las figuras 5.2, 5.6, y 5.3, las cuales permiten evaluar el aprendizaje de la GAN y su variabilidad.

4.2.4. Visualización del aprendizaje para un comunidad de tres especies

Un diagrama ternario es un diagrama triangular que muestra la proporción de tres variables que suman una constante y que lo hace mediante coordenadas baricéntricas [27]. Los ejes de coordenadas de dicho diagrama se muestran en la figura 4.4, donde cada uno de los ejes x, y, y z están escalados para que $0 \leq x, y, z \leq 1$.

Una forma de visualizar datos composicionales en el caso de tres especies es usando un *diagrama ternario*, como los que se muestran en la figura 4.5.

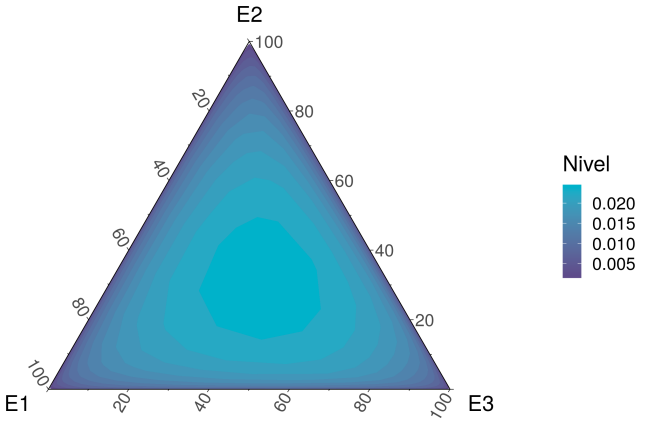

Figura 4.4: Diagrama ternario de datos de una distribución de Dirichlet. El nivel representa una estimación de la función de densidad de probabilidad. Los datos se les aplica el isomorfismo de transformación isométrica de la relación logarítmica (ilr por sus siglas en inglés, isometric log ratio). Este isomorfismo $ilr : S^D \rightarrow \mathbb{R}^{D-1}$ permite aplicar métodos estadísticos convencionales a datos composicionales. Los datos de éste diagrama ternario son los correspondientes a la distribución de Dirichlet con α_1 descritos en la sección 4.2.1.1

4.2.5. Efecto de la cantidad de distribuciones de Dirichlet combinadas

Para construir conjunto de datos más complejos, realizamos combinaciones lineales de datos de diferentes conjuntos de datos de distribuciones de Dirichlet con diferentes parámetros α de la siguiente forma:

$$K_n^* = \lambda_1 K_1 + \lambda_2 K_2 + \lambda_3 K_3 + \ldots + \lambda_n K_n \tag{4.2}$$

Donde:

$$\lambda_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_n = 1, \quad \lambda_i > 0 \tag{4.3}$$

Nótese que el conjunto de datos K_n^* resultante conserva las propiedades de datos composicionales.

La GAN puede aprender a generar datos de combinaciones de distribuciones de Dirichlet de tres especies, como se observa en los diagramas ternarios de la figura 4.5, además de los bajos valores error composicionales agregado. Los resultados para comunidades más grandes de detallan en la sección 5.1.5 y en la figura 5.6.

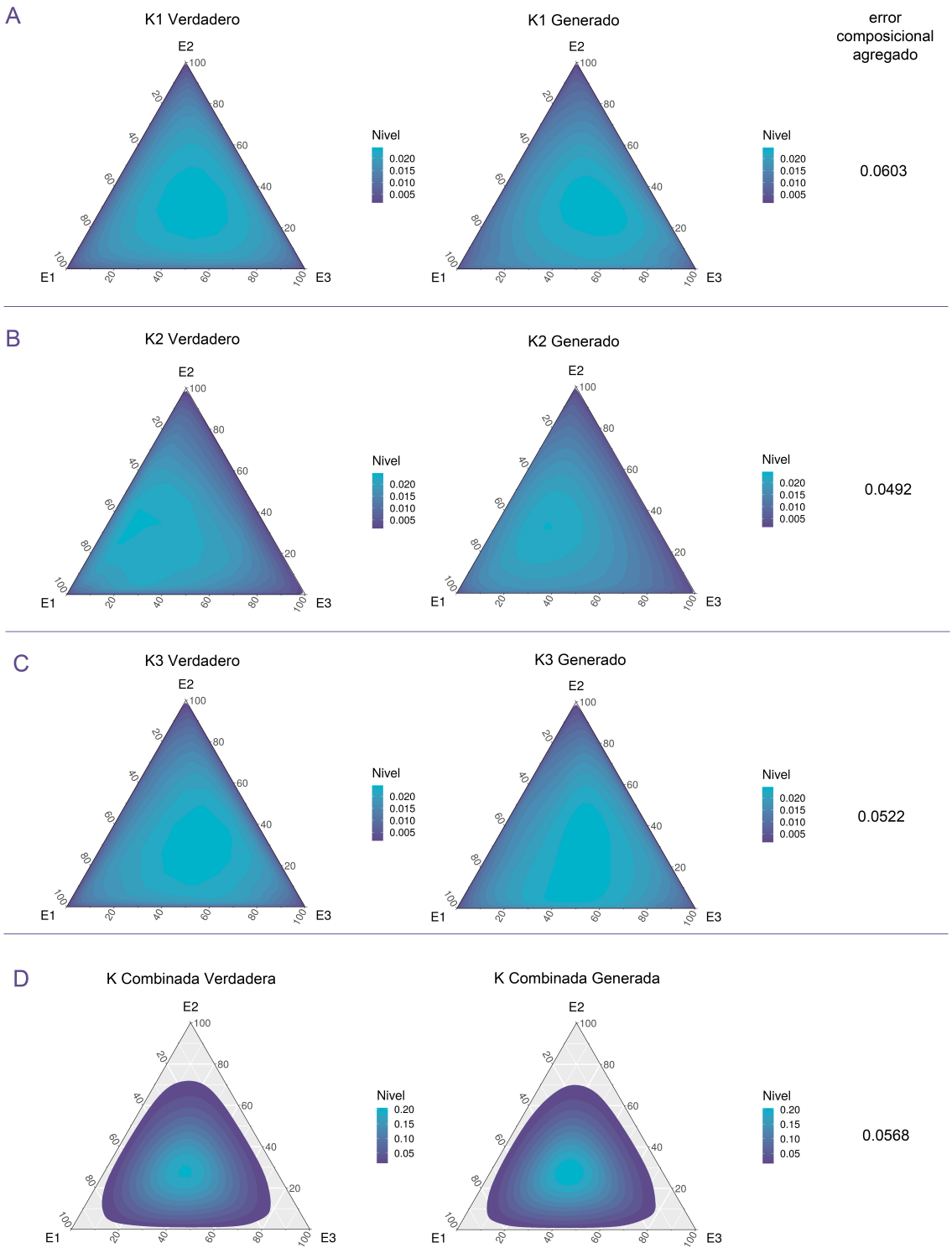

Figura 4.5: Diagrama de ternario de distintas distribuciones de Dirichlet.Para la estimación de densidad de kernel, a los datos se les aplica una *transformación isométrica de la relación logarítmica* (ILR por sus siglas en inglés), por lo que a mayor valor de ILR, menor cantidad de puntos existen. (A) Para generar los datos, se utilizó un vector $\alpha_1(0.33582739, 0.26988812, 0.39428449)$. (B) Para generar los datos, se utilizó un vector $\alpha_2(0.50053169, 0.25917199, 0.24029632)$. (C) Para generar los datos, se utilizó un vector $\alpha_3(0.37175041, 0.24177349, 0.3864761)$. (D) Los datos son la combinación de los datos en (A),(B) y (C) usando la ecuación 4.2 con coeficientes $\lambda = \frac{1}{3}$

4.2.6. Arquitecturas del generador de la GAN

En general, para todas las arquitecturas usamos un número de nodos mayor que el número de especies. Adicionalmente a las arquitecturas mostradas en la figura 4.6, se probaron arquitecturas con funciones de activación Sparsemax, y diferentes cantidades de nodos, sin embargo no tuvieron buen rendimiento como se muestra en las figuras 5.2 y 5.3.

1. **Arquitectura plana**, figura 4.6 (A). Esta arquitectura es caracterizada por las capas ocultas tienen el mismo número de nodos. Las capas ocultas siguen la secuencia: 2 DIM, 2 DIM, 2 DIM, donde *DIM* equivale a 512 nodos.

2. **Arquitectura rombo**, figura 4.6(B). Esta arquitectura es caracterizada porque los nodos de las capas ocultas forman un romboide. Las capas ocultas siguen la secuencia: 1 DIM, 2 DIM, 2 DIM, donde *DIM* equivale a 512 nodos.

3. **Arquitectura moño**, figura 4.6 (C). Esta arquitectura se caracteriza por que el número de nodos en la mitad de las capas ocultas se reduce y se incrementa en la segunda mitad. El resultado es que los nodos de las capas ocultas forman un rombo. Las capas ocultas siguen la secuencia: 3 DIM, 2 DIM, 1 DIM, 2 DIM, 3 DIM, donde *DIM* equivale a 512 nodos.

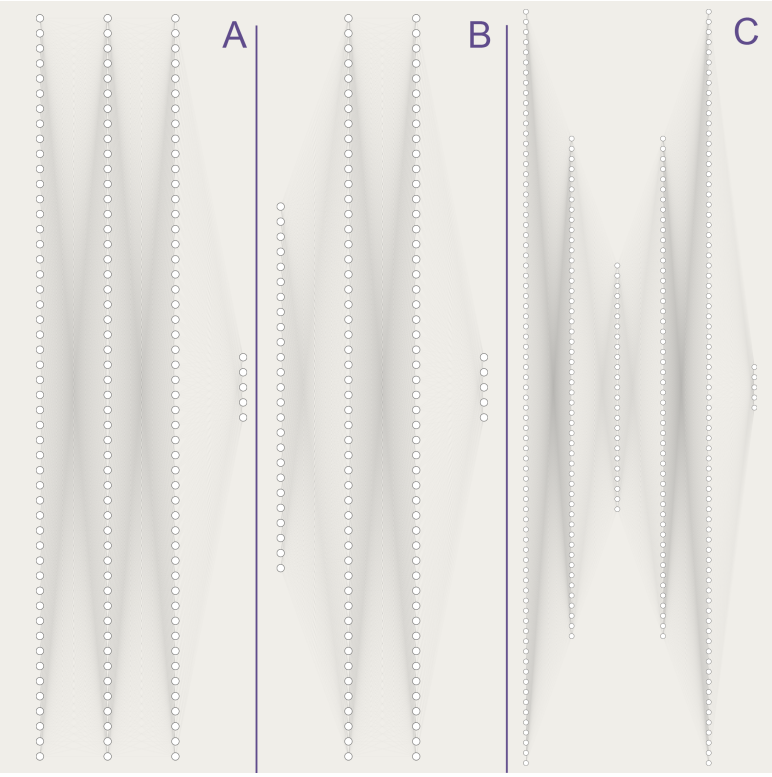

Figura 4.6: Diagrama de la distintas arquitectura. **(A)** Esta arquitectura es caracterizada porque las capas ocultas tienen el mismo número de nodos. **(B)**. Esta arquitectura es caracterizada porque los nodos de las capas ocultas forman un romboide. **(C)** Esta arquitectura se caracteriza porque el número de nodos en la mitad de las capas ocultas se reduce y se incrementa en la segunda mitad. El resultado es que los nodos de las capas ocultas forman un rombo.

Capítulo 5

Resultados

Resumen

El utilizar funciones de activación Softmax en el generador perceptrón multicapa tiene un rendimiento visiblemente mejor a un generador perceptrón multicapa convencional que utiliza funciones de activación ReLu.

Un generador con mayor cantidad de nodos obtuvo un rendimiento similar a las arquitecturas más complejas propuestas en la sección 4.2.6 y algunas arquitecturas adicionales. Aun cuando el error agregado composicional de las diferentes arquitecturas está en el mismo orden de magnitud, la arquitectura de mayor cantidad de nodos tiene un error visiblemente menor en un régimen de menos de 20000 muestras.

Conforme se incrementa la complejidad del conjunto de datos se vuelve más relevante el *learning rate* para un aprendizaje correcto. En específico, un *learning rate* que funciona adecuadamente para un conjunto de datos provoca un aprendizaje errático en otro conjunto de datos. Es por ello que los algoritmos de *learning rate* resultaron inoportunos.

Por otra parte, la GAN entrenada con conjunto de datos más complejos tienen menor error composicional agregado, no obstante su entrenamiento es más lento por los *learning rate* utilizados.

El error composicional agregado se puede ajustar a una curva en función del número de muestras. En particular se propone un modelo exponencial de un parámetro. Una de sus ventajas es que el coeficiente de dicha exponencial tiene una relación lineal con el número de distribuciones de Dirichlet combinadas. Un modelo de más parámetros requeriría una cantidad horrorosa de datos de entrenamiento y cómputo.

Finalmente, utilizamos la GAN para generar datos experimentales de microbioma de suelo a distintos niveles taxonómicos. Para evaluar la utilidad de la GAN, usamos una distribución de Dirichlet como modelo de los datos experimentales, y discutimos sus ventajas.

5.1. Resultados de la validación numérica

5.1.1. Comparación con perceptrón multicapa convencional

Como primera validación de la GAN, el objetivo es mostrar el rendimiento superior de usar funciones de activación *Softmax* propuestas en la sección 4.1.1. Si bien la función Softmax es usada en clasificadores multiclase, las GAN son un modelo generativo.

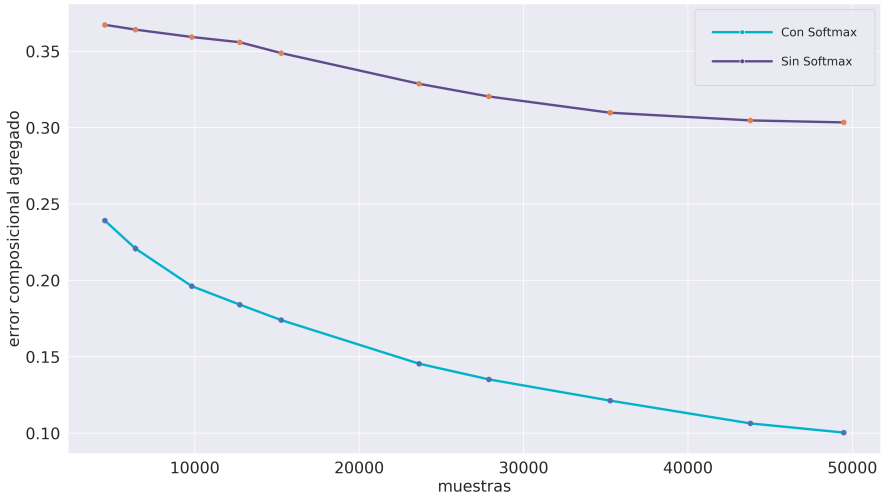

Figura 5.1: Error composicional agregado de la GAN utilizando con y sin activaciones Softmax. Se entrenó dos generadores, ambos son perceptrones multicapa de 4 capas donde la diferencia es que una tiene una capa de salida con una función de activación Softmax y la otra con una función de activación ReLu.

5.1.2. Aprendizaje de las arquitecturas

Una forma detallada de valorar la eficiencia del aprendizaje de la GAN respecto a las muestras es graficando el *error composicional agregado*, introducido en la sección 6.1 contra el número de muestras como en la figura 5.2.

Entrenamos las arquitecturas descritas en la sección 4.2.6 con un conjunto de datos de una distribución de Dirichlet de 100 especies. Además, añadimos una arquitectura *Sparsemax* que es una arquitectura plana como la GAN base, pero donde reemplazamos la función *Softmax* en la capa de salida por una función *Sparsemax*.

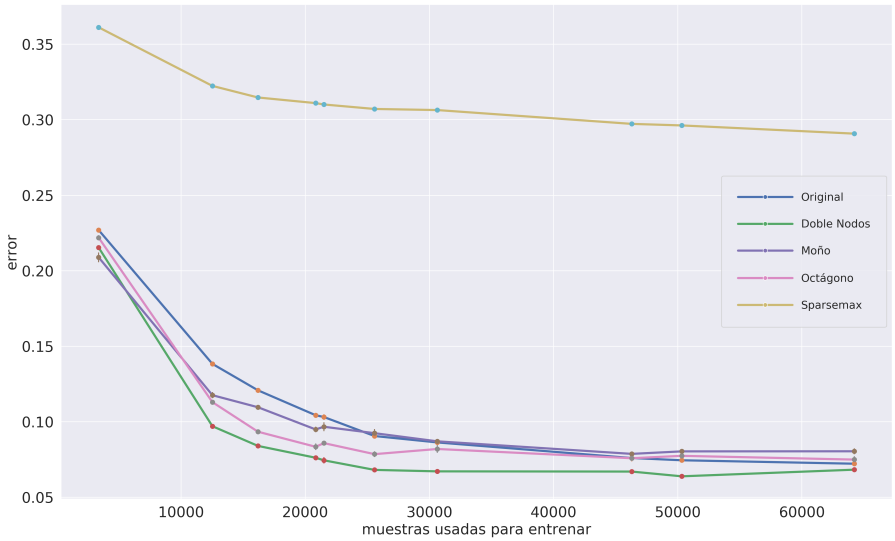

Figura 5.2: Rendimiento de las diferentes arquitecturas de GAN. Se entrenó las arquitecturas descritas en la sección 4.2.6 a 1000 épocas. A excepción de la arquitectura plana que utiliza activaciones Sparsemax, las arquitecturas de GAN muestran un rendimiento comparable, con una diferencia de error composicional agregado no mayor a $\frac{25}{1000}$. El perceptrón multicapa plano de 1024 nodos mostró el mejor rendimiento en especial cuando se tienen menos de 35000 muestras.

5.1.3. Efecto del número de capas

Dos de los parámetros esenciales de un perceptrón multicapa son el número de nodos y el número de capas. En la sección 5.1.2 encontramos que la arquitectura que tuvo mayor rendimiento fue el perceptrón multicapa plano con mayor cantidad de nodos. Entonces, evaluamos el rendimiento de la GAN al incrementar el número de capas utilizando la misma metodología que usamos en la sección 5.1.2. A continuación, enlistamos las variantes de perceptrón multicapa que evaluamos.

- **Normal (plana)** : Perceptrón multicapa con 4 capas y 512 nodos por capa oculta.
- **Faster (plana)** : Perceptrón multicapa con 4 capas y 512 nodos por capa oculta. El learning rate es ligeramente mayor.
- **5L (plana)** : Perceptrón multicapa con 5 capas y 512 nodos por capa oculta.
- **6L (plana)** : Perceptrón multicapa con 6 capas y 512 nodos por capa oculta.
- **2N Faster(plana)** : Perceptrón multicapa con 4 capas y 1024 nodos por capa oculta.

Los resultados de variar el número de capas de la GAN se resumen en la figura 5.3. Incrementar el número de nodos tiene un rendimiento superior que incrementar el número de capas. Además, el tiempo de entrenamiento requerido es menor con menor número de capas.

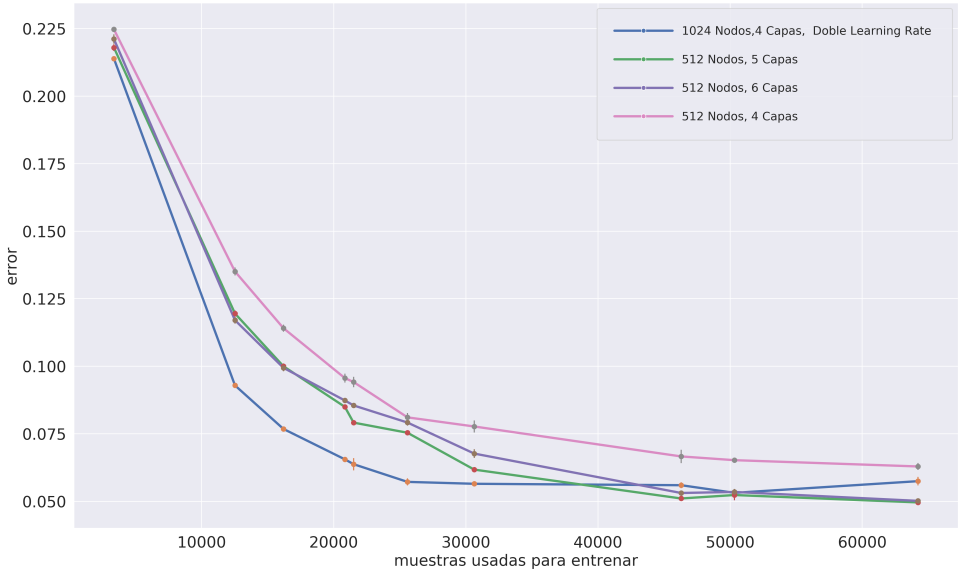

Figura 5.3: Rendimiento de las variantes de la GAN con diferente número de capas y nodos. Error composicional agregado para las GAN con diferentes cantidades de capas y nodos. Incrementar el número de capas aumenta el tiempo de entrenamiento de forma significativa mientras que el la diferencia del error composicional agregado es pequeña, no mayor a $\frac{25}{1000}$.

5.1.4. Efecto del learning rate

Utilizamos un *learning rate* que variaba de acuerdo a una de onda triangular como el de la figura 5.4.

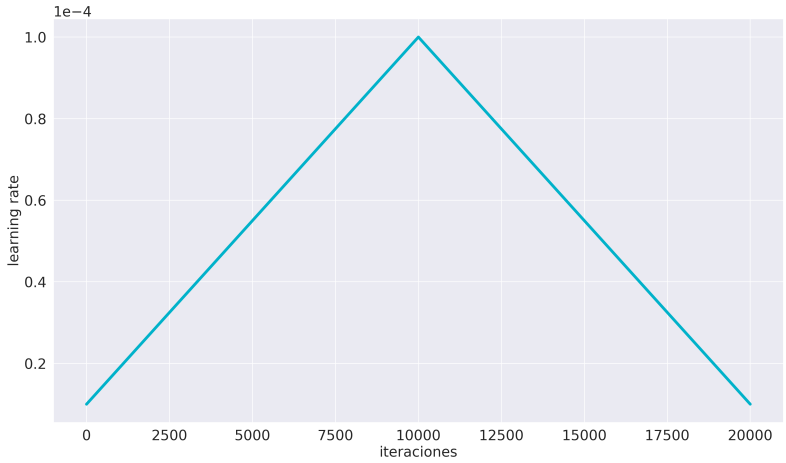

Figura 5.4: Programa triangular. Ejemplo de un programa de *learning rate* triangular. Este ejemplo considera un conjunto de datos de 10240 muestras, con un *minibatch* de 512 muestras, y un entrenamiento de 1000 epochs, lo cual equivale a 20000 iteraciones de entrenamiento.

Sin embargo, utilizar un *learning rate* variable no funcionó para conjunto de datos de combinaciones de varias distribuciones de Dirichlet. La figura 5.5 muestra el error composicional agregado utilizando un *learning rate* bajo (10^{-5}) y uno alto (10^{-4}). Al incrementar la complejidad del conjunto de datos se requiere utilizar un *learning rate* bajo, de lo contrario se tiene un aprendizaje errático.

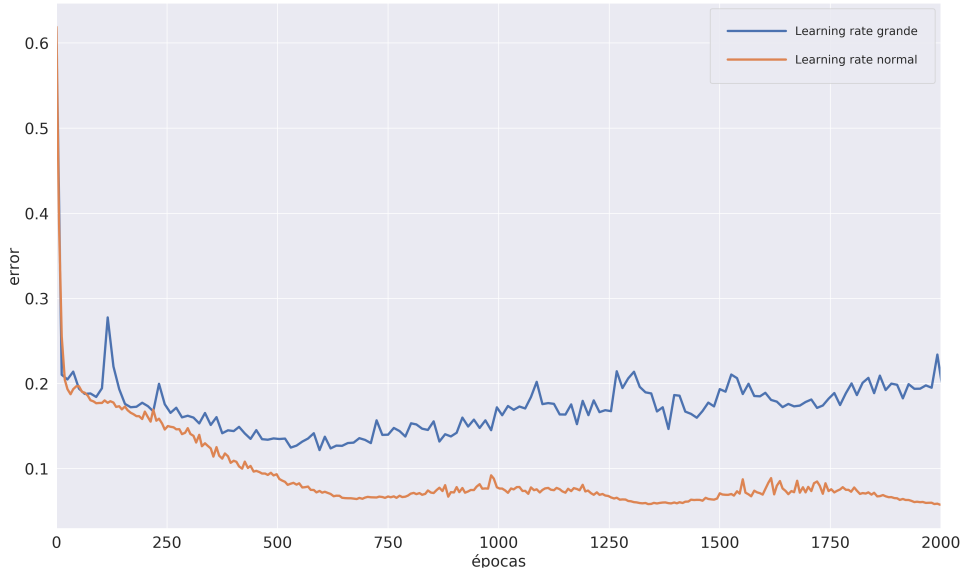

Figura 5.5: Error composicional de la GAN utilizando un *learning rate* bajo (10^{-5}) y uno alto (10^{-4}). Al incrementar la complejidad del conjunto de datos se requiere utilizar un *learning rate* bajo, de lo contrario se tiene un aprendizaje errático. Las GANs fueron entrenadas en un conjunto de datos de tres distribuciones de Dirichlet combinadas. El conjunto de datos contiene 49478 muestras de 100 especies.

5.1.5. Efecto de la complejidad de los datos

En la sección 4.2.5 se presentó un método para crear conjunto de datos de diversa complejidad combinando distribuciones de Dirichlet. Los errores composicional agregado para la GAN entrenada con los conjuntos de datos de distinta complejidad se muestra en la figura 5.6.

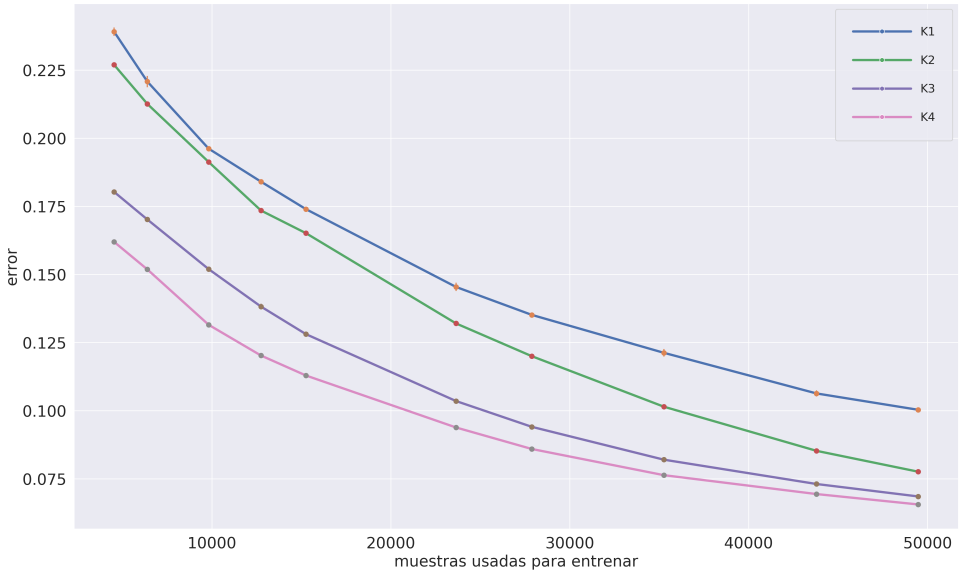

Figura 5.6: Error agregado composicional para la GAN entrenada con conjunto de datos con diferentes combinaciones de datos de distribuciones de Dirichlet de 100 especies. Mientras mayor sea la complejidad del conjunto de datos, mayor sea la cantidad de distribuciones Dirichlet combinadas, más eficiente es el aprendizaje de la GAN.

Contrario a la intuición, la GAN tiene un mejor aprendizaje con los conjuntos de datos de mayor complejidad. Para confirmar esta observación, calculamos entropía de Shannon promediada por el número de especies para cada conjunto de datos.

conjunto de datos	Entropía de Shannon promediada por especie.
K1	2.2095 nats
K2	3.051 nats
K3	3.3656 nats
K4	3.4956 nats

Tabla 5.1: Shannon promediada por el número de especies.

5.1.6. Efecto del número de muestras de entrenamiento para un aprendizaje correcto

Las gráficas de las secciones anteriores sugieren que la relación entre el error composicional agregado y el número de muestras es no lineal. Para encontrar la función que describe la curva *error-número de muestras* realizamos un ajuste de curva. Utilizamos regresión de mínimos cuadrados no lineales, en particular, el algoritmo iterativo del Algoritmo de Levenberg-Marquardt. [1].

Entonces, utilizamos el algoritmo previamente mencionado para encontrar los parámetros del modelo de función no lineal de la ecuación 5.1.

$$\text{error} = \frac{AK_n}{\sqrt{N}} + BK_n \tag{5.1}$$

Donde:

- A y B son constantes a determinar.
- K_n es el número de distribuciones de Dirichlet combinadas. Por ejemplo, el conjunto de datos K2 tendría un $K_n = 2$.
- N es el número de muestras usadas en el entrenamiento del modelo generativo.

En la tabla 5.2 y figura se resumen los resultados del ajuste de curva para el modelo exponencial de la ecuación 5.1.

Como se observa en la figura 5.7 A), B), C), D) el modelo propuesto en la ecuación 5.1 se ajusta correctamente a las curvas experimentales. Para evaluar el ajuste de las curvas, utilizamos el error medio cuadrático. Los resultados se muestran en la tabla 5.3. En general los errores medio cuadráticos fueron bajos, menores a una milésima.

conjunto de datos	A	B
K1	13.5437	0.0519
K2	7.3759	0.0142
K3	3.7956	0.0087
K4	2.4278	0.007

Tabla 5.2: Parámetros del ajuste de curva para el modelo exponencial de la curva de aprendizaje de la GAN

conjunto de datos	Error cuadrático medio entre el modelo exponencial y la curva experimental.
K1	8.59e-05
K2	1.71e-04
K3	7.34e-05
K4	3.01e-05

Tabla 5.3: Error medio cuadrático para cada conjunto de datos

[1] Este algoritmo se encuentra en la librería Scipy de Python

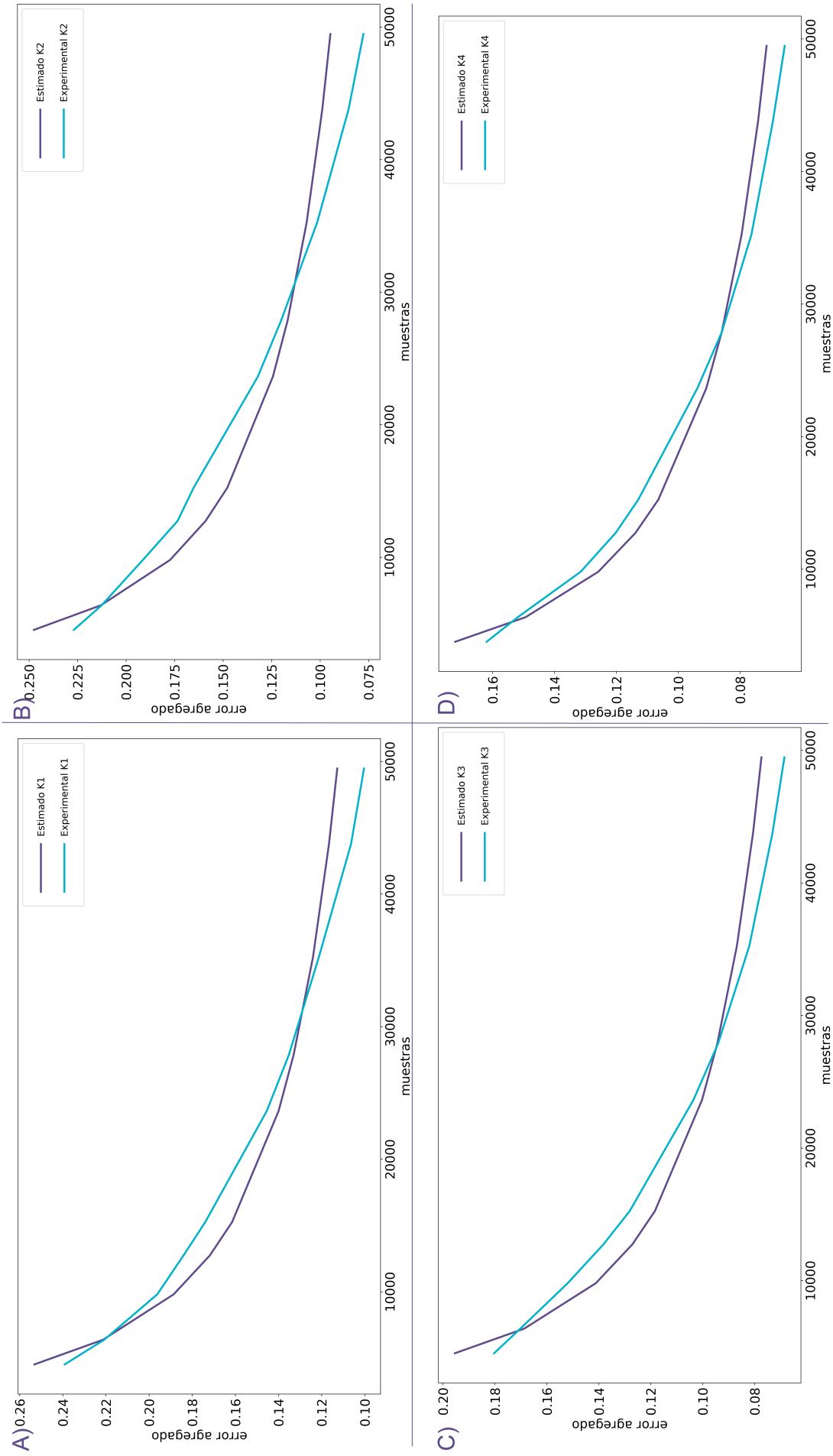

Figura 5.7: Comparación de curva de error agregado experimental contra estimadas A),B),C),D) en todas las curvas el modelo subestima el error dentro del intervalo de 10000 a 3000 muestras. Mientras que para más de 30000 muestras el modelo sobrestima el error.

A partir de la tabla 5.2 podemos obtener una expresión para los parámetros A y B en función de las n distribuciones de Dirichlet combinadas.

$$A(n) = -3.6928 \cdot n + 16.0178 \tag{5.2}$$

$$B(n) = -0.014 \cdot n + 0.0555 \tag{5.3}$$

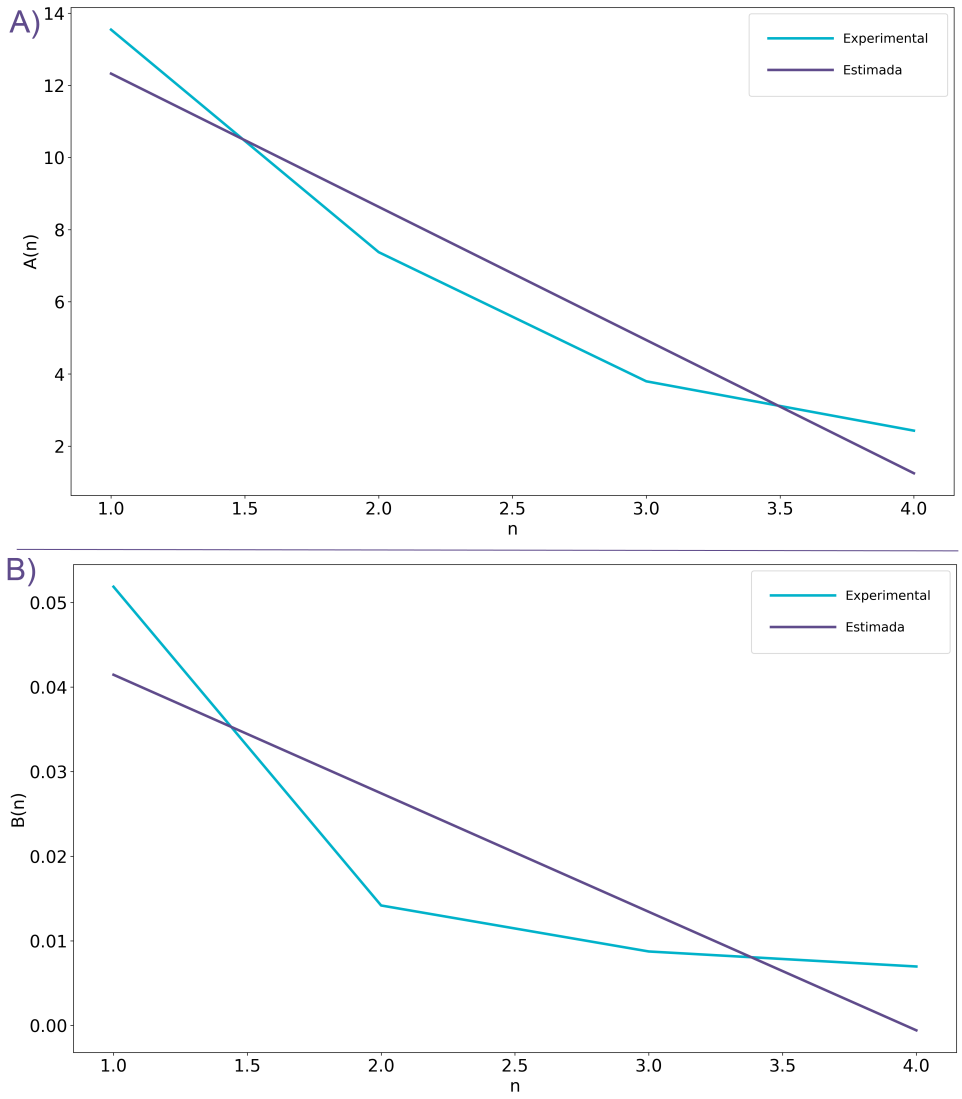

Figura 5.8: Datos de los parámetros del modelo en función de conjunto de datos usado. **A)** La recta estimada por regresión lineal para el parámetro A tiene un coeficiente de correlación $r = -0.9602$.**B)** La recta estimada por regresión lineal para el parámetro B tiene un coeficiente de correlación $r = -0.8544$

5.2. Resultados en datos experimentales

De forma general, el proceso para obtener los datos experimentales consiste en los siguientes pasos [28]:

1. Extraer el ADN de las muestras de suelo y purificarlas.
2. Amplificar [2] el ADN por *reacción en cadena de la polimerasa* (PCR por sus siglas en inglés).
3. Secuenciación del ARN ribosomal 16S.
4. Análisis bioinformático de las secuencias.

Durante el análisis de las secuencias, se agrupan grupos de secuencias. El cómo se agrupan depende de la situación específica. A estos grupos de secuencias se les llama unidad taxonómica operativa (OTU por sus siglas en inglés). El número de OTUs funciona como el número de componentes del conjunto de datos.

La taxonomía de suelos de USDA se compone de seis niveles: orden, suborden, gran grupo, subgrupo, familia, y serie. En la tabla 5.4 damos una descripción de los conjuntos de datos experimentales usados. Cada uno de esto corresponde a un nivel taxonómico, de tal forma que varía el número de unidades taxonómicas operativas.

conjunto de datos	Dimensiones
L2	1132 x 36
L3	1132 x 112
L3	1132 x 233
L5	1132 x 365
L6	1132 x 654

Tabla 5.4: Tabla de conjunto de datoss experimentales

Uno de los problemas de tener pocas muestras en los conjuntos de datos experimentales es que no encontramos una forma de dividir el conjunto de datos de tal forma que las distribuciones de abundancia por especie sean iguales (o por lo menos muy cercanas). Es por ello por lo que decidimos utilizar todas las 1132 muestras de cada conjunto de datos en el entrenamiento de la GAN.

La distribución de Dirichlet comúnmente es utilizada como modelo para datos composicionales [20]. Entonces, la utilizamos para comparar las muestras generadas por la GAN. Para modelar los datos experimentales con una distribución de Dirichlet, debemos especificar un vector de parámetros α. Una forma para encontrar dicho vector es por *estimación por máxima verosimilitud*, en particular utilizamos el algoritmo propuesto por Minka [19][3]. Por dificultades para visualizar los vectores α estimados debido a que son multidimensionales, puede econtrar los CSV con los valores de α aqui.

[2] Crear copias de una secuencia de ADN

[3] Una implementación del algoritmo se puede encontrar en R como *dirichlet.mle* dentro del paquete *sirt*

5.2.1. L2

Este conjunto de datos contiene 1132 muestras y 36 OTU/especies. Como primer análisis exploratorio graficamos la abundancia de cada especie promediada por el número de muestras, figura 5.9.

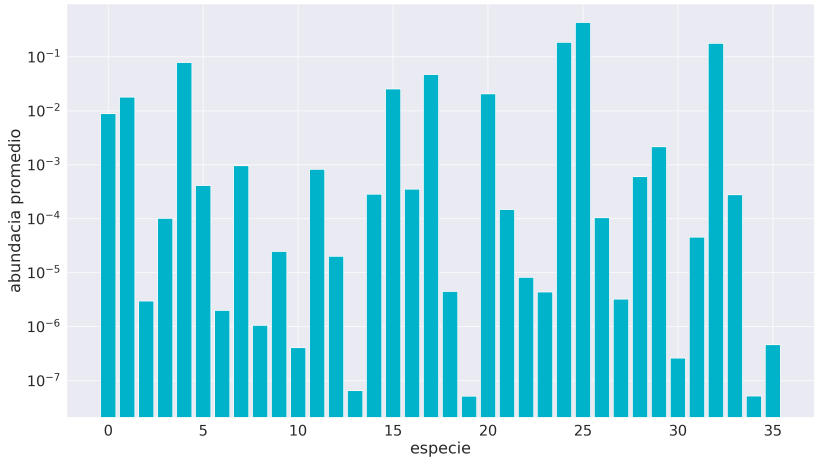

Figura 5.9: Abundancia de las especies promediada por el número de especies. Una minoría de especies tiene mayor abundancia que las demás. Esta minoría la calculamos con el número de especies cuya abundancia promedio es mayor a $\frac{1}{36}$, la abundancia que tendría cada especie si la abundancia se distribuye de forma uniforme. Para este conjunto de datos 5 especies o el 13.89 % de especies dominan a las demás: (17, 0.05), (4, 0.08), (32, 0.18), (24, 0.19), y (25, 0.43).

Una pregunta que surge al observar la figura 5.9, si ¿es posible generar correctamente especies con abundancia promedio pequeña, menor a $\frac{1}{N}$?, donde N es el número de especies. Entonces, analizamos más a fondo los elementos del error composicional agregado, es decir la distancia Jensen Shannon entre las distribuciones de abundancia de cada especie. Debido a que los histogramas se vuelven enmarañados conforme incrementa el número de especies/OTU, es conveniente utilizar visualizaciones como los diagramas de caja y de enjambre de abeja.

Encontramos que la GAN y el modelo de Dirichlet necesitan alrededor de 400 muestras no nulas para aprender a generar una especie del conjunto de datos L2 correctamente. Este hecho se observa en los histogramas de abundancias para las especies 3 y 14, figura A) y B), con 342 y 472 muestras no nulas respectivamente. Cuando se tienen pocas muestras no nulas, ambos modelos generan muestras deficientes.

Entonces, limitamos el cálculo del error composicional agregado a especies con más de 400 muestras no nulas. En la tabla 5.5 se muestran los subconjuntos de especies.

$\mathbb{L}2_S$ **: Especies con suficientes muestras no nulas**	{0, 1, 4, 5, 7, 11, 14, 15, 16, 17, 20, 24, 25, 28, 29, 32, 33}
$\mathbb{L}2_I$ **: Especies con insuficientes muestras no nulas**	{2, 3, 6, 8, 9, 10, 12, 13, 18, 19, 21, 22, 23, 26, 27, 30, 31, 34, 35}

Tabla 5.5: Especies con suficientes e insuficientes muestras no nulas. Con las especies de $\mathbb{L}2_S$ podemos calcular sus distancias Jensen-Shannon.

Cuando se tienen menos de 400 especies no nulas, el modelo de Dirichlet y la GAN no logran generar muestras diversas como en el conjunto de datos real.

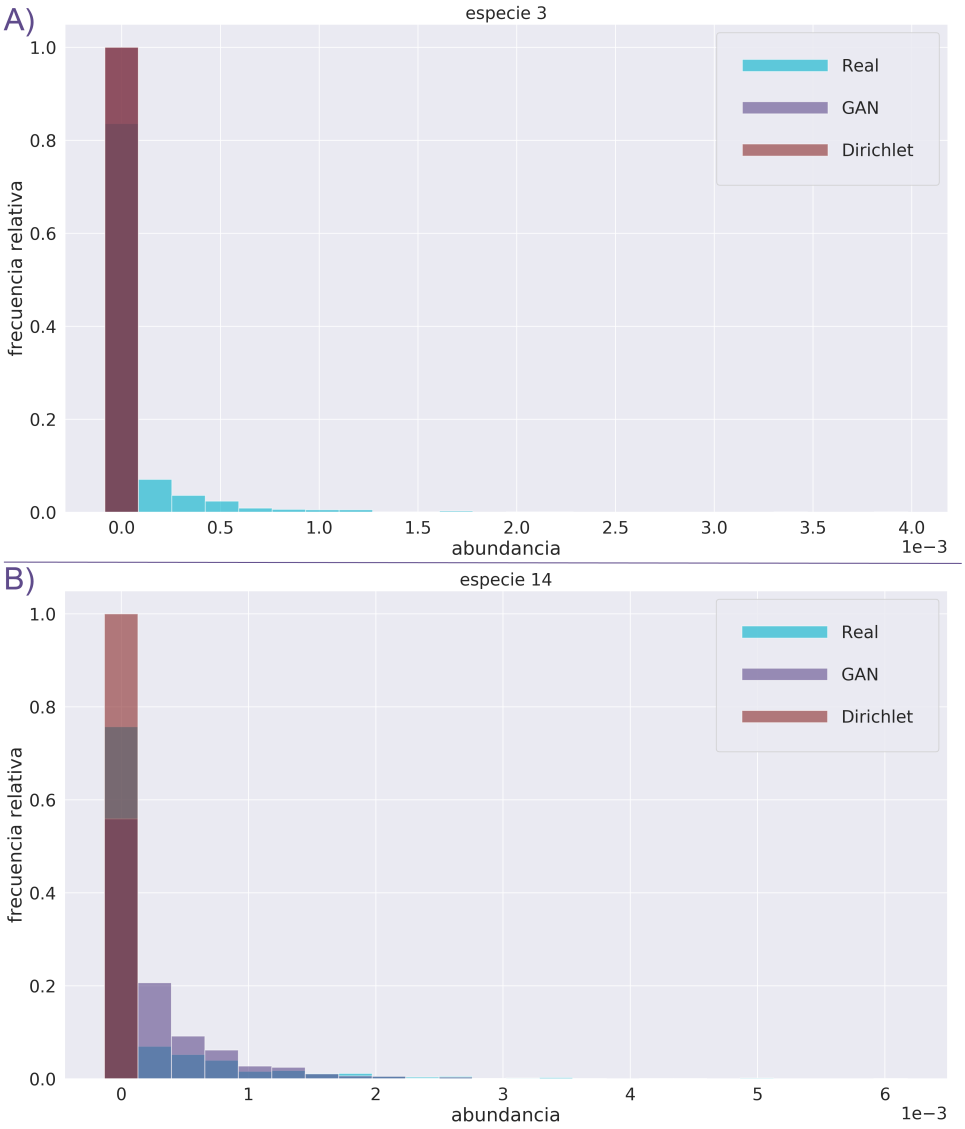

Figura 5.10: Histograma de abundancias para la especie 3 y 14.A), Ambos modelos son incapaces de generar abundancia de especie 3. Esta especie tiene 342 muestras no nulas. **B)** Ambos modelos aprenden a generar abundancia de especie 14. Esta especie tiene 472 muestras no nulas.

5.2.1.1. Especies con suficientes muestras no nulas

A partir del subconjunto de especies donde tenemos histogramas útiles para calcular el error composicional agregado obtenemos la figura 5.11 A). Debido a que los gráficos de barras no escalan eficientemente con el número de especies/OTU, es mejor utilizar diagramas de caja como el de la figura 5.11 B) para conjunto de datoss con más especies/OTU.

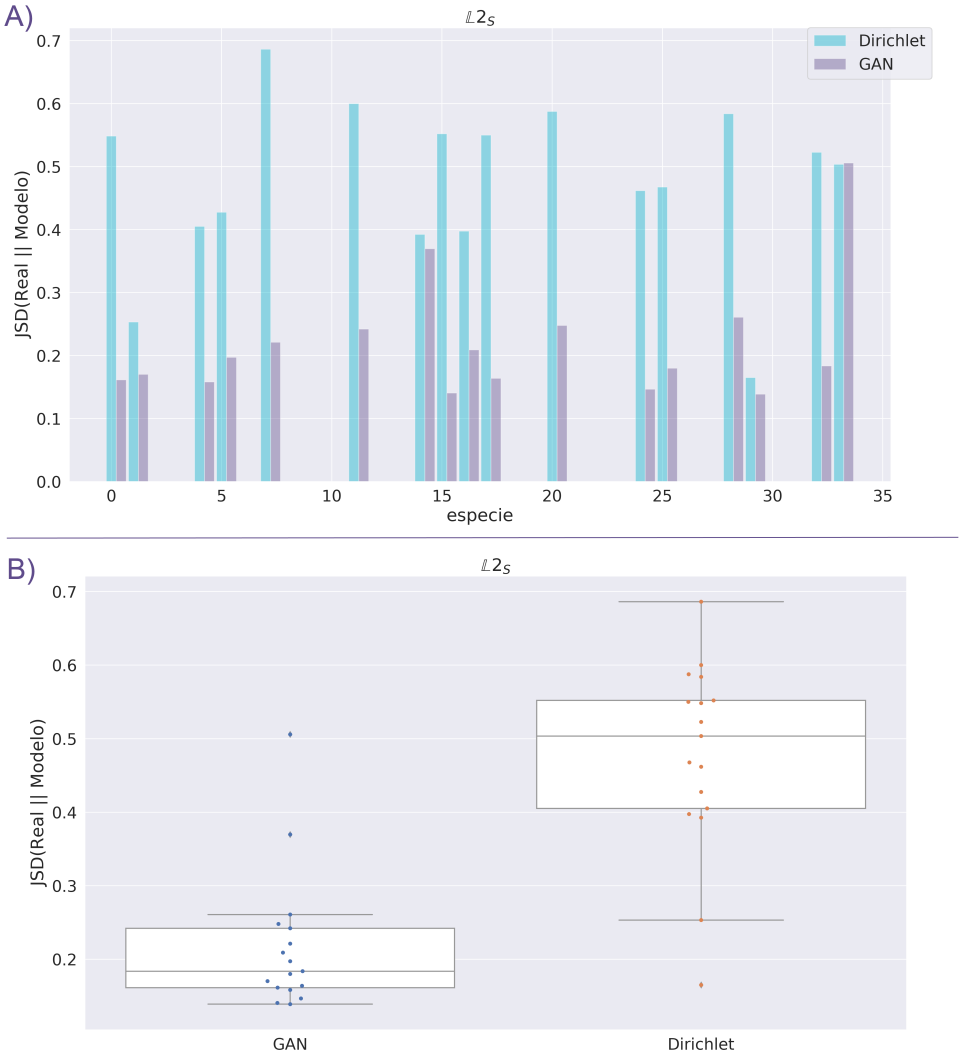

Figura 5.11: Gráfico de distancia JSD para cada especie del subconjunto $\mathbb{L}2_S$. **A)**, La GAN genera mejores muestras de abundancias para la gran mayoría de especies. **B)** El modelo de Dirichlet genera especies con distancias JSD mayores que de la GAN. Además, las distancias tienen mayor varianza.

Una duda que surge es: ¿cuál es la abundancia promedio importante para que los modelos aprendan a generar muestras correctamente? En la figura 5.12 se muestra un gráfico de dispersión de abundancia promedio y distancia Jensen-Shannon. Para este gráfico utilizamos únicamente la especie del subconjunto $\mathbb{L}2_S$.

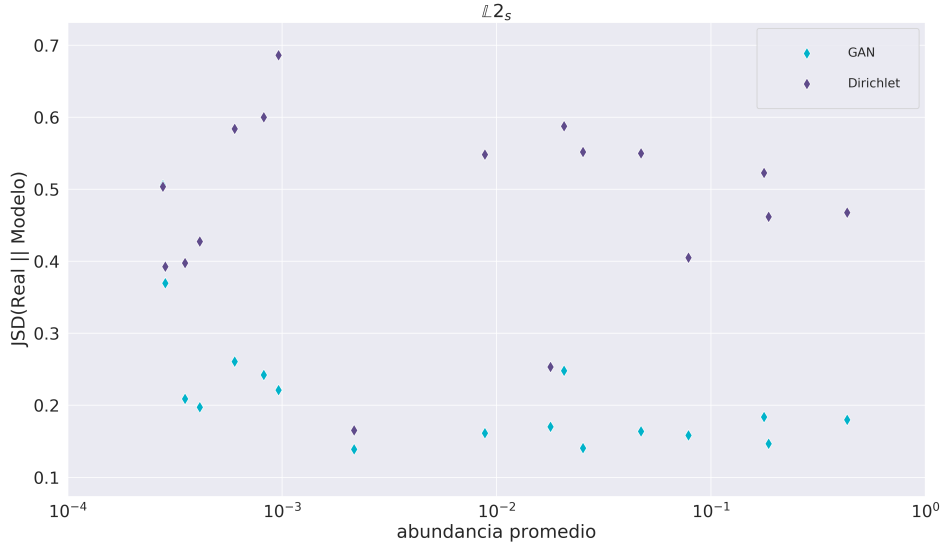

Figura 5.12: Diagrama dispersión abundancia promedio y distancia Jensen-Shannon. Incluso con abundancias promedio pequeñas se puede obtener una distancia Jensen-Shannon agregado baja.

La figura 5.12 sugiere que la GAN aprenden aproximadamente igual las especies con abundancias promedio mayores a 10^{-3}, incrementando su error para especie con menor abundancia promedio. La figura 5.13 muestra una especie con mayor cantidad de muestras no nulas es una especie con mayor abundancia promedio.

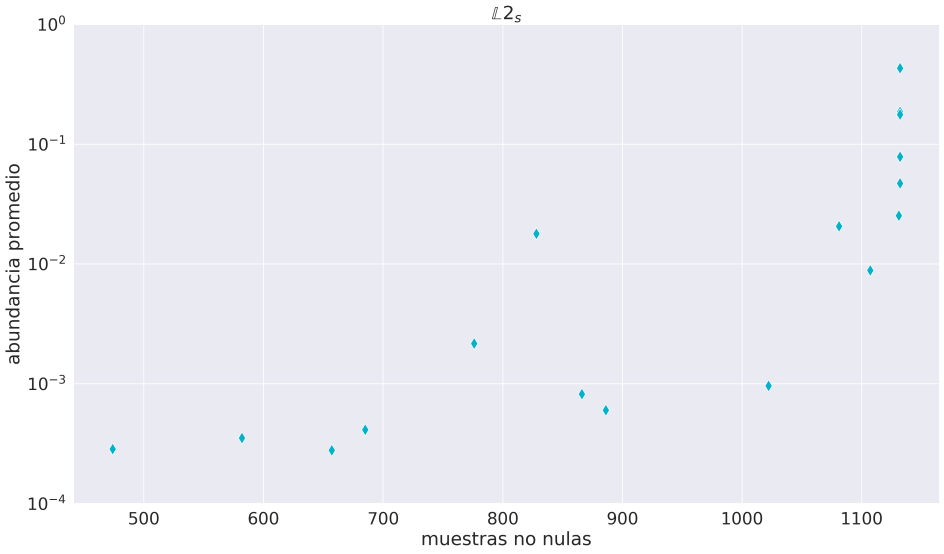

Figura 5.13: Diagrama dispersión muestras no nulas y abundancia promedio. La cantidad de muestras no nulas de una especie está directamente relacionada con su abundancia promedio.

Como tenemos una cantidad relativamente pequeña de muestras (1132), podemos utilizar diagramas de enjambre por especie para visualizar las abundancias de las muestras reales y generadas. Dos casos interesantes

son el de la especie 15 y 33, figura . La GAN genera muestras de abundancia de especie 15 de forma superior, con una distancia Jensen-Shannon mucho menor que la del modelo de Dirichlet. Por otro lado, para la especie 33, ambos modelos generan muestras de abundancia muy parecidas.

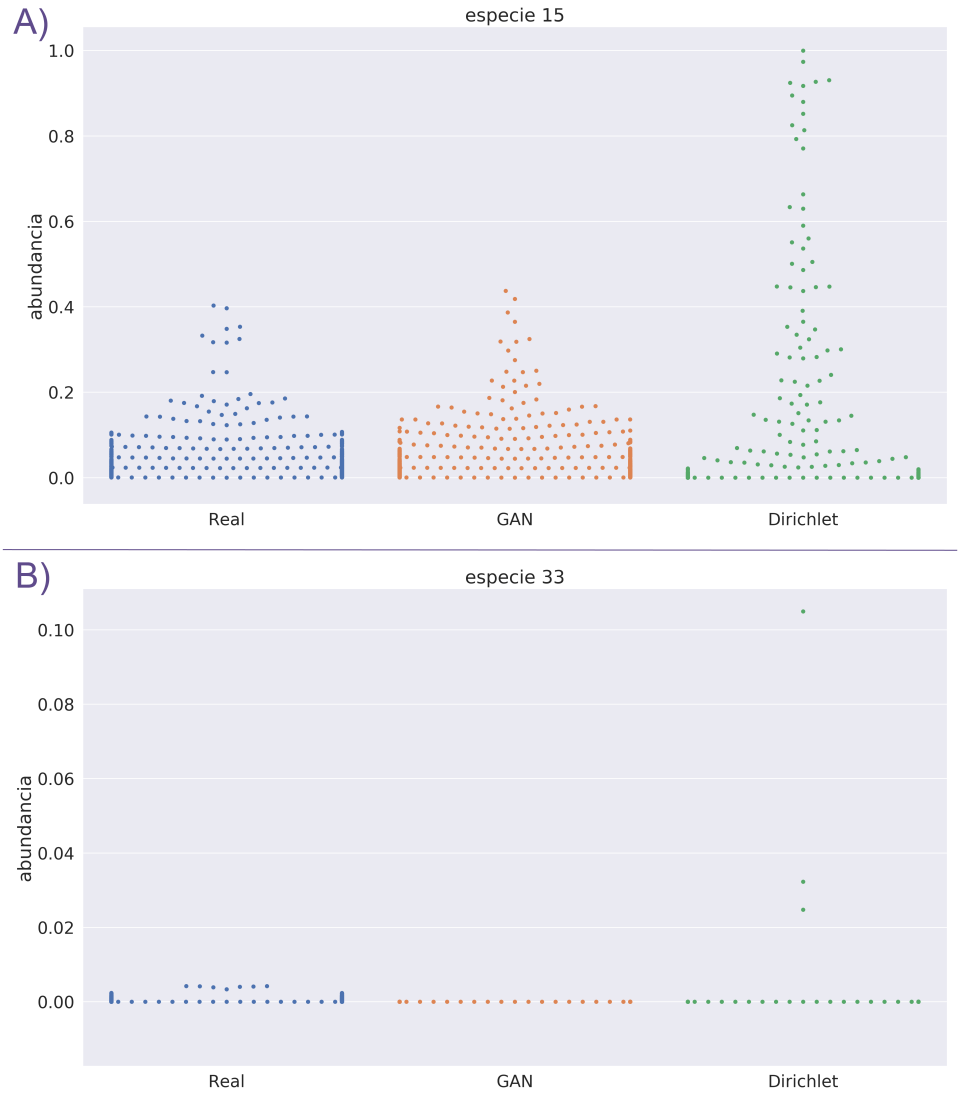

Figura 5.14: Gráfico de enjambre para las abundancias de la especie 15 y 33. **A)** El modelo de la GAN produce mejores muestras que el modelo de Dirichlet. **B)** como sugiere la figura 5.11 B), la diferencia de error entre estas especies no es mucha. El modelo de Dirichlet supera ligeramente a la GAN con una distancia Jensen-Shannon de 0.5035 contra 0.5057.

5.2.1.2. Especies con pocas muestras no nulas

Para las especies del subconjunto $\mathbb{L}2_I$, de las cuales es difícil obtener y comparar sus histogramas, examinamos las medias de las abundancias. En específico, las comparamos de acuerdo a la ecuación 5.4, la cual llamaremos distancia logarítmica.

$$d_{log} = -log_{10}|\hat{R}_n - \hat{M}_n| \tag{5.4}$$

Donde:

- $\hat{R}_k$ Es el promedio de abundancia de las muestras reales para la enésima especie.
- $\hat{M}_k$ Es el promedio de abundancia de las muestras generadas por el modelo para la enésima especie.

Como las abundancias de las muestras son números no mayores a la unidad, siempre positivos, y pequeños, en su mayoría menores a $\frac{1}{36}, N = 36$, se utilizó un logaritmo en la ecuación 5.4 para facilitar la comparación de valores. **Una observación muy importante es que por el logaritmo de la ecuación 5.4, mientras mayor sea el valor de d_{log}, más semejantes son las muestras generadas a las reales.**

Con la distancia logarítmica también graficamos un diagrama de cajas para especie, figura 5.15.

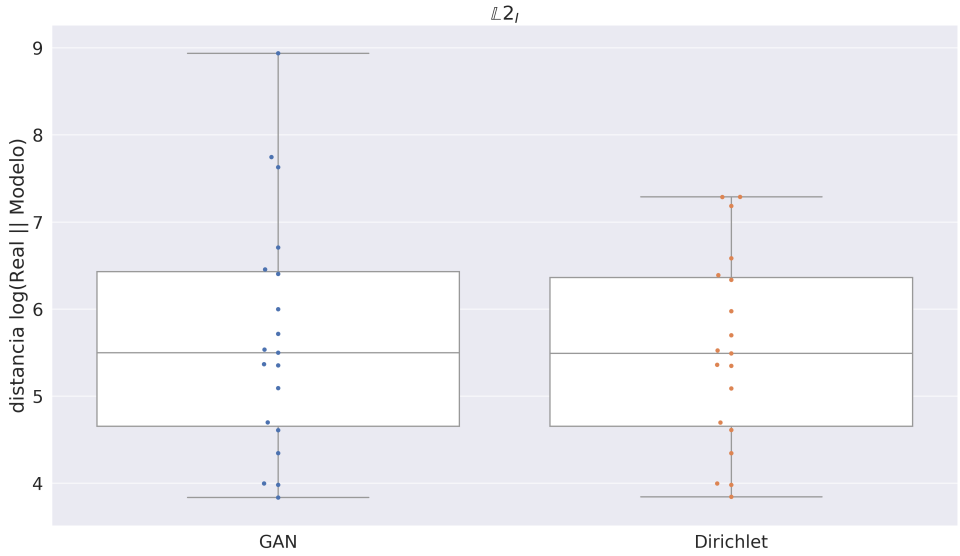

Figura 5.15: Diagrama de caja para la distancia logarítmica para cada especie del subconjunto $\mathbb{L}2_I$. El modelo de Dirichlet genera especies con distancias logarítmicas menores que de la GAN. A Pesar de ello, ambos modelos tienen una media semejante.

A pesar de la sencillez de la ecuación 5.4, ésta nos permite comparar las muestras generadas por ambos modelos. Para ubicar de forma eficiente cuál de los modelos aprende mejor una especie/OTU, calculamos las razones entre las distancias logarítmicas o error composicional agregado de ambos modelos para cada especie como se indica en la ecuación 5.5.

$$Qe_{\mathrm{n}} = \frac{d(Real_n||Dirichlet_n)}{d(Real_n||GAN_n)} \tag{5.5}$$

Donde:

- $d(Real_n||Modelo_n)$: Es la distancia logarítmica o error composicional agregado para la enésima especie.

Al usar la ecuación 5.5 con el subconjunto $\mathbb{L}2_I$, encontramos que en las especies 13 y 21 ambos modelos generan muestras de abundancia con distancia logarítmica semejante. Los diagramas de caja de abundancia para estas especies lo confirman, figuras 5.16 A), B).

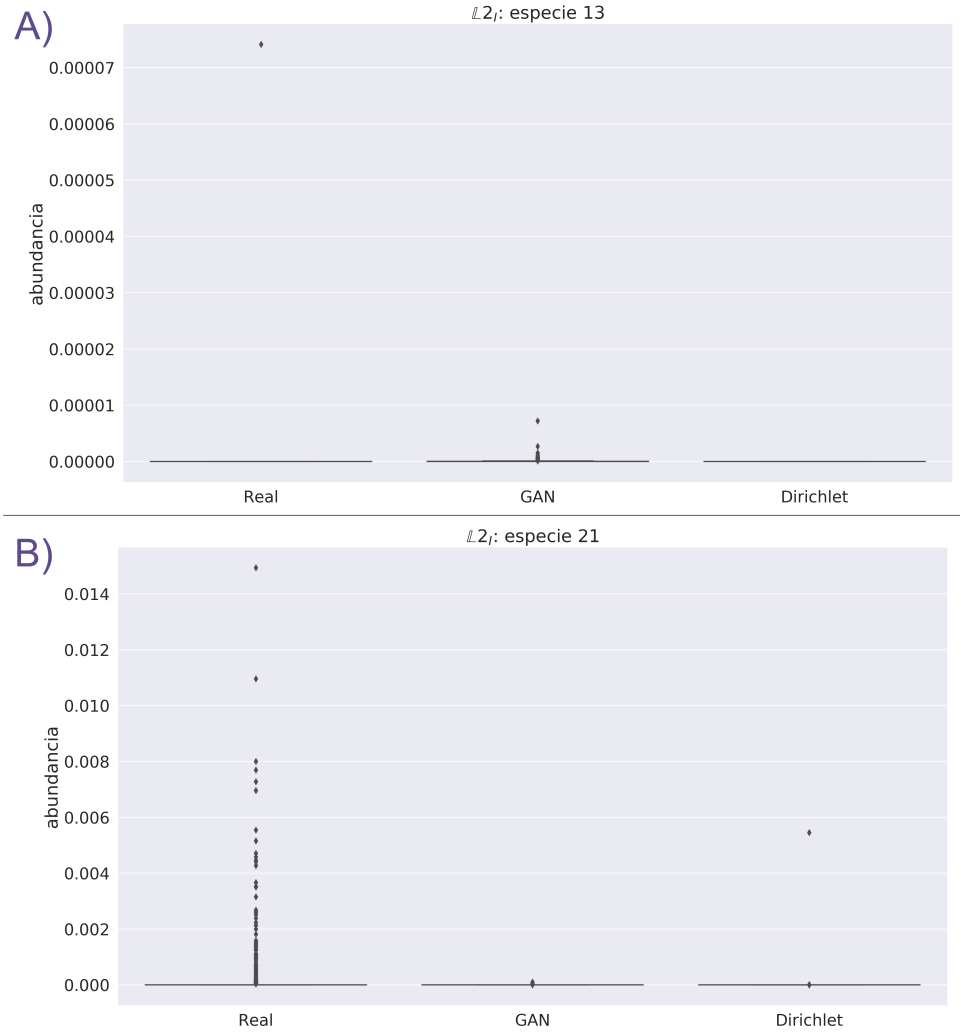

Figura 5.16: Diagrama de caja de abundancia para la especie 13 y 21. **A)** El modelo de Dirichlet tiene una distancia logarítmica menor a la GAN. Sin embargo, la diferencia entre ambos modelos es pequeña, con distancias logarítmicas de 8.93854 para la GAN, y 7.18403 para el modelo de Dirichlet. Ambos modelos producen abundancias de forma deficiente. **B)** De igual forma, el modelo de Dirichlet produce un poco mejor abundancia para la especie 21, pero de forma insignificante. Las distancias logarítmicas para esta especie son de 3.83365 para la GAN y 3.84207 para el modelo de Dirichlet.

A pesar de que el diagrama de cajas de la figura 5.15 A) sugiere que la GAN genera mejores muestras, *observamos que en cuando se tienen pocas muestras no nulas, ambos modelos hacen un trabajo deficiente en generar muestras de abundancia de especie.*

5.2.2. L3

Este conjunto de datos tiene 1132 muestras y 112 especies/OTU. Una diferencia importante es que al incrementar el número de especies/OTU, la abundancia promedio de las especies también se reduce.

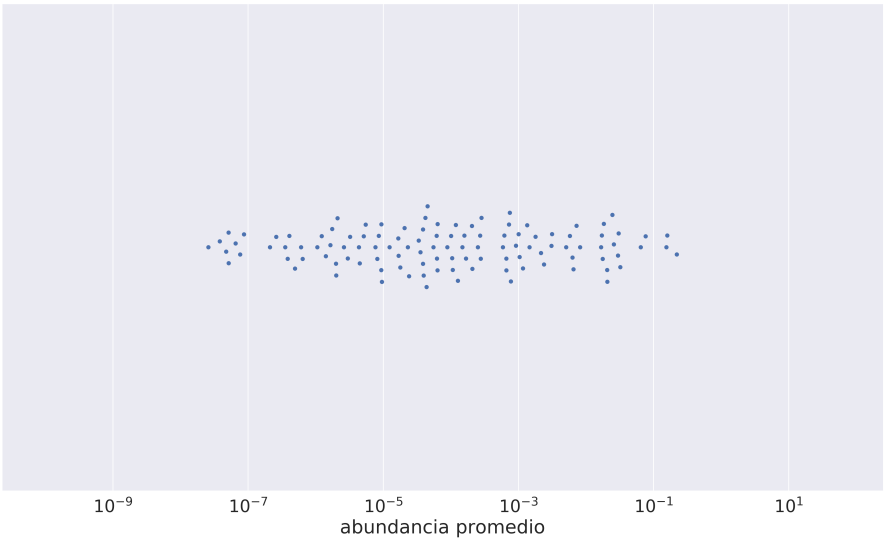

Figura 5.17: Gráfico de enjambre para las abundancias promedio. Conforme se incrementa el número de especies/OTU, se reduce la abundancia promedio de las especies.

Para este conjunto de datos, el modelo de Dirichlet requiere de menos muestras no nulas para generar muestras de abundancia de especie de forma razonable. Por ejemplo, la especie 31 tiene 20 muestras no nulas, además el modelo de Dirichlet supera notablemente a la GAN para ésta especie, figura 5.18.

A partir de esta observación dividimos las especies en dos subconjuntos para su evaluación, $\mathbb{L}3_S$ y $\mathbb{L}3_I$, como se muestran en la tabla 5.6. Se tienen 82 elementos en el subconjunto $\mathbb{L}3_S$ y 30 en el subconjunto $\mathbb{L}3_I$, dando una razón de 2.73.

$\mathbb{L}3_S$	{0,2,3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,21,22,23,24,25,26,27,30, 33,34,37,38,40,42,43,48,52,53,54,55,56,57,59,60,61,62,63,64,65,66,70, 71,73,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,97,98, 100,101,102,103,104,105,106,107,108}
$\mathbb{L}3_I$	{1,6,17,28,29,31,32,35,36,39,41,44,45,46,47,49,50,51, 58,67,68,69,72,74,95,96,99,109,110,111}

Tabla 5.6: Especies con suficientes e insuficientes muestras no nulas

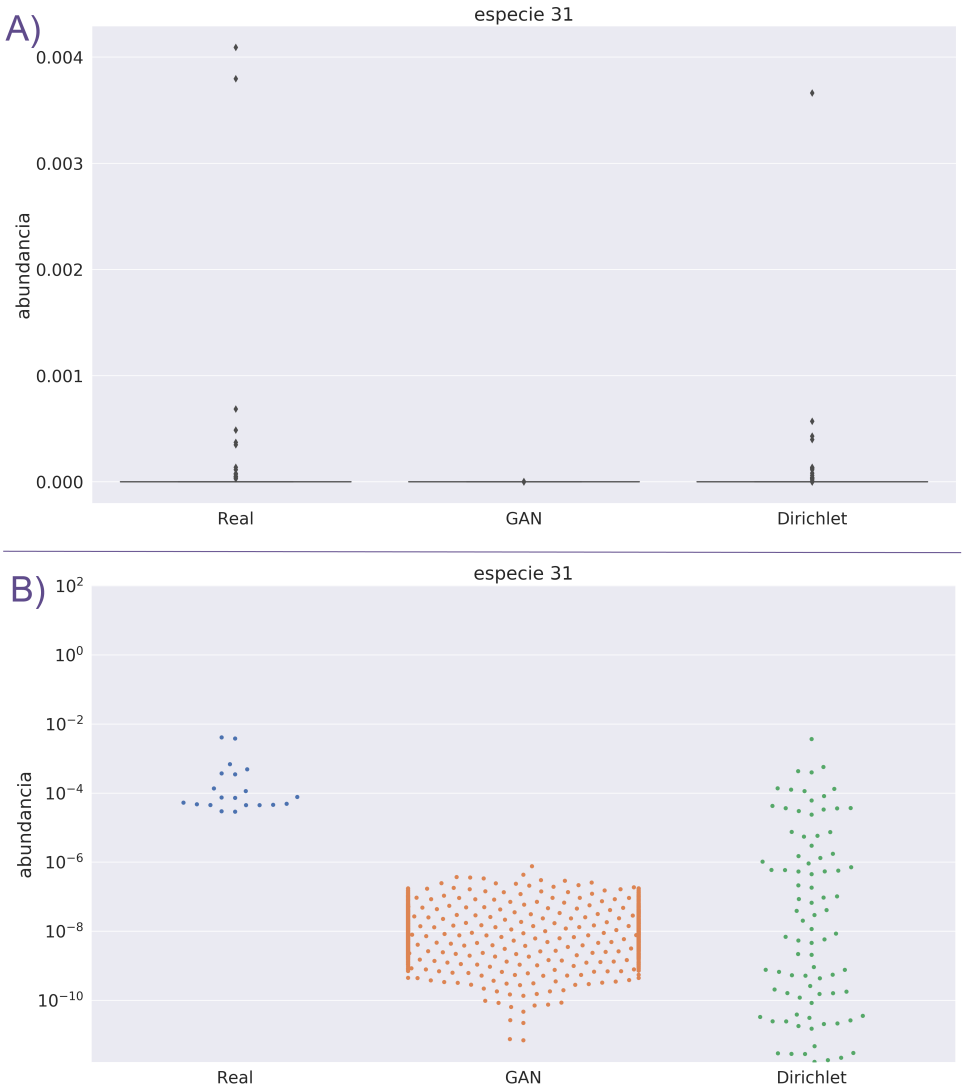

Figura 5.18: Gráfico de enjambre para las abundancias promedio de la especie 31. Conforme se incrementa el número de especies/OTU, se reduce la abundancia promedio de las especies. **A)** escala normal de la abundancia. **B)** escala logarítmica

Con pocas muestras, el modelo de Dirichlet genera abundancias de especie 31 mejor que la GAN. Aunque, tomando en cuenta las especies de $\mathbb{L}3_I$, la media de las distancias logarítmicas para ambos modelos es muy parecida, figura 5.21, por lo que no son muy diferentes las muestras.

Con el subconjunto de especies $\mathbb{L}3_S$ podemos calcular las distancias Jensen-Shannon y graficarlas contra su su abundancia promedio 5.19.

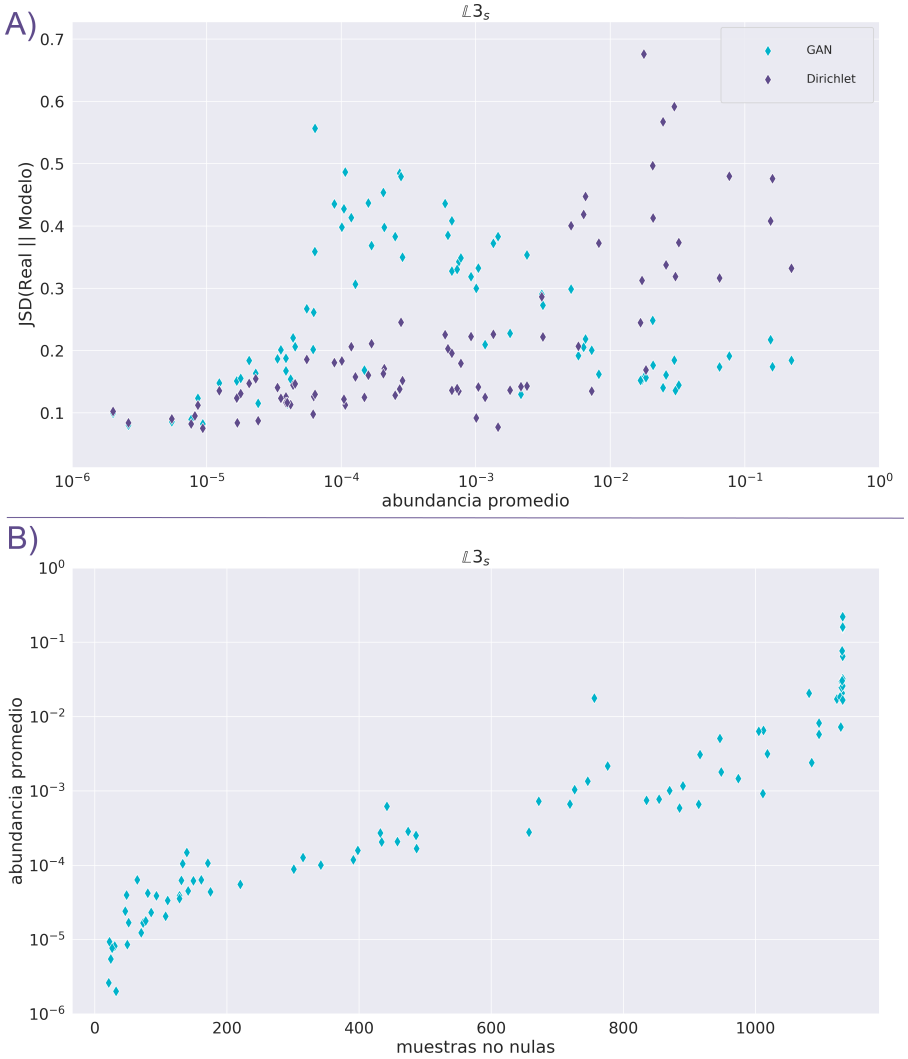

Figura 5.19: Gráfico de dispersión de distancia Jensen-Shannon y abundancia de especie. **A)** la distancia Jensen-Shannon es calculada únicamente para las especies del subconjunto $\mathbb{L}3_S$. **B)** de forma general, la abundancia promedio incrementa con el número de muestras no nulas de la especie.

La figura 5.19 A) sugiere que la GAN aprende significativamente mejor que el modelo de Dirichlet las especies con mayor abundancia promedio. La figura 5.19 B) muestra que el número de muestras no nulas de una especie está directamente relacionado su abundancia promedio.

Las especies 5 y 12 fueron en donde la GAN tuvo mayor desempeño sobre el modelo de Dirichlet. Estas especies tienen 756 y 1130 muestras no nulas respectivamente.

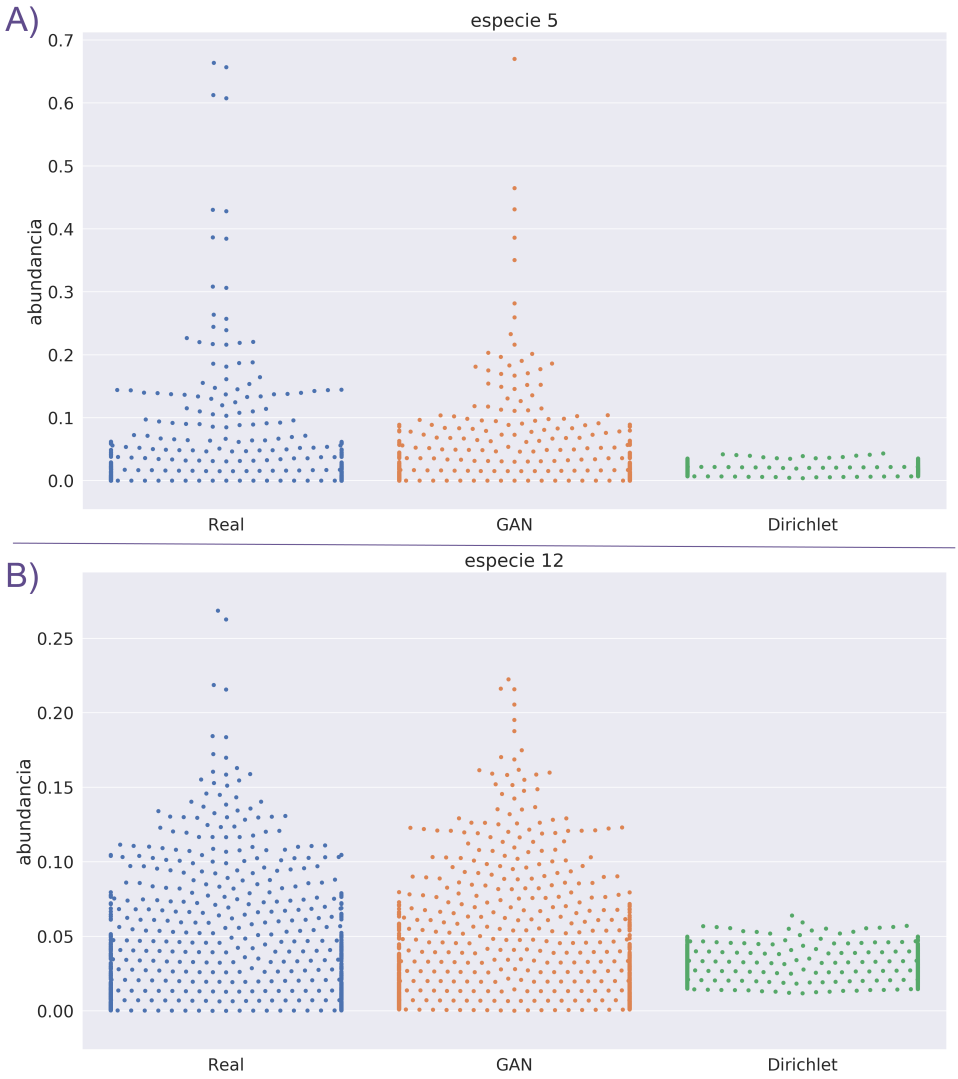

Figura 5.20: Gráficos de enjambre para la especie 5 y 12. **A)** La especie 5 tiene varias muestras con abundancia mayor a 0.05. La GAN genera abundancia mucho mejor que el modelo de Dirichlet. **B)** El modelo de Dirichlet no logra generar la mayoría de las abundancias mayores a 0.05.

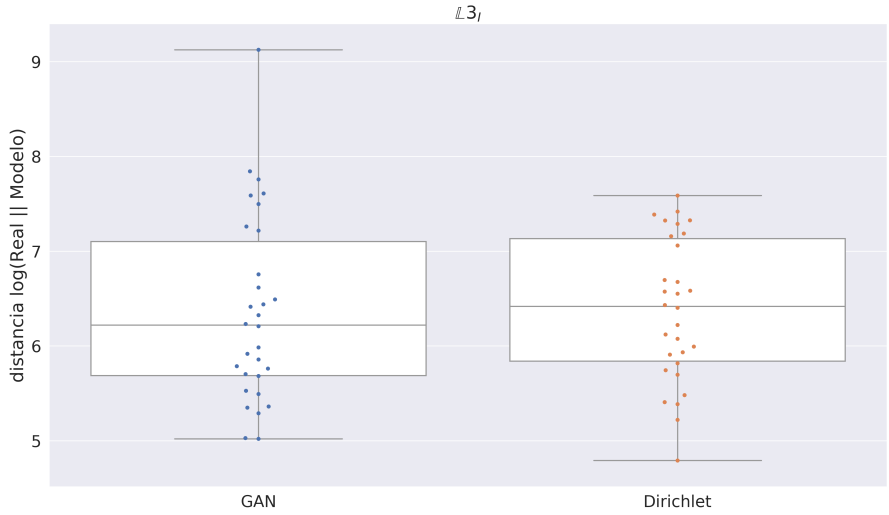

Figura 5.21: Diagrama de caja para la distancia logarítmica para cada especie del subconjunto $\mathbb{L}3_I$. El modelo de Dirichlet tiene una media mayor de distancias logarítmicas, aunque en algunas especies la GAN tiene valores notablemente más altos.

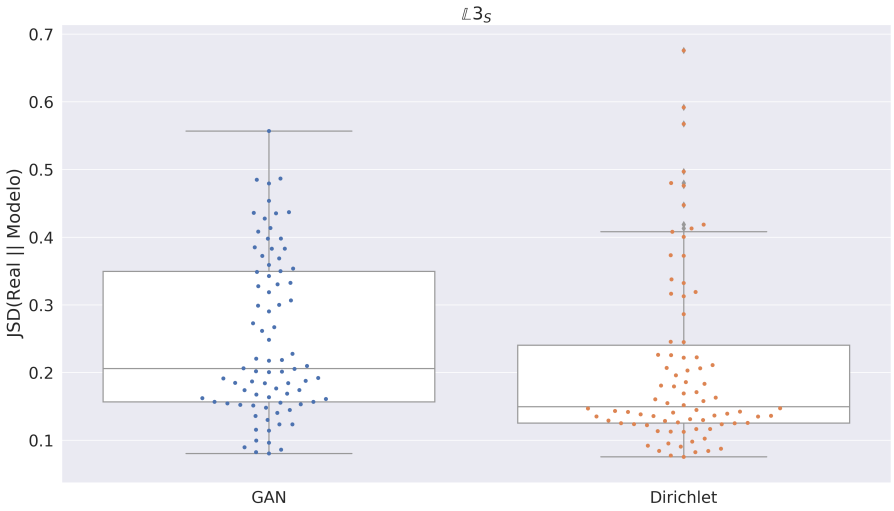

Figura 5.22: Diagrama de caja para la distancia Jensen-Shannon para cada especie del subconjunto $\mathbb{L}3_S$. Al comparar las medias, notamos que el modelo de Dirichlet es mejor que la GAN. Sin embargo, hay algunas excepciones donde el modelo de Dirichlet hace muy mal trabajo.

El modelo de Dirichlet genera mejor muestras de abundancia en especies con una abundancia promedio pequeña, figura 5.19. Como se tienen más especies, y la misma cantidad de muestras, la mayoría de especies son de baja abundancia promedio, lo cual se refleja una distancia Jensen-Shannon agregada sobresaliente para la muestras del modelo de Dirichlet, figura 5.22.

5.2.3. L4

Este conjunto de datos tiene 1132 muestras y 233 especies/OTU. Como se tienen inclusive más especies, las abundancias promedio por especie son menores. Entonces, al igual que para el conjunto de datos L3, dividimos las especies en dos subconjuntos para su evaluación, $\mathbb{L}4_S$ y $\mathbb{L}4_I$ con un umbral de 20 especies, tabla 5.7.

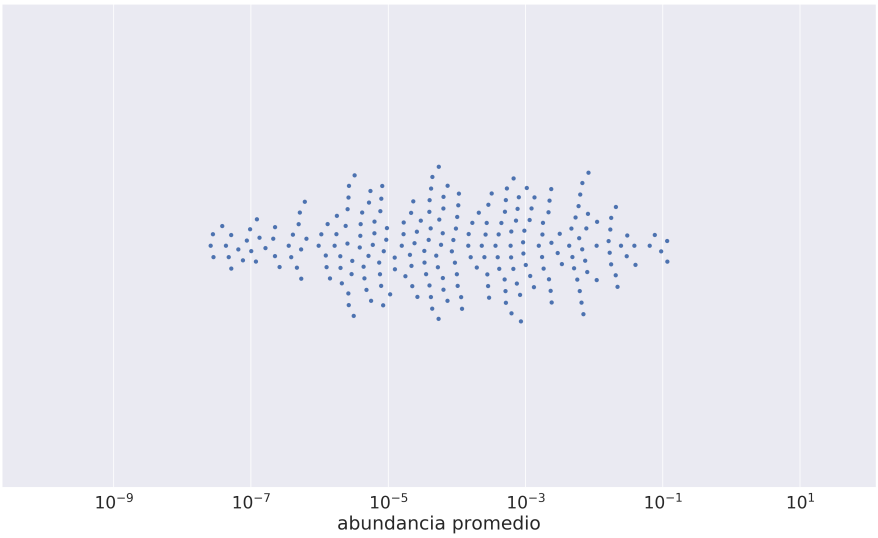

Figura 5.23: Gráfico de enjambre para las abundancias promedio. Conforme se incrementa el número de especies/OTU, se reduce la abundancia promedio de las especies.

Se tienen 170 elementos en el subconjunto $\mathbb{L}4_S$ y 63 en el subconjunto $\mathbb{L}4_I$, dando una razón de 2.7.

$\mathbb{L}4_S$	{0,2,3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,21,22,23,24,25,26,27,30 ,33,34,37,38,40,42,43,48,52,53,54,55,56,57,59,60,61,62,63,64,65,66,70, 71,73,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,97,98, 100,101,102,103,104,105,106,107,108}
$\mathbb{L}4_I$	{1,5,6,9,15,19,20,25,29,39,40,46,47,49,50,51,54,55,58,60,61,62,63,65,66, 68,69,70,71,72,74,75,76,80,82,87,101,102,103,106,108,117,133,134,141,154, 160,164,179,188,191,199,201,204,208,209,212,214,215,220,230,231,232}

Tabla 5.7: Especies con suficientes e insuficientes muestras no nulas Estas especies tienen a lo mucho dos muestras donde no son nulas. Es por esta razón que no dividimos el conjunto de datos.

Observamos que el rendimiento de la GAN es mejor para abundancias promedio mayores, las cuales están directamente relacionados con el número de muestras no nulas, 5.24.

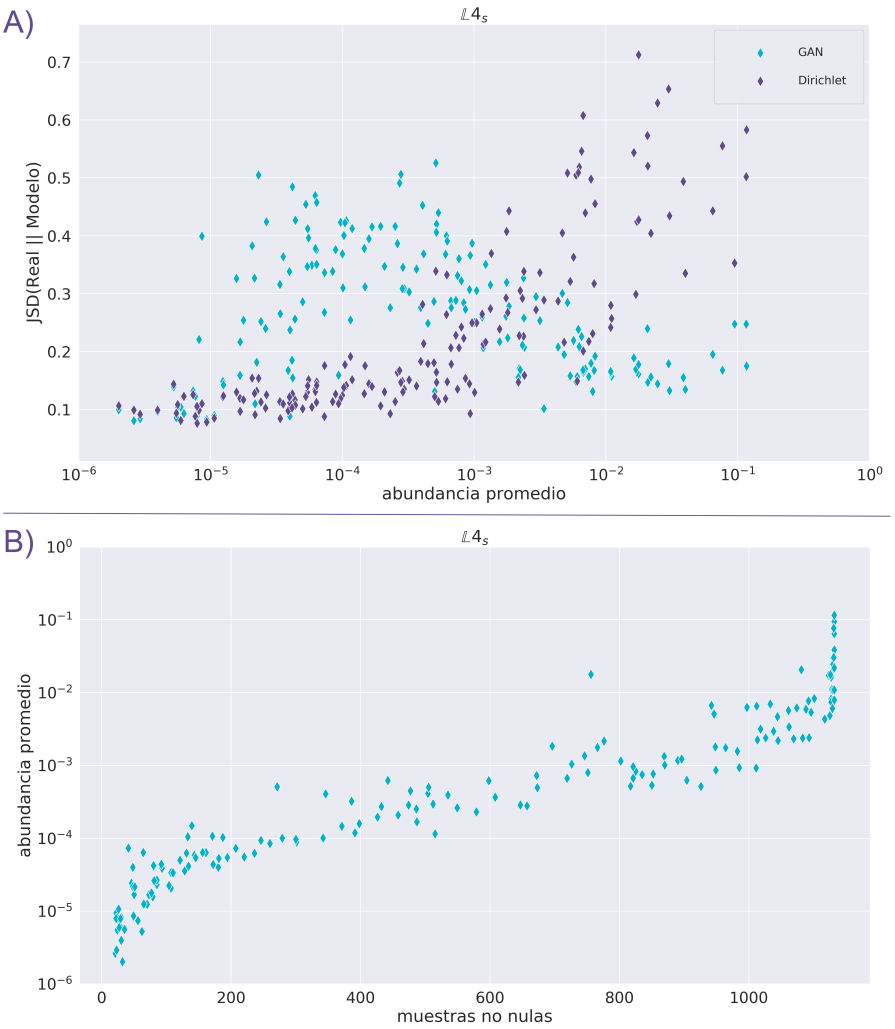

Figura 5.24: Gráfico de dispersión de distancia Jensen-Shannon y abundancia de especie. **A)** la distancia Jensen-Shannon es calculada únicamente para las especies del subconjunto $\mathbb{L}4_S$. **B)** de forma general, la abundancia promedio incrementa con el número de muestras no nulas de la especie.

Si bien no es sorprendente que las muestras generadas por la GAN sean mejores cuando se tienen más muestras no nulas, el modelo de Dirichlet reduce su rendimiento. Por otro lado, para abundancias promedio menores a 10^{-5}, ambos modelos tienen un distancias Jensen-Shannon muy similares. Una posible razón por la cual los dos modelos tienen distancias parecidas es que para abundancias promedio bajas, se tiene pocas muestras no nulas, entonces ambos modelos generan la mayoría de las abundancias nulas.

El modelo de Dirichlet aprende mejor especies con una abundancia promedio pequeña, figura 5.24. Como se tienen más especies, y la misma cantidad de muestras, la mayoría de especie son de baja abundancia promedio, lo cual se refleja una distancia Jensen-Shannon agregada sobresaliente para la muestras del modelo de Dirichlet, figura 5.25.

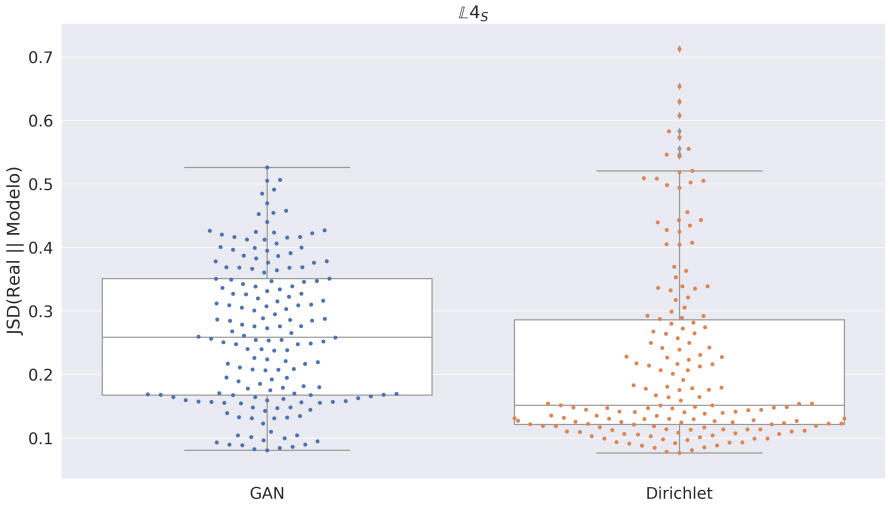

Figura 5.25: Diagrama de caja para la distancia Jensen-Shannon para cada especie del subconjunto $\mathbb{L}4_S$. Al comparar las medias, notamos que el modelo de Dirichlet es mejor que la GAN. Sin embargo, hay algunas excepciones donde el modelo de Dirichlet hace muy mal trabajo.

Al examinar las distancias logarítmicas de ambos modelos, el modelo de Dirichlet supera una vez más a la GAN, figura 5.26

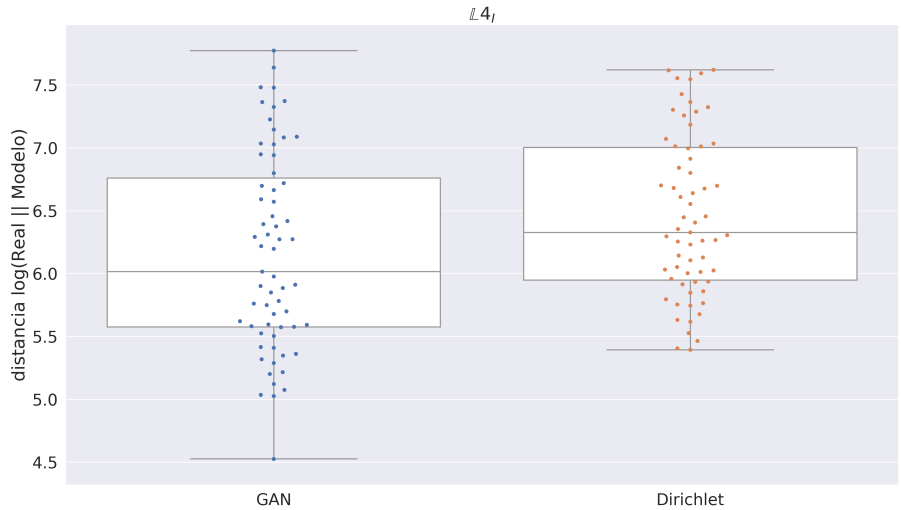

Figura 5.26: Diagrama de caja para la distancia logarítmica para cada especie del subconjunto $\mathbb{L}4_I$. El modelo de Dirichlet tiene una media mayor de distancias logarítmicas, aunque en algunas especies la GAN tiene valores notablemente más altos.

Las especies donde la GAN genera mejores muestras que el modelo de Dirichlet son las especies 8 y 78, con 756 y 1130 muestras no nulas respectivamente. Además de la alta cantidad de muestras no nulas, también hay que notar que esta especie tiene varias muestras con abundancias mayores a 10^{-2}, las cuales el modelo de Dirichlet genera con dificultad.

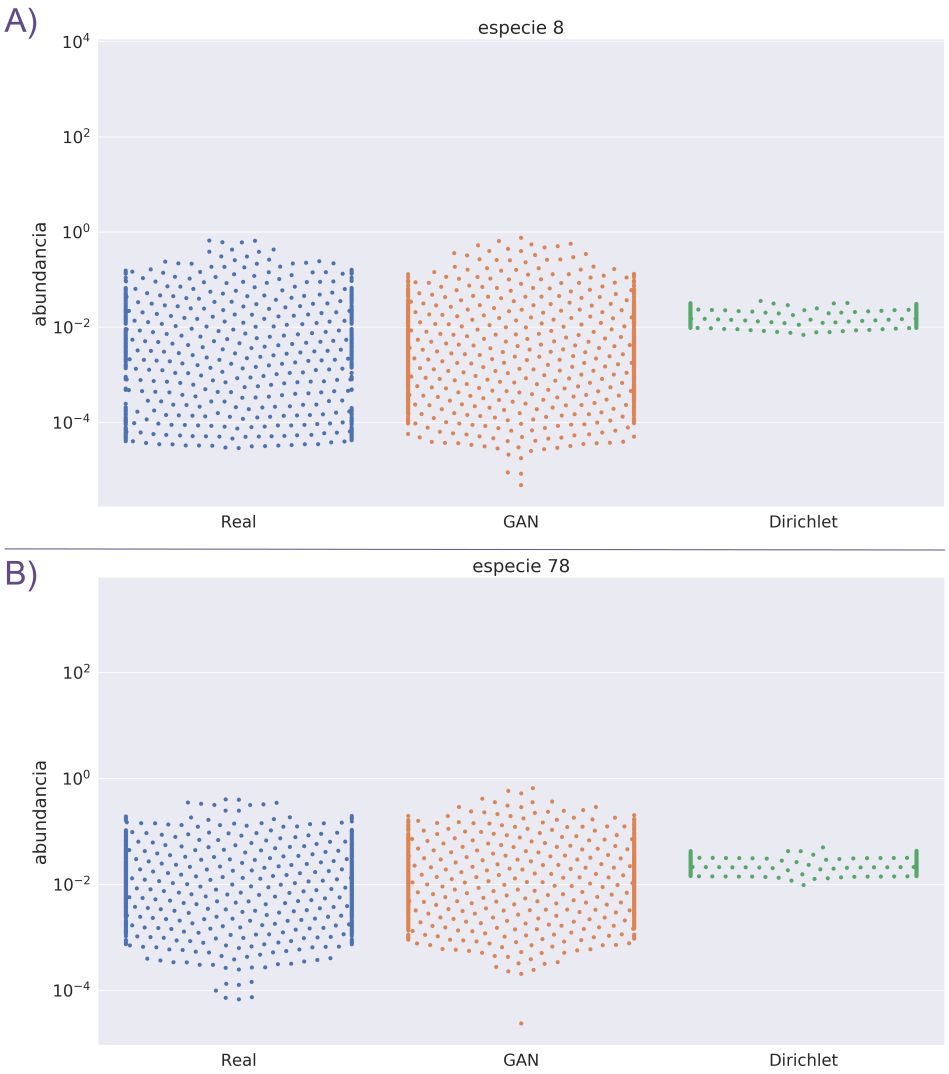

Figura 5.27: Gráfico de enjambre para las abundancias promedio de las especies 8 y 78. **A)** Esta especie tiene 756 muestras no nulas. **B)** Esta especie tiene 1130 muestras no nulas.

Por otro lado, la especie donde la GAN y el modelo de Dirichlet generan muestras muy parecidas es la especie 99, con 982 muestras no nulas. Una característica de ésta especie, es que la mayoría de muestras tienen una abundancia menor a 10^{-21}, lo cual ayuda al modelo de Dirichlet, encontraste de la especies 8 y 78, figura 5.27.

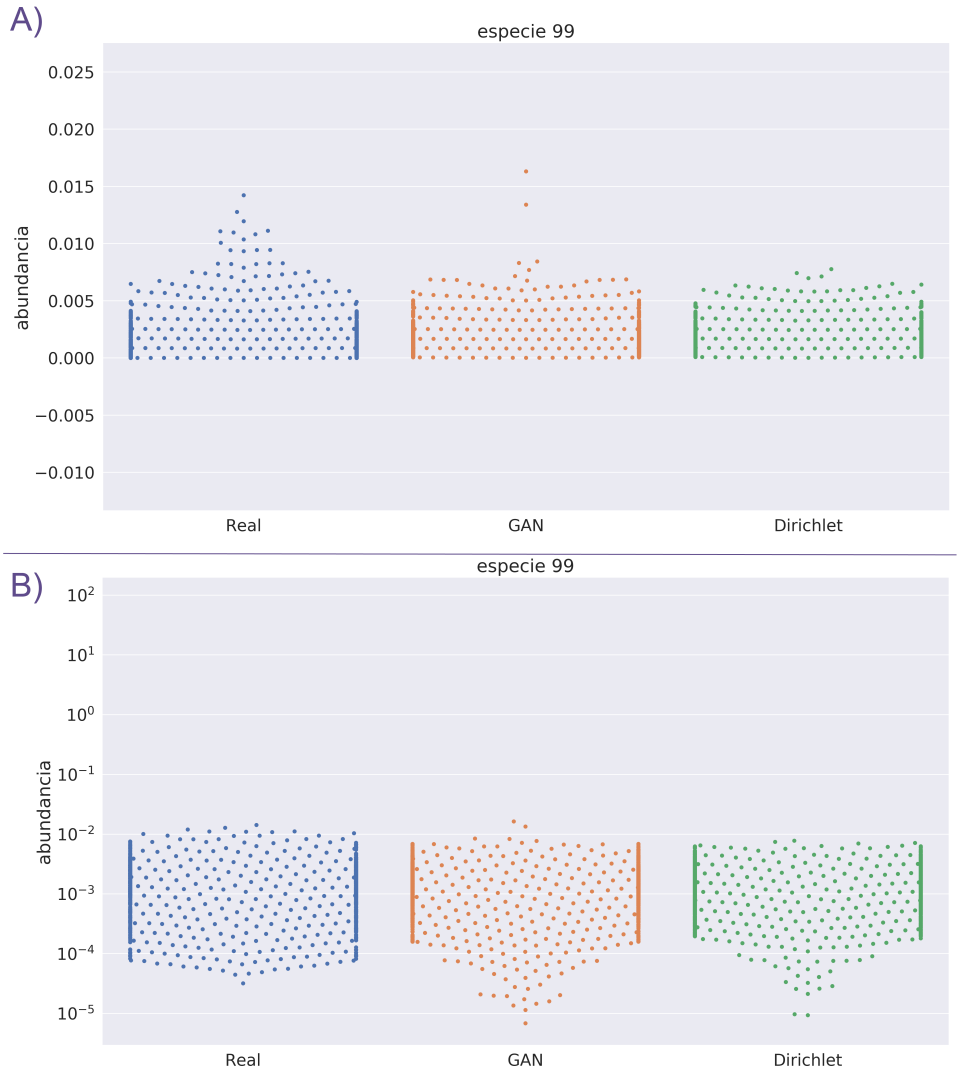

Figura 5.28: Gráfico de enjambre para las abundancias promedio de la especie 99. Esta especie tiene 982 muestras no nulas. El modelo de Dirichlet genera mejores muestras cuando las abundancias de la especie son bajas.

5.2.4. L5

Este conjunto de datos tiene 1132 muestras y 365 especies/OTU. Como se tienen más especies, las abundancias promedio por especie son menores. Entonces, al igual que para el conjunto de datos L3, dividimos las especies en dos subconjuntos para su evaluación, $\mathbb{L}5_S$ y $\mathbb{L}5_I$ con un umbral de 20 especies, tabla 5.8.

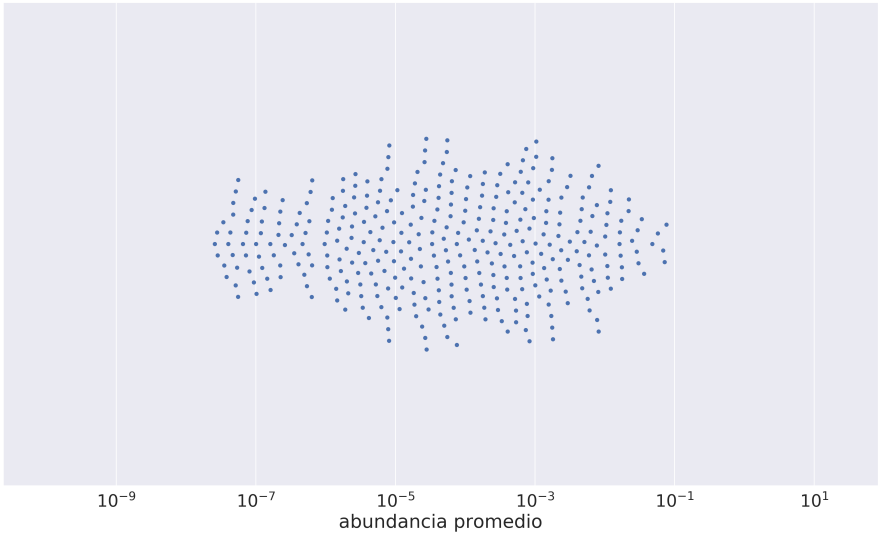

Figura 5.29: Gráfico de enjambre para las abundancias promedio. Conforme se incrementa el número de especies/OTU, se reduce la abundancia promedio de las especies.

Se tienen 260 elementos en el subconjunto $\mathbb{L}5_S$ y 105 en el subconjunto $\mathbb{L}5_I$, dando una razón de 2.48.

$\mathbb{L}5_S$	{0,2,3,4,5,9,10,12,14,15,16,17,18,20,21,22,23,25,26,29,30,31,32,34,35, 36,38,39,40,41,43,44,45,46,47,48,49,50,51,52,53,56,57,58,59,63,65,70,75, 76,82,84,90,94,100,104,105,107,110,115,116,117,123,125,126,127,128,130, 131,132,133,134,135,136,137,138,139,140,141,142,143,144,148,149,150,151, 152,153,155,157,158,159,160,161,162,163,164,166,167,168,169,170,171,172, 173,174,175,176,177,178,179,180,181,184,185,186,187,188,189,191,192,193, 194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,211,212, 213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,229,230,231, 232,234,235,237,238,239,241,242,243,244,245,246,247,248,249,250,251,252, 253,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270, 273,274,275,276,277,278,279,280,281,282,283,284,285,286,287,288,295,296, 297,299,302,303,304,306,308,309,310,311,312,313,314,316,317,318,319,320, 322,323,326,327,328,330,331,332,335,336,338,341,342,343,344,346,347,348, 349,351,352,353,354,355,356,357,358,359,360,361}
$\mathbb{L}5_I$	{1,6,7,8,11,13,19,24,27,28,33,37,42,54,55,60,61,62,64,66,67,68,69,71, 72,73,74,77,78,79,80,81,83,85,86,87,88,89,91,92,93,95,96,97,98,99,101, 102,103,106,108,109,111,112,113,114,118,119,120,121,122,124,129,145, 146,147,154,156,165,182,183,190,210,228,233,236,240,271,272,289,290, 291,292,293,294,298,300,301,305,307,315,321,324,325,329,333,334,337,339, 340,345,350,362,363,364}

Tabla 5.8: Especies con suficientes e insuficientes muestras no nulas Estas especies tienen a lo mucho dos muestras donde no son nulas. Es por esta razón que no dividimos el conjunto de datos.

Observamos que el rendimiento de la GAN es mejor para abundancias promedio mayores, las cuales están directamente relacionados con el número de muestras no nulas, 5.30. Al igual que en los conjuntos de datos anteriores, la GAN es particularmente buena para especies con abundancia promedio mayor a 10^{-3}

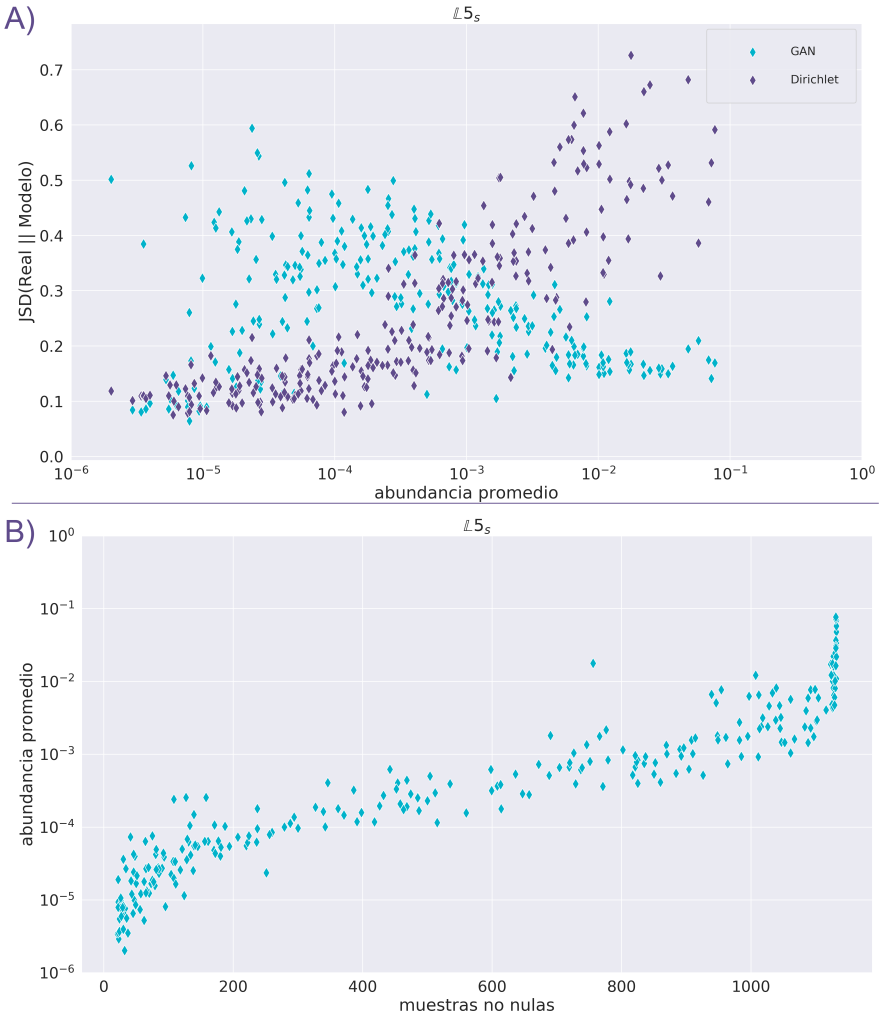

Figura 5.30: Gráfico de dispersión de distancia Jensen-Shannon y abundancia de especie. **A)** la distancia Jensen-Shannon es calculada únicamente para las especies del subconjunto $\mathbb{L}5_S$. **B)** de forma general, la abundancia promedio incrementa con el número de muestras no nulas de la especie

Una vez más, el modelo de Dirichlet aprende mejor especies con una abundancia promedio pequeña, figura 5.30. Sin embargo, la media de distancia Jensen-Shannon por especie, es esperado, pues tenemos más especies y la misma cantidad de muestras 5.31.

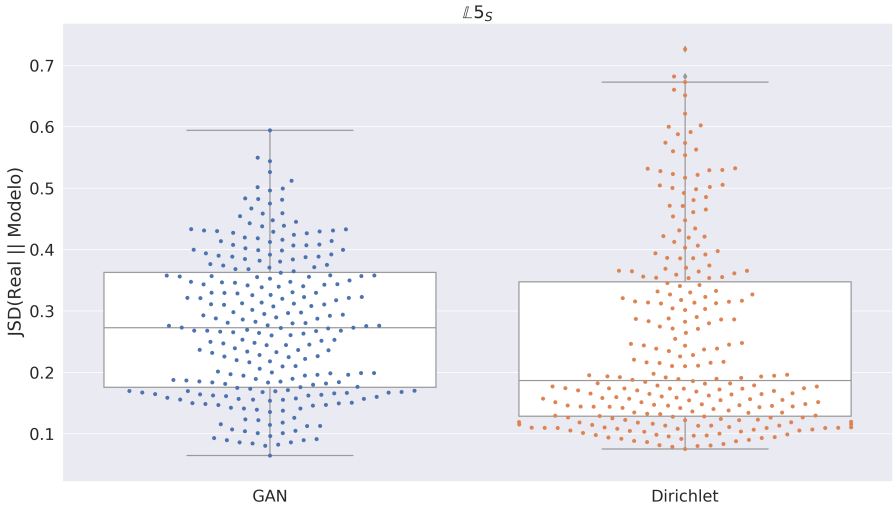

Figura 5.31: Diagrama de caja para la distancia Jensen-Shannon para cada especie del subconjunto $\mathbb{L}5_S$. Al comparar las medias, notamos que el modelo de Dirichlet es mejor que la GAN. Sin embargo, hay algunas excepciones donde el modelo de Dirichlet hace muy mal trabajo.

Una vez más, el modelo de Dirichlet genera mejores muestras de abundancia para las especies de $\mathbb{L}5_I$.

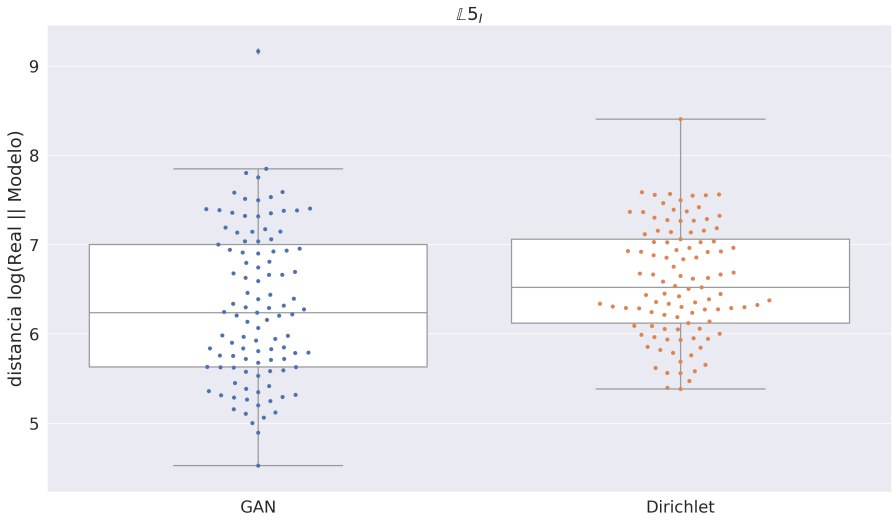

Figura 5.32: Diagrama de caja para la distancia logarítmica para cada especie del subconjunto $\mathbb{L}5_I$. El modelo de Dirichlet tiene una media mayor de distancias logarítmicas, aunque en algunas especies la GAN tiene valores notablemente más altos.

Las especies donde la GAN genera mejores muestras que el modelo de Dirichlet son las especies 12, 26, 105, 144 con 756, 1127, 1130, y 1088 especies no nulas respectivamente, figura 5.33.

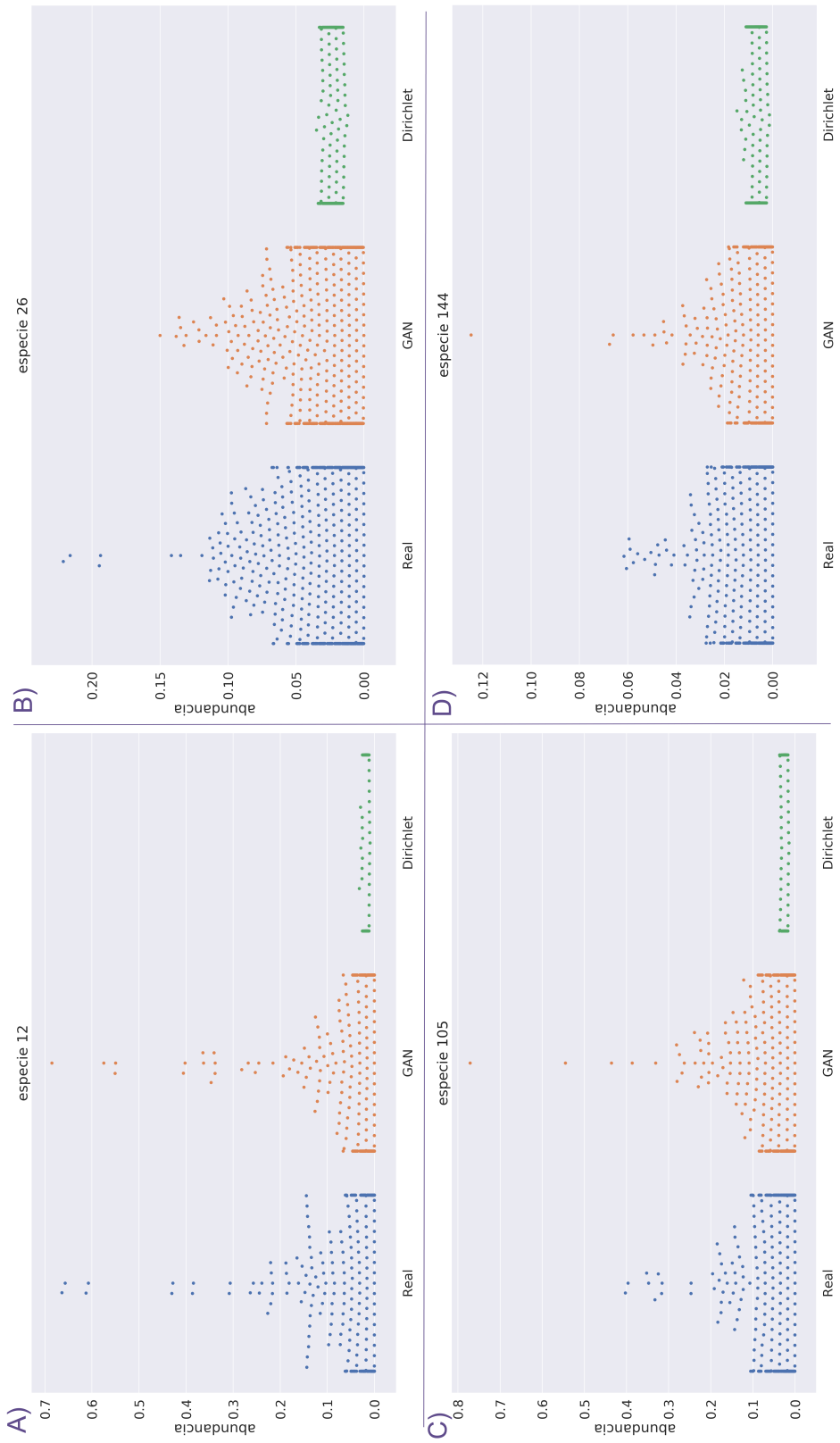

Figura 5.33: Gráfico de enjambre para las abundancias promedio de las especies 12, 26, 105, y 144. Todas las especies tienen un común tener muestras con abundancias mayores a 10^{-2} y tener más de 600 muestras no nulas.

Por otro lado, la especie donde la GAN y el modelo de Dirichlet generan muestras muy parecidas es las especies 59 y 209, con 486 y 1047 muestras no nulas respectivamente.

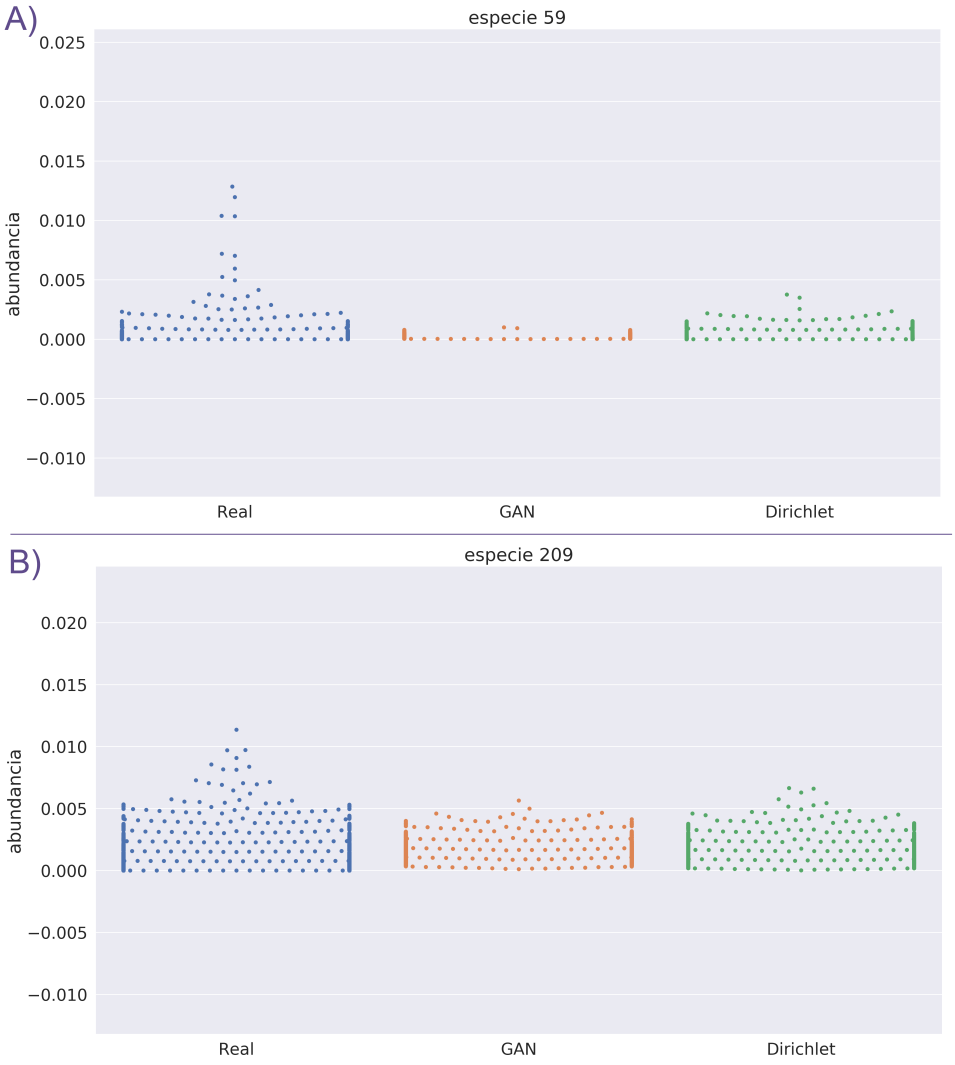

Figura 5.34: Gráfico de enjambre para las abundancias promedio de las especies 59 y 209. La mayoría de muestras tienen una abundancia menor a 10^{-2} .

5.2.5. L6

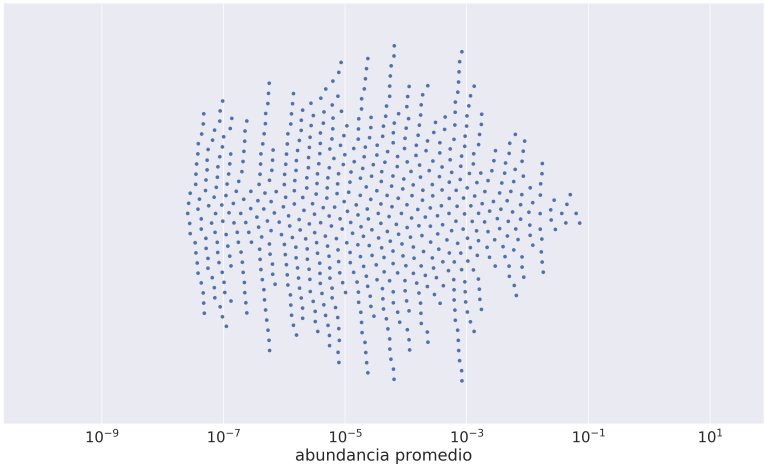

Figura 5.35: Gráfico de enjambre para las abundancias promedio. Conforme se incrementa el número de especies/OTU, se reduce la abundancia promedio de las especies.

Este conjunto de datos tiene 1132 muestras y 654 especies/OTU. Como se tienen más especies, las abundancias promedio por especie son menores. Entonces, al igual que para el conjunto de datos L3, dividimos las especies en dos subconjuntos para su evaluación, $\mathbb{L}6_S$ y $\mathbb{L}6_I$ con un umbral de 20 especies, tabla 5.9.

Se tienen 414 elementos en el subconjunto $\mathbb{L}6_S$ y 240 en el subconjunto $\mathbb{L}6_I$, dando una razón de 1.73.

$\mathbb{L}6_S$	{0,2,3,4,5,7,14,15,17,18,21,22,23,24,25,27,28,29,30,32,34,35,38,39,40,41,42,44,45,46,48,49,50,51,53,54,55,56 ,57,58,59,60,61,62,63,64,67,68,69,70,74,76,81,86,87,93,95,102,106,112,116,117,118,121,131,132,134,135,136,139 ,147,150,151,152,153,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,173,174,175,176,177,178,180 ,181,182,184,186,187,188,189,190,191,192,193,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210 ,211,212,214,215,216,219,220,221,222,223,224,225,226,227,228,229,231,232,234,235,236,237,238,239,240,241,242 ,243,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,261,262,263,265,266,268,269,270,271,272,273 ,274,275,278,279,280,284,285,288,290,291,294,295,297,298,299,300,304,305,306,307,308,309,310,311,312,313,314 ,317,318,319,320,321,322,324,325,328,330,334,335,336,337,339,340,341,342,344,345,346,347,348,349,350,356,357 ,358,361,364,366,370,371,372,373,374,376,377,378,381,384,387,390,392,394,396,397,398,399,402,404,408,409,410 ,412,413,415,416,417,418,421,422,423,424,425,426,427,430,431,432,433,434,435,436,437,438,439,440,441,442,443, 444,445,446,447,448,449,450,451,452,453,454,455,456,457,460,461,464,467,468,469,470,471,472,478,480,481,482, 483,486,487,488,489,493,494,495,504,505,506,508,511,512,513,517,520,521,523,525,526,529,530,532,536,537,538, 539,540,541,543,544,545,547,548,549,552,554,555,556,557,558,560,563,564,565,566,568,569,570,572,574,576,577, 579,581,583,586,592,595,596,598,601,602,604,607,608,609,612,613,615,618,619,620,621,622,624,625,626,628,629, 630,631,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,648,649,650}
$\mathbb{L}6_I$	{1,6,8,9,10,11,12,13,16,19,20,26,31,33,36,37,43,47,52,65,66,71,72,73,75,77,78,79,80,82,83,84,85,88,89,90,91 ,92,94,96,97,98,99,100,101,103,104,105,107,108,109,110,111,113,114,115,119,120,122,123,124,125,126,127,128 ,129,130,133,137,138,140,141,142,143,144,145,146,148,149,154,170,171,172,179,183,185,194,213,217,218,230,233 ,244,260,264,267,276,277,281,282,283,286,287,289,292,293,296,301,302,303,315,316,323,326,327,329,331,332,333 ,338,343,351,352,353,354,355,359,360,362,363,365,367,368,369,375,379,380,382,383,385,386,388,389,391,393,395 ,400,401,403,405,406,407,411,414,419,420,428,429,458,459,462,463,465,466,473,474,475,476,477,479,484,485,490 ,491,492,496,497,498,499,500,501,502,503,507,509,510,514,515,516,518,519,522,524,527,528,531,533,534,535,542 ,546,550,551,553,559,561,562,567,571,573,575,578,580,582,584,585,587,588,589,590,591,593,594,597,599,600,603 ,605,606,610,611,614,616,617,623,627,632,651,652,653}

Tabla 5.9: Especies con suficientes e insuficientes muestras no nulas Estas especies tienen a lo mucho dos muestras donde no son nulas. Es por esta razón que no dividimos el conjunto de datos.

El modelo de Dirichlet genera mejores muestras para especies con abundancias promedio menores a 10^{-3}, figura 5.2.5 A). Estas especies suelen tener menos de 600 muestras no nulas, figura 5.2.5 B).

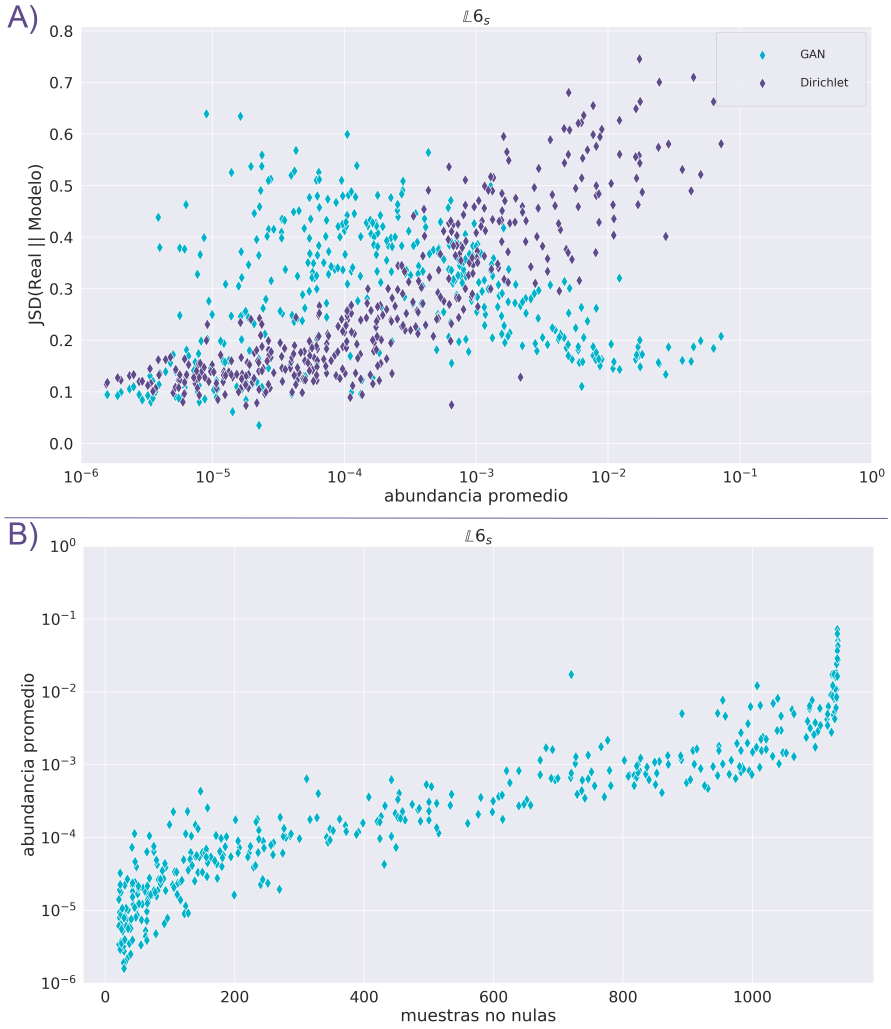

Figura 5.36: Gráfico de dispersión de distancia Jensen-Shannon y abundancia de especie. **A)** la distancia Jensen-Shannon es calculada únicamente para las especies del subconjunto $\mathbb{L}6_S$. **B)** de forma general, la abundancia promedio incrementa con el número de muestras no nulas de la especie.

Si bien el número de especies incrementa considerablemente, la tendencia del aprendizaje es consistente con los conjuntos de datos experimentales anteriores. La GAN genera mucho mejores muestras abundancias que el modelo de Dirichlet para especies con abundancia promedio mayor a 10^{-3}, mientras que las muestras de abundancia del modelo de Dirichlet empeoran.

Tomando en cuenta, todas las especies de $\mathbb{L}6_S$, el modelo de Dirichlet tiene una mejor media de distancia Jensen-Shannon, figura 5.37. Aunque, la figura A) muestra que esto depende fuertemente de la abundancia promedio de la especie.

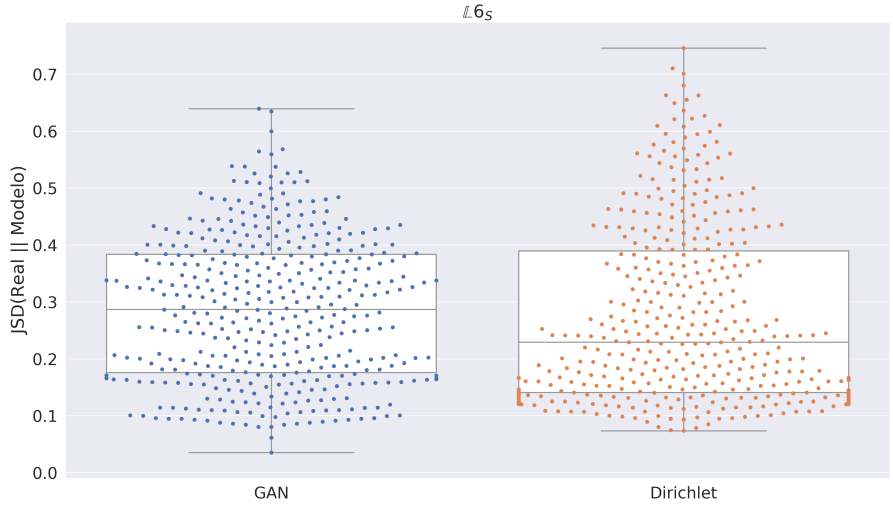

Figura 5.37: Diagrama de caja para la distancia Jensen-Shannon para cada especie del subconjunto $\mathbb{L}6_S$. Al comparar las medias, notamos que el modelo de Dirichlet es mejor que la GAN.

Para especies con pocas muestras no nulas, el modelo de Dirichlet es mejor, figura 5.38. Esto es consistente con los conjuntos de datos experimentales anteriores.

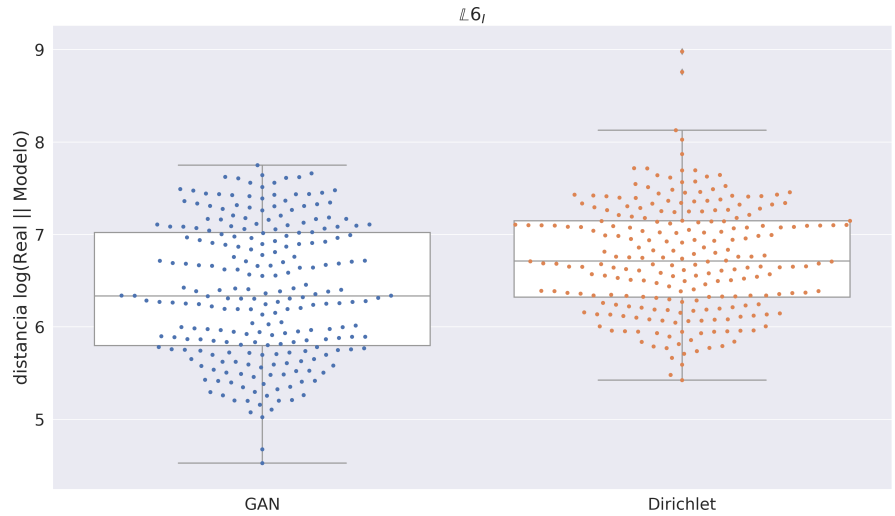

Figura 5.38: Diagrama de caja para la distancia logarítmica para cada especie del subconjunto $\mathbb{L}6_I$. El modelo de Dirichlet tiene una media mayor de distancias logarítmicas.

Las especies donde la GAN genera mejores muestras que el modelo de Dirichlet son las especies 17 y 621, con 720 y 1117 especies no nulas.

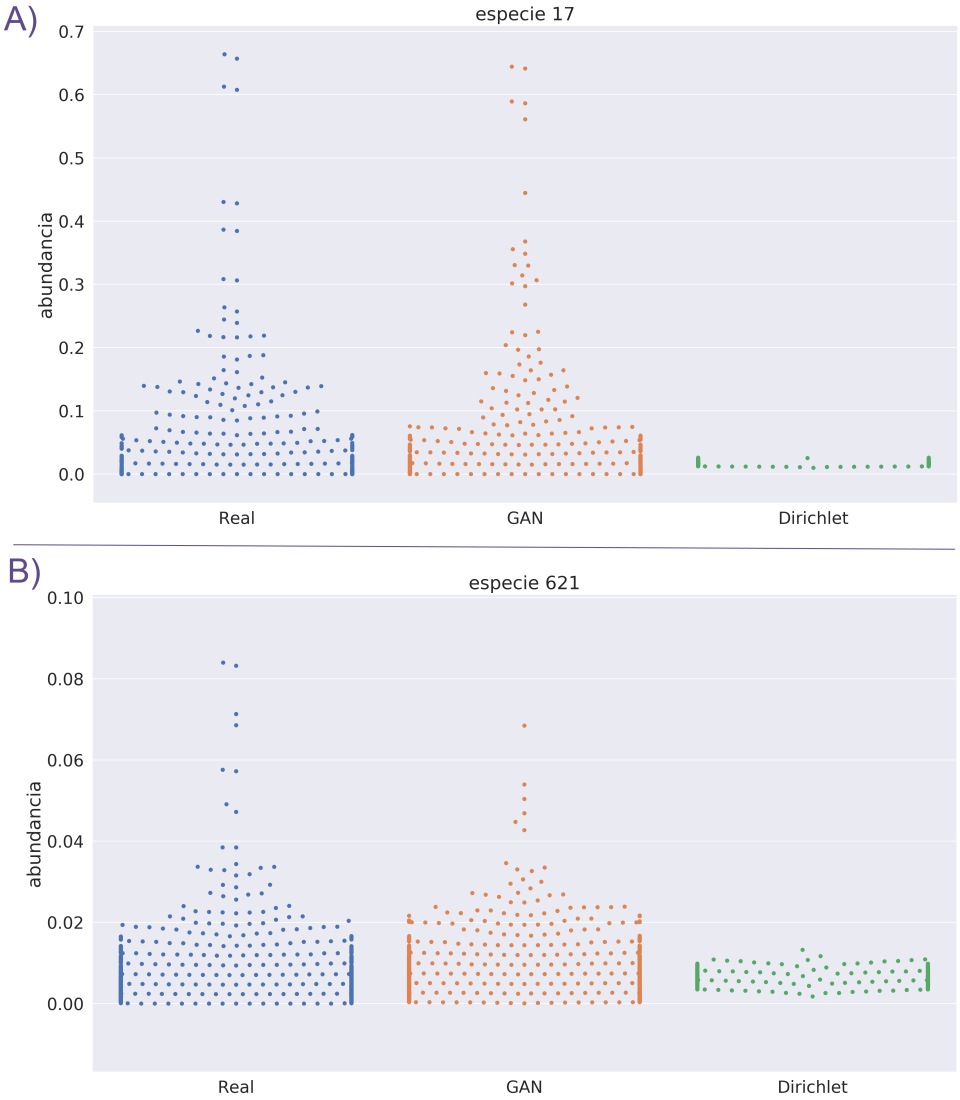

Figura 5.39: Gráfico de enjambre para las abundancias promedio de las especies 17 y 621. **A)** Esta especie tiene 720 muestras no nulas. **B)** Esta especie tiene 1117 muestras no nulas.

Nuevamente, el modelo de Dirichlet tiene dificultades para generar muestras con abundancias mayores a 10^{-2}.

Por otro lado, las especies donde la GAN y el modelo de Dirichlet generan muestras con distancias parecidas con las especies 168 y 286 con 982 y 6 muestras no nulas. Estas especies tienen la característica de que la mayoría de muestras reales tienen una abundancia menor a 10^{-2}.

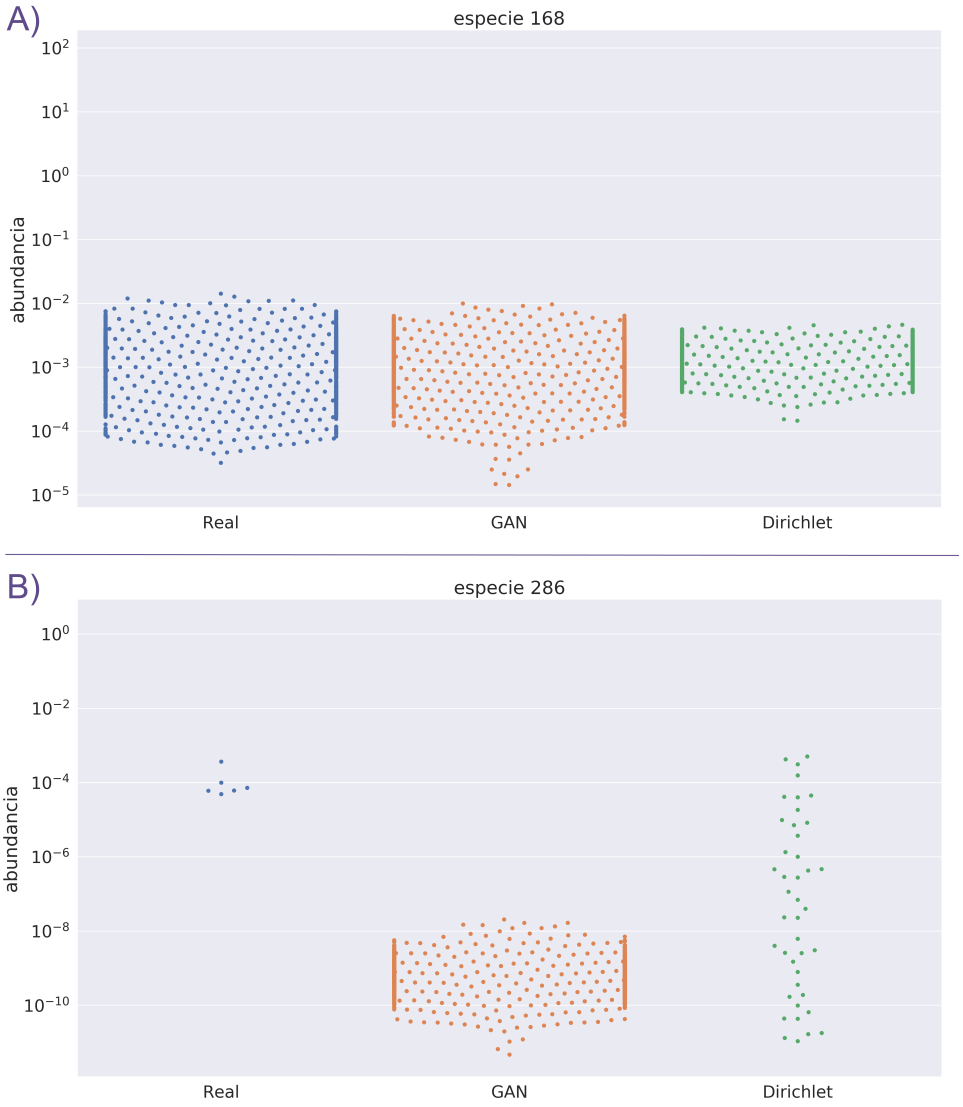

Figura 5.40: Gráfico de enjambre para las abundancias promedio de las especies 168 y 286. Estas especies tienen 982 y 6 muestras no nulas.

5.3. Resumen de resultados en datos experimentales

1. La GAN al ser un método basado en *Deep Learning*, genera mejores muestras conforme más datos de entrenamiento se tienen. Para los conjuntos de datos experimentales, observamos que la GAN genera mejores muestras comparadas con el modelo de Dirichlet, cuando una especie tiene una abundancia promedio mayor a 10^{-3}. La abundancia promedio de una especie está directamente relacionada con el número de muestras no nulas.

2. El modelo de Dirichlet no genera buenas muestras de abundancia cuando en las muestras reales la abundancia es mayor a 10^{-2}. Sin embargo, para abundancias menores, y cuando hay pocos datos no nulos, genera mejores muestras de la GAN.

3. Debido a la limitación de datos, la GAN tiene problemas para aprender algunas especies. Mientras el modelo de Dirichlet genera mejores muestras cuando las especies tienen abundancia promedio pequeña. Los diagramas de dispersión de abundancia promedio y distancia Jensen-Shannon, muestran que ambos modelos se pueden complementar. La GAN se podría utilizar para generar muestras cuando la abundancia promedio de la especie es mayor a 10^{-3}, mientras que el modelo de Dirichlet sería usado para especies con menor abundancia promedio a 10^{-3}.

5.4. Recomendaciones para GAN que genera muestras de datos de microbioma

A continuacion, se enlistan recomendaciones para una GAN que genere datos composicionales:

- Utilizar la función de activación Softmax en la capa de salida del generador. En la sección 5.1.1 mostramos que tiene un rendimiento superior a una activación ReLu de un perceptrón multicapa.
- Un *learning rate* grande puede causar un comportamiento errático en el proceso de aprendizaje de la GAN. Por ello, se recomienda utilizar un *learning rate* constante cuando se entrena una GAN con diferentes conjunto de datos.
- Incrementar el número de nodos tiene mejores resultados que incrementar la profundidad de la GAN. Sin embargo, algún otro conjunto de datos de datos composicionales podría beneficiarse de incrementar el número de capas.
- Utilizar un modelo de Dirichlet y la GAN pueden generar mejores muestras de abundancias cuando se tienen pocos datos. La GAN genera buenas muestras cuando se tiene abundancias promedio mayores a 10^{-3} y el modelo de Dirichlet genera muestras de abundancias buenas para especies con abundancia menor a 10^{-2}.

Capítulo 6

Conclusiones y perspectivas

6.1. Conclusiones

La función Softmax como función de activación en la capa de salida del generador resulta en un mejor proceso de aprendizaje comparada con un generador perceptrón multicapa simple, tal como se ve en la sección 5.1.1. La función Softmax introduce una estructura de probabilidades a los datos generados, la cual es una de las interpretaciones de los datos composicionales. Si bien la función Softmax es usada en clasificadores multiclase, las GAN son un modelo generativo.

Basándose en el error composicional agregado definido en la sección , observamos que la pequeña cantidad de datos experimentales de microbiomas de suelo limita la capacidad de la GAN en generar buenas muestras. En los conjuntos de datos experimentales L2, L3, L4, L5, L6 tenemos 1132 muestras en cada uno, pero la dimensionalidad o número de especies es diferente en cada conjunto de datos, tenemos 36, 112, 233, 365, 654 especies respectivamente. Con mayor número de especies, incrementa el número de muestras nulas, lo cual dificulta el aprendizaje la GAN para generar abundancias de algunas especies.

6.2. Resumen de las contribuciones

1. Se desarrolló una arquitectura de GAN para datos composicionales basada en la distancia de Wasserstein con penalización se gradiente. Una de sus principales características es el uso de la función Softmax en la capa de salida del generador.
2. Se creó una metodología para evaluar el proceso de aprendizaje de una GAN que genera muestras de una distribución de probabilidad de datos composicionales.
3. Se validó la GAN creada con datos composicionales obtenidos de tomar muestras de diferentes distribuciones de Dirichlet. Observamos que la GAN que proponemos tienen un proceso de aprendizaje más eficiente en términos de cantidad de muestras e iteraciones de entrenamiento.
4. Se realizo una validación detallada de la GAN en datos experimentales de microbioma de suelo, encontrando que la arquitectura de GAN propuesta tiene un buen desempeño si existen suficientes datos de entrenamiento.

6.3. Perspectivas y trabajo futuro

En repetidas ocasiones, las especies donde la GAN hace un trabajo deficiente en generar abundancias son las especies donde el modelo de Dirichlet hace un mejor trabajo. El modelo de Dirichlet no genera buenas muestras de abundancia cuando en las muestras reales la abundancia es mayor a 10^{-2}. Sin embargo, para abundancias menores, y cuando hay pocos datos no nulos, genera mejores muestras de la GAN. Entonces, *debido a que los modelos se complementan en situaciones con pocos datos, debido a que los modelos se complementan en situaciones con pocos datos, crear una combinación de los modelos, conocido como ensemble learner en inglés, es una alternativa interesante a la GAN o modelo de Dirichlet para el problema de aprender las distribuciones de los datos experimentales que tienen pocas muestras.*

Índice de tablas

Índice de figuras

Índice de referencias

[1] Arjovsky, M., Chintala, S. y Bottou, L. «Wasserstein GAN». En: *ArXiv e-prints* (ene. de 2017). arXiv: 1701.07875 [stat.ML].

[2] Council, N.R. y col. *The New Science of Metagenomics: Revealing the Secrets of Our Microbial Planet*. National Academies Press, 2007, págs. 15, 16, 17, 18, 19. ISBN: 9780309106764.

[3] Fernandes, Andrew y col. «ANOVA-like differential expression (ALDEx) analysis for mixed population RNA-seq». En: *PloS one* vol. 8 (jul. de 2013), e67019. DOI: 10.1371/journal.pone.0067019.

[4] Filzmoser, Peter, Hron, Karel y Templ, Matthias. *Applied Compositional Data Analysis: With Worked Examples in R*. Springer International Publishing, 2018.

[5] Glendinning, Laura y Free, Andrew. «Supra-organismal interactions in the human intestine». En: *Frontiers in cellular and infection microbiology* vol. 4 (abr. de 2014). PMC4005949[pmcid], págs. 47-47. ISSN: 2235-2988. DOI: 10.3389/fcimb.2014.00047. URL: https://www.ncbi.nlm.nih.gov/pubmed/24795867.

[6] Gloor, Gregory B. y col. «Microbiome Datasets Are Compositional: And This Is Not Optional». En: *Frontiers in Microbiology* vol. 8 (2017), pág. 2224. ISSN: 1664-302X. DOI: 10.3389/fmicb.2017.02224. URL: https://www.frontiersin.org/article/10.3389/fmicb.2017.02224.

[7] Goodfellow, I. J. y col. «Generative Adversarial Networks». En: *ArXiv e-prints* (jun. de 2014). arXiv: 1406.2661 [stat.ML].

[8] Goodfellow, Ian J. «NIPS 2016 Tutorial: Generative Adversarial Networks». En: *CoRR* vol. abs/1701.00160 (2017). arXiv: 1701.00160. URL: http://arxiv.org/abs/1701.00160.

[9] Goodfellow, Ian, Bengio, Yoshua y Courville, Aaron. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[10] Gulrajani, Ishaan y col. «Improved Training of Wasserstein GANs». En: *CoRR* vol. abs/1704.00028 (2017). arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028.

[11] He, Kaiming y col. «Deep Residual Learning for Image Recognition». En: *CoRR* vol. abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[12] Isola, Phillip y col. «Image-to-Image Translation with Conditional Adversarial Networks». En: *CoRR* vol. abs/1611.07004 (2016). arXiv: 1611.07004. URL: http://arxiv.org/abs/1611.07004.

[13] Karczewski, Jacek y col. «The effects of the microbiota on the host immune system.» En: *Autoimmunity* vol. 47, n.º 8 (2014), págs. 494-504. ISSN: 1607-842X. URL: http://search.ebscohost.com/login.aspx?direct=true&AuthType=shib&db=mnh&AN=25019177&lang=es&site=ehost-live&custid=s2037900.

[14] Karras, Tero y col. «Progressive Growing of GANs for Improved Quality, Stability, and Variation». En: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=Hk99zCeAb.

[15] Ledig, Christian y col. «Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network». En: *CoRR* vol. abs/1609.04802 (2016). arXiv: 1609.04802. URL: http://arxiv.org/abs/1609.04802.

[16] McMurdie, Paul J. y Holmes, Susan. «Waste Not, Want Not: Why Rarefying Microbiome Data Is Inadmissible». En: *PLOS Computational Biology* vol. 10, n.º 4 (abr. de 2014), págs. 1-12. DOI: 10.1371/journal.pcbi.1003531. URL: https://doi.org/10.1371/journal.pcbi.1003531.

[17] *Measures of Similarity and Dissimilarity: STAT 508*. URL: https://online.stat.psu.edu/stat508/lesson/1b/1b.2/1b.2.1.

[18] Mehryar Mohri, Afshin Rostamizadeh y Ameet Talwalkar. *Foundations of Machine Learning (Adaptive Computation and Machine Learning series)*. Cambridge, Massachusetts: The MIT Press, 2018. ISBN: 978-0262039406.

[19] Minka, Thomas O. *Estimating a Dirichlet distribution*. 2012. URL: https://tminka.github.io/papers/dirichlet/.

[20] Ng, Kai Wang, Tian, Guo-Liang y Tang, Man-Lai. *Dirichlet and related distributions: theory, methods and applications*. eng. Wiley series in probability and statistics. OCLC: 837772619. Chichester: Wiley, 2011. ISBN: 9781119998419 9780470688199.

[21] Org, Imagenet. *Large Scale Visual Recognition Challenge 2015 (ILSVRC2015). (n.d.)* http://image-net.org/challenges/LSVRC/2015/ . 2015 (accessed October 17, 2018).

[22] Pathak, Deepak y col. «Context Encoders: Feature Learning by Inpainting». En: *CoRR* vol. abs/1604.07379 (2016). arXiv: 1604.07379. URL: http://arxiv.org/abs/1604.07379.

[23] Pawlowsky-Glahn, Vera, Egozcue, Juan J. y Tolosana-Delgado, Raimon. *Modeling and analysis of compositional data*. John Wiley & Sons, Inc., 2015.

[24] Pepper, John W. y Rosenfeld, Simon. «The emerging medical ecology of the human gut microbiome». En: *Trends in Ecology & Evolution* vol. 27, n.º 7 (2012), págs. 381-384. ISSN: 0169-5347. URL: http://www.sciencedirect.com/science/article/pii/S0169534712000663.

[25] Qin, Nan y col. «Influence of H7N9 virus infection and associated treatment on human gut microbiota». En: *Scientific Reports* vol. 5 (oct. de 2015). Article, pág. 14771. URL: https://doi.org/10.1038/srep14771.

[26] Salimans, Tim y col. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].

[27] *Ternary Diagram*. URL: http://mathworld.wolfram.com/TernaryDiagram.html.

[28] Terrat, Sébastien y col. «Mapping and predictive variations of soil bacterial richness across France». En: *PLOS ONE* vol. 12 (oct. de 2017). DOI: 10.1371/journal.pone.0186766.

[29] Turnbaugh, Peter J. y col. «The Human Microbiome Project». En: *Nature* vol. 449 (oct. de 2007), pág. 804. URL: http://dx.doi.org/10.1038/nature06244.

[30] Villani, Cédric. *Optimal Transport: Old and New (Grundlehren der mathematischen Wissenschaften)*. Springer, 2008. ISBN: 9788793102132. URL: https://www.amazon.com/Optimal-Transport-Grundlehren-mathematischen-Wissenschaften/dp/3540710493?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3540710493.

[31] Wang, Ting-Chun y col. «High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs». En: *CoRR* vol. abs/1711.11585 (2017). arXiv: 1711.11585. URL: http://arxiv.org/abs/1711.11585.

[32] Weiss, Sophie y col. «Normalization and microbial differential abundance strategies depend upon data characteristics». En: *Microbiome* vol. 5, n.º 1 (2017), pág. 27. ISSN: 2049-2618. DOI: 10.1186/s40168-017-0237-y. URL: https://doi.org/10.1186/s40168-017-0237-y.

[33] Yeh, Raymond A. y col. «Semantic Image Inpainting with Perceptual and Contextual Losses». En: *CoRR* vol. abs/1607.07539 (2016). arXiv: 1607.07539. URL: http://arxiv.org/abs/1607.07539.

[34] Zhang, Han y col. «StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks». En: *CoRR* vol. abs/1612.03242 (2016). arXiv: 1612.03242. URL: http://arxiv.org/abs/1612.03242.