UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

Estudio de Redes Generativas Adversarias

José David Ocampo Rojas

Matemática

Trabajo de titulación presentado como requisito para la obtención

del titulo de

Matemático

Quito, 14 Diciembre de 2022

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

HOJA DE CALIFICACIÓN DE TRABAJO DE FIN DE CARRERA

Estudio de Redes Generativas Adversarias

José David Ocampo Rojas

Nombre del profesor, Titulo académico: Julio Ibarra, MSc.

Quito, 14 de Diciembre de 2022

© Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en la Ley Orgánica de Educación Superior del Ecuador.

Nombres y apellidos:	José David Ocampo Rojas
Código:	00329278
Cédula de Identidad:	1105810822
Lugar y fecha:	Quito, Diciembre de 2022

ACLARACIÓN PARA LA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en `http://bit.ly/COPETheses`

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on `http://bit.ly/COPETheses`

Resumen

Las Redes Generativas Adversarias son un tipo de modelo generativo que enfrenta a dos redes neuronales, a las que llamamos Discriminador y Generador, bajo un juego de minimización-maximización para crear nuevos ejemplos de una categoría dado un conjunto de observaciones. Para esto se adopta un enfoque probabilístico, donde partimos asumiendo que el conjunto de entrenamiento tiene una distribución subyacente que podemos aproximar iterativamente a través de distribuciones generadas que finalmente convergen a la distribución real.

En este trabajo desarrollamos a detalle la matemática que sustenta a las Redes Generativas Adversarias y hacemos una implementación sobre la base de datos de dígitos escritos a mano MNIST. Los resultados indican que las imágenes evolucionan gradualmente desde ruido absoluto hacia imágenes indistinguibles de los ejemplos de partida a medida que se entrena el modelo, y valores de salida del Discriminador concuerdan con las predicciones teóricas.

Palabras clave: *modelos generativos, modelos discriminativos, redes neuronales, probabilidad y estadística, aprendizaje automático.*

Abstract

Generative Adversarial Networks are a type of generative model that confronts two neural networks, which we call Discriminator and Generator, under a minimization-maximization game to create new examples of a category given a set of observations. For this, a probabilistic approach is adopted, where we start by assuming that the training set has an underlying distribution that we can approximate iteratively through generated distributions that finally converge to the real distribution.

In this work we develop in detail the mathematics that supports the Generative Adversarial Networks and we make an implementation on the MNIST database of handwritten digits images. The results indicate that the images gradually evolve from absolute noise towards images indistinguishable from the starting samples as the model is trained, and output values of the Discriminator agree with the theoretical predictions.

Keywords: *generative models, discriminative models, neural networks, probability and statistics, machine learning.*

Índice general

Índice de cuadros **9**

Índice de figuras **10**

1. Introducción **12**

1.1. Motivación 12

1.2. Modelación discriminativa vs Modelación generativa 13

1.3. Distancia entre distribuciones 17

1.4. Redes generativas adversarias (GANs) 19

1.4.1. Óptimo global 22

1.4.2. Método del subgradiente 25

1.4.3. Convergencia del algoritmo 27

2. Experimentos **31**

2.1. Base de datos MNIST 32

2.2. Arquitectura de la red 33

2.3. Resultados 34

3. Conclusiones **37**

Bibliografía **39**

Índice de cuadros

2.1. Hiperparámetros y parámetros del optimizador 32

2.2. Arquitectura del discriminador 33

2.3. Arquitectura del generador 34

Índice de figuras

1.1. Imágenes generadas artificialmente en `https://www.thispersondoesnotexist.com/` por una versión avanzada del método de Redes Generativas Adversarias conocido como StyleGAN2 [12] 13

1.2. Ejemplo de la relación entre un vector de características $\boldsymbol{x}$, en este caso el conjunto de pixeles de una imagen 28×28 de un número del 0 al 9, y su variable de respuesta asociada y, que representa el número al que corresponde la imagen. Cada pixel está representado por un número entre 0 y 255, que denota su intensidad en la escala de grises. 14

1.3. Diagrama general de las GANs. A partir de un conjunto de entrenamiento optimizamos una red neuronal que llamamos Discriminador para distinguir entre elementos del conjunto y elementos fuera del conjunto asignando un valor cercano 1 a los primeros y cercano a 0 a los segundos. Luego entrenamos una red neuronal llamada Generador para mapear vectores z con distribución definida a priori hacia el espacio de los datos del conjunto de entrenamiento, de tal forma que estos elementos sean indistinguibles de los elementos reales. 19

2.1. Ejemplos de la base de datos MNIST[6] 32

2.2. Imágenes generadas para diferentes iteraciones. Inicialmente en **(a)** las imágenes generadas son ruido puro, pero a medida que entrenamos la red **(b)**-**(d)** incrementamos la calidad de las imágenes al punto de ser casi indistinguibles de las imágenes reales. 35

2.3. Probabilidad de pertenecer al conjunto de los datos reales para distintas iteraciones del modelo evaluada en las imágenes generadas por G. 36

Capítulo 1

Introducción

1.1. Motivación

Las Redes Generativas Adversarias o GANs (del inglés *Generative Adversarial Networks*) son un tipo de modelo generativo introducido por Goodfellow et al[9] en 2014, que desde entonces han recibido considerable atención de la comunidad de aprendizaje automático por sus potenciales aplicaciones. Su objetivo es responder a la pregunta: dado un conjunto de objetos con cierto grado de consistencia (por ejemplo, conjuntos de imágenes de personas o una colección de imágenes de tumores en pulmones), ¿es posible generar objetos similares artificialmente? [20].

Aunque a la fecha existen numerosas propuestas para resolver este problema como los Variational Autoencoders [13] o los Deep boltzman machines [18], las GANs son novedosas en tanto aprovechan el desarrollo reciente de los modelos

discriminativos para formular esta pregunta en términos de una competición entre dos redes neuronales (de ahí el nombre adversarias). Recientes avances han mostrado que las GANs son efectivas para, entre otras cosas, generar imágenes hiperrealistas de personas, incrementar de la resolución de imágenes pixeladas y generar imágenes de arte.

(a) (b)

Figura 1.1: Imágenes generadas artificialmente en `https://www.thispersondoesnotexist.com/` por una versión avanzada del método de Redes Generativas Adversarias conocido como StyleGAN2 [12]

1.2. Modelación discriminativa vs Modelación generativa

Dentro del aprendizaje automático podemos distinguir entre dos tipos de modelaciones: discriminativa y generativa.

En la modelación discriminativa el objetivo es predecir una variable de res-

puesta $y \in \mathcal{Y} \subset \mathbb{R}^m$, que puede ser finita, discreta o continua, dado un vector $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ al que llamamos vector de características. Por ejemplo, $\boldsymbol{x}$ puede ser el conjunto de pixeles de una imagen de un dígito entre 0 y 9 en blanco y negro de dimensiones 28×28, donde cada entrada tiene un valor entero entre 0 y 255 representando la intensidad en escala de grises, y y una variable numérica que toma valores en $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, representando el número correspondiente a la imagen.

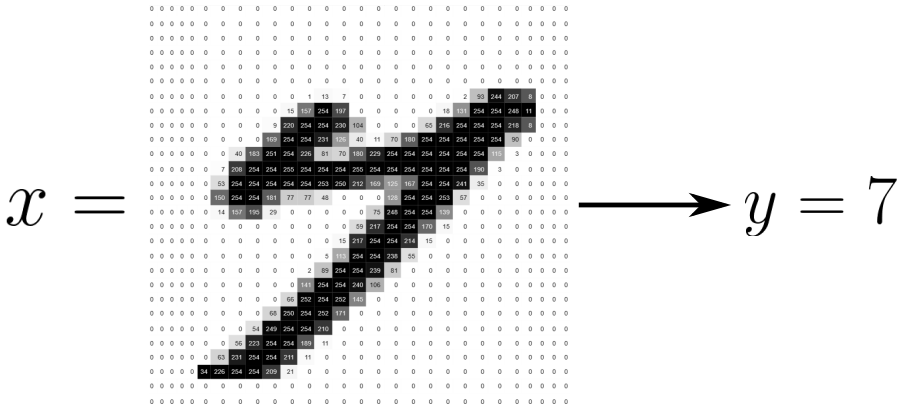


Figura 1.2: Ejemplo de la relación entre un vector de características $\boldsymbol{x}$, en este caso el conjunto de pixeles de una imagen 28×28 de un número del 0 al 9, y su variable de respuesta asociada y, que representa el número al que corresponde la imagen. Cada pixel está representado por un número entre 0 y 255, que denota su intensidad en la escala de grises.

La relación entre $\boldsymbol{x}$ e y se expresa a través de una función $g : \mathcal{X} \to \hat{\mathcal{Y}}$ llamada *función de predicción*, que toma como argumento $\boldsymbol{x}$ y devuelve una predicción del posible valor de y, usualmente expresada como $\hat{y} = g(\boldsymbol{x})$. Observemos que $\hat{\mathcal{Y}}$ no necesariamente es igual a $\mathcal{Y}$, es decir, la predicción $g(\boldsymbol{x})$ no necesariamente pertenece al conjunto que la variable de respuesta y. Finalmente, la precisión de la predicción de $\hat{y}$ respecto a y luego se cuantifica con una *función de pérdida* $L(\hat{y}, y)$

[14].

Difícilmente una función de predicción específica será capaz de hacer predicciones correctas para todos los posibles valores $(\boldsymbol{x}, y)$. Resulta beneficioso entonces adoptar un enfoque probabilístico y asumir que el par $(\boldsymbol{x}, y)$ es el resultado del par aleatorio $(\boldsymbol{X}, Y)$ con densidad de probabilidad conjunta $f(\boldsymbol{x}, y)$. Podemos entonces evaluar el desempeño de la función de predicción g con el valor esperado de la función de perdida , o riesgo:

$$l(g) = \mathbb{E}\Big[L(Y, g(\boldsymbol{X}))\Big] \tag{1.1}$$

En la práctica, es común que la densidad conjunta $f(\boldsymbol{x}, y)$ sea desconocida, y en su lugar tenemos un número finito n de pares $T = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)\}$. Buscamos de esta manera encontrar la función de predicción óptima g^* a partir de este conjunto de observaciones, usualmente llamado conjunto de entrenamiento. Es necesario aclarar que limitamos la búsqueda de g a una clase de funciones definidas a priori G, tal que g generalice a todo $\mathcal{X}$. De otra forma, arriesgamos cometer errores del tipo overfitting, donde g predice con altos niveles de precisión en conjunto de entrenamiento, pero falla en ejemplos fuera del conjunto [14].

Aplicando lo visto hasta ahora al ejemplo de la Figura 1.2, dada la función de predicción que asigna cada imagen a un número entre 0 y 9

$$g : \{0, \cdots, 255\}^{28\times 28} \to \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

una posible función de pérdida sería el valor absoluto de la diferencia entre el valor real y el predicho por g

$$L(y, \hat{y}) = |y - \hat{y}| = |y - g(\boldsymbol{x})| \tag{1.2}$$

Ya que la distribución real usualmente es desconocida o en algunos casos ni siquiera existe, para calcular (1.1) hacemos una estimación basada en la Ley de los grandes números, que establece que para una muestra grande la media es un estimador no sesgado del valor esperado. Por tanto, para el ejemplo anterior un posible riesgo es

$$l_T(g) = \frac{1}{n} \sum_{i=1}^{n} |y_i - g(\boldsymbol{x}_i)| \tag{1.3}$$

En contraste, la modelación generativa intenta aprender la distribución subyacente del conjunto de entrenamiento $T = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$, esto es, $f(\boldsymbol{x})$. g es entonces una aproximación de $f(\boldsymbol{x})$ y el riesgo toma la forma:

$$l(g) = \mathbb{E}\Big[L(f(\boldsymbol{X}), g(\boldsymbol{X}))\Big] \tag{1.4}$$

La idea de los modelos generativos surge de que, independientemente del poder de predicción de un modelo discriminativo, este no es capaz de generar nuevos ejemplos de la clase a la que pertenece el conjunto de entrenamiento. En cambio, al aproximar $f(\boldsymbol{x})$ caracterizamos todos los posibles valores de $\boldsymbol{x}$, y luego podemos por ejemplo usar esta caracterización para encontrar $f(\boldsymbol{x}|y)$.

Retomando el ejemplo de la Figura 1.2, dadas n imágenes de dígitos entre 0 y

9 en blanco y negro, al aproximar la distribución real f podemos obtener nuevas imágenes que estén fuera del conjunto de entrenamiento por métodos como el que veremos en el Apartado 1.4.

1.3. Distancia entre distribuciones

La función de pérdida L en (1.4) usualmente se expresa en términos de divergencias estadísticas, que cuantifican el grado de disimilaridad entre dos distribuciones. En el contexto de las GANs usualmente se usa divergencias que pertenecen a la familia de las $f-divergencias$.

Definición 1.3.1. *Sea $f : (0, \infty) \to \mathbb{R}$ una función estrictamente convexa tal que $f(1) = 0$. Dadas dos funciones de densidad de probabilidad p y q en $\mathbb{R}^n$, la* ***f-divergencia*** *entre p y q se define como*

$$D_f(p\|q) := \mathbb{E}_{\boldsymbol{X}\sim q}\left[f\left(\frac{p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)\right] = \int_{\mathbb{R}^n} f\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) q(\boldsymbol{x})d\boldsymbol{x} \tag{1.5}$$

donde adoptamos la convención $f\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) q(\boldsymbol{x}) = 0$ si $q(\boldsymbol{x}) = 0$.

Teorema 1. *$D_f(p\|q) \geq 0$ con igualdad si y solo si $p = q$.*

Demostración. La no negatividad se sigue inmediatamente de la densigualdad de Shannon, que dice que si f es una función convexa y X es una variable aleatoria se cumple

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \tag{1.6}$$

Consideremos entonces la variable aleatoria $p(\boldsymbol{X})/q(\boldsymbol{X})$ con densidad $q(\boldsymbol{x})$. Tenemos entonces

$$\begin{aligned} D_f(p\|q) &= \mathbb{E}_{\boldsymbol{X}\sim q}\left[f\left(\frac{p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)\right] \\ &\geq f\left(\mathbb{E}_{\boldsymbol{X}\sim q}\left[\frac{p(\boldsymbol{X})}{q(\boldsymbol{X})}\right]\right) \\ &= f\left(\int_{\mathbb{R}^n} q(\boldsymbol{x})\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}d\boldsymbol{x}\right) \\ &= f\left(\int_{\mathbb{R}^n} p(\boldsymbol{x})d\boldsymbol{x}\right) = f\left(1\right) = 0 \end{aligned} \tag{1.7}$$

Ya que f es una función estrictamente convexa, la igualdad se cumple si y solo si nuestra variable aleatoria $p(\boldsymbol{X})/q(\boldsymbol{X})$ es constante, que ocurre precisamente cuando $p = q$. [10]

□

Un caso particular de (1.5) muy usado en el aprendizaje automático es la divergencia de Kullback-Leibler, que resulta de tomar $f(x) = x\log(x)$:

$$D_{KL}(p\|q) := \int_{\mathbb{R}^n} \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) p(\boldsymbol{x})d\boldsymbol{x} \tag{1.8}$$

Notemos que la divergencia KL no es una métrica porque en general la propiedad de simetría no se cumple. Esto es, en general

$$D_{KL}(p\|q) \neq D_{KL}(q\|p)$$

Sin embargo, podemos usar D_{KL} para definir una divergencia simétrica, conocida

como la divergencia de Jensen-Shannon:

$$D_{JS}(p\|q) := \frac{1}{2}D_{KL}(p\|M) + \frac{1}{2}D_{KL}(q\|M) \tag{1.9}$$

donde $M := \frac{p(\boldsymbol{x})+q(\boldsymbol{x})}{2}$. El siguiente corolario es una consecuencia directa del Teorema 1.

Corolario 2. *$D_{JS}(p\|q) \geq 0$ con igualdad si y solo si $p = q$.*

1.4. Redes generativas adversarias (GANs)

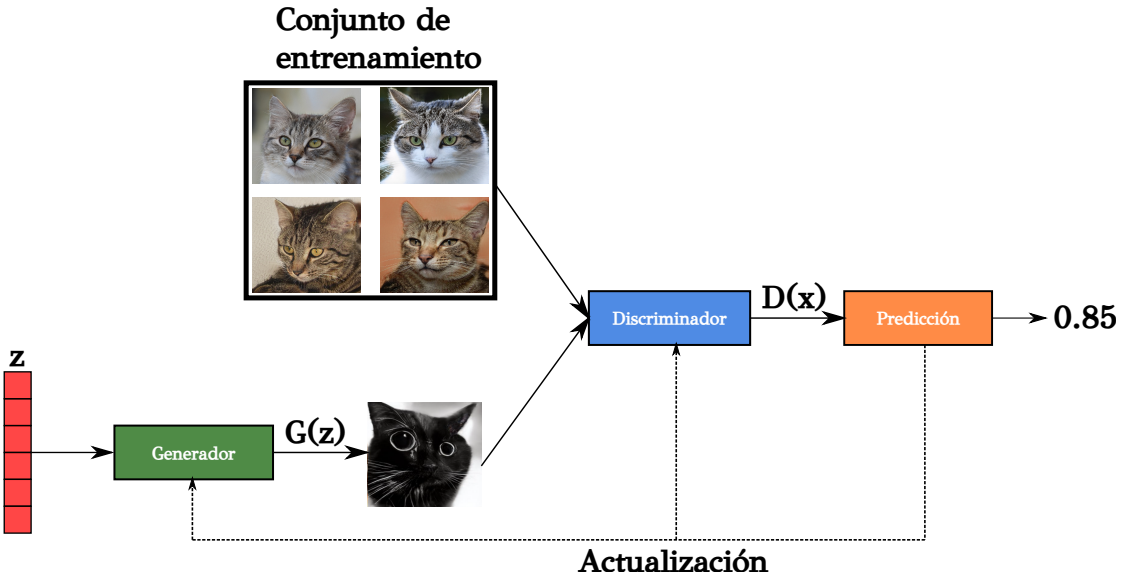


Figura 1.3: Diagrama general de las GANs. A partir de un conjunto de entrenamiento optimizamos una red neuronal que llamamos Discriminador para distinguir entre elementos del conjunto y elementos fuera del conjunto asignando un valor cercano 1 a los primeros y cercano a 0 a los segundos. Luego entrenamos una red neuronal llamada Generador para mapear vectores z con distribución definida a priori hacia el espacio de los datos del conjunto de entrenamiento, de tal forma que estos elementos sean indistinguibles de los elementos reales.

En el contexto descrito en el Apartado 1.2, queremos aproximar la distribución real, desde ahora p_{data} con densidad $p_{data}(\boldsymbol{x})$, a partir de un conjunto de entrenamiento T. Para esto definimos una función diferenciable $G : \mathbb{R}^d \to \mathcal{X}$ representada por una red neuronal con parámetros θ_g, a la que llamamos el *generador*. G toma como argumento un vector $\boldsymbol{Z} \in \mathbb{R}^d$, $d \in \mathbb{N}$, que es una variable aleatoria de distribución $p_{\boldsymbol{z}}$ y densidad $p_{\boldsymbol{z}}(\boldsymbol{z})$, y devuelve un elemento en el espacio de los datos reales $\mathcal{X}$ (de aquí en adelante $\mathcal{X} = \mathbb{R}^n$, $n \in \mathbb{N}$). Notemos que G implícitamente induce una distribución p_g, que es la distribución de las muestras $G(\boldsymbol{z})$ cuando $\boldsymbol{z} \sim p_{\boldsymbol{z}}$.

Asimismo, definimos una segunda función diferenciable $D : \mathcal{X} \to [0, 1]$, a la que llamamos el *discriminador*, que también es una red neuronal con parámetros θ_d. $D(\boldsymbol{x})$ representa la probabilidad de que $\boldsymbol{x}$ provenga de p_{data} en lugar de p_g, esto es, que $\boldsymbol{x}$ provenga del espacio de los datos reales. Queremos entrenar D para maximizar la probabilidad de que D distinga correctamente las muestras reales de las generadas, y al mismo tiempo entrenar G para minimizar la misma probabilidad, o equivalentemente generar elementos que sean indistinguibles de los datos reales.

Para esto enfrentamos a D y G en un juego de minimización-maximización $\min_G \max_D V(D, G)$ bajo la siguiente función de pérdida:

$$V(D, G) = \mathbb{E}_{\boldsymbol{X} \sim p_{data}}[\log D(\boldsymbol{X})] + \mathbb{E}_{\boldsymbol{Z} \sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{Z})))] \tag{1.10}$$

Intuitivamente, primero fijamos G de tal forma que $\max_D V(D, G)$ optimiza D

Algoritmo 1: Entrenamiento de las GANs por gradiente descendiente estocástico en mini-batch. El número de pasos k en los que se entrena el discriminador es un hiperparámetro.

for *número de iteraciones de entrenamiento* **do**

  for k *pasos* **do**

    1. Obtener una muestra de m ejemplos de ruido $\{\boldsymbol{z}^{(1)}, \cdots, \boldsymbol{z}^{(m)}\}$ de la distribución $p_{\boldsymbol{z}}$.
    2. Obtener una muestra de m ejemplos $\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(m)}\}$ del conjunto de entrenamiento T.
    3. Actualizar el discriminador por medio del gradiente ascendente estocástico con respecto a θ_d:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right] \tag{1.11}$$

  end

  1. Obtener una muestra de m ejemplos de ruido $\{\boldsymbol{z}^{(1)}, \cdots, \boldsymbol{z}^{(m)}\}$ de la distribución $p_{\boldsymbol{z}}$.
  2. Actualizar el generador por medio del gradiente descendente estocástico con respecto a θ_g:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \tag{1.12}$$

end

para asignar valores altos a $D(\boldsymbol{X})$, donde $\boldsymbol{X} \sim p_{data}$, y valores bajos a $D(G(\boldsymbol{Z}))$ tal que $1 - D(G(\boldsymbol{Z}))$ también tenga un valor alto. Conversamente, fijando D después de haber sido entrenado hasta cierto nivel de precisión, $\min_G V(D, G)$ optimiza G utilizando la retroalimentación de D para generar muestras tales que $D(G(\boldsymbol{Z}))$ tengo un valor alto. Notemos que el objetivo de D es el de un problema discriminativo como se describió en el Apartado 1.2, en tanto busca maximizar la log-verosimilitud para estimar la probabilidad condicional $P(Y = y|\boldsymbol{X} = \boldsymbol{x})$, donde Y es una varaible binaria que indica si $\boldsymbol{x} \sim p_{data}$ $(y = 1)$ o si $\boldsymbol{x} \sim p_g$ $(y = 0)$.

El procedimiento anterior se expresa formalmente en el Algoritmo 1.

En la siguiente sección demostramos que el Algoritmo 1 optimiza (1.10) hasta el óptimo global $p_g = p_{data}$ en el caso no paramétrico.

1.4.1. Óptimo global

Lema 3. *El segundo término en (1.10) se puede escribir como*

$$\begin{aligned}\mathbb{E}_{\boldsymbol{Z}\sim p_{\boldsymbol{z}}}\Big[\log(1-D(G(\boldsymbol{Z})))\Big] &= \int_{\mathbb{R}^d} p_{\boldsymbol{z}}(\boldsymbol{z})\log\Big(1-D(G(\boldsymbol{z}))\Big)d\boldsymbol{z} \\ &= \int_{\mathbb{R}^n} p_g(\boldsymbol{x})\log\Big(1-D(\boldsymbol{x})\Big)d\boldsymbol{x} \\ &= \mathbb{E}_{\boldsymbol{X}\sim p_g}\Big[\log(1-D(\boldsymbol{X}))\Big]\end{aligned} \tag{1.13}$$

Demostración. Primero notemos que

$$\begin{aligned}\int_{\mathbb{R}^n} p_g(\boldsymbol{x})\log(1-D(\boldsymbol{x}))d\boldsymbol{x} &= \int_{\mathbb{R}^n}\left[\int_{\mathbb{R}^d} p_g(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{z}\right]\log(1-D(\boldsymbol{x}))d\boldsymbol{x} \\ &= \int_{\mathbb{R}^n}\left[\int_{\mathbb{R}^d} p_{\boldsymbol{z}}(\boldsymbol{z})p_g(\boldsymbol{x}\mid\boldsymbol{z})d\boldsymbol{z}\right]\log(1-D(\boldsymbol{x}))d\boldsymbol{x}\end{aligned} \tag{1.14}$$

Por otro lado, observemos que al fijar G obtenemos un mapeo determinista, en tanto el mismo $\boldsymbol{z}$ siempre generará el mismo $\boldsymbol{x}$, de manera que

$$p_g(\boldsymbol{x}\mid\boldsymbol{z}) = \delta(\boldsymbol{x}-G(\boldsymbol{z})) \tag{1.15}$$

Por tanto, por las propiedades de la delta de Dirac

$$\begin{aligned}\int_{\mathbb{R}^n} p_g(\boldsymbol{x})\log(1-D(\boldsymbol{x}))d\boldsymbol{x} &= \int_{\mathbb{R}^n}\left[\int_{\mathbb{R}^d} p_{\boldsymbol{z}}(\boldsymbol{z})\delta(\boldsymbol{x}-G(\boldsymbol{z}))d\boldsymbol{z}\right]\log(1-D(\boldsymbol{x}))d\boldsymbol{x}\\ &= \int_{\mathbb{R}^d}\left[\int_{\mathbb{R}^n}\log(1-D(\boldsymbol{x}))\cdot\delta(\boldsymbol{x}-G(\boldsymbol{z}))d\boldsymbol{x}\right]p_{\boldsymbol{z}}(\boldsymbol{z})d\boldsymbol{z}\\ &= \int_{\mathbb{R}^d}\log(1-D(G(\boldsymbol{z})))p_{\boldsymbol{z}}(\boldsymbol{z})d\boldsymbol{z}\end{aligned} \tag{1.16}$$

□

Proposición 4. *Dado un G fijo, el discriminador óptimo tiene la forma*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})} \tag{1.17}$$

Demostración. Usando el lema anterior podemos expresar (1.10) de la siguiente manera:

$$\begin{aligned}V(G,D) &= \mathbb{E}_{\boldsymbol{X}\sim p_{data}}\Big[\log D(\boldsymbol{X})\Big] + \mathbb{E}_{\boldsymbol{Z}\sim p_{\boldsymbol{z}}}\Big[\log(1-D(G(\boldsymbol{Z})))\Big]\\ &= \mathbb{E}_{\boldsymbol{X}\sim p_{data}}\Big[\log D(\boldsymbol{X})\Big] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\Big[\log(1-D(\boldsymbol{X}))\Big]\\ &= \int_{\mathbb{R}^n} p_{data}(\boldsymbol{x})\log\Big(D(\boldsymbol{x})\Big)d\boldsymbol{x} + \int_{\mathbb{R}^n} p_g(\boldsymbol{x})\log\Big(1-D(\boldsymbol{x})\Big)d\boldsymbol{x}\\ &= \int_{\mathbb{R}^n}\Big(p_{data}(\boldsymbol{x})\log(D(\boldsymbol{x})) + p_g(\boldsymbol{x})\log(1-D(\boldsymbol{x}))\Big)d\boldsymbol{x}\end{aligned} \tag{1.18}$$

Observemos que para cualquier $(a,b)\in\mathbb{R}^2_+$, la función

$$f(y) = a\log(y) + b\log(1-y)$$

alcanza su máximo en $[0,1]$ en $\frac{a}{a+b}$:

$$f'(y) = \frac{a}{y} - \frac{b}{1-y} = 0 \rightarrow y_c = \frac{a}{a+b} \tag{1.19}$$

$$f''(y_c) = -\frac{a}{y_c^2} - \frac{b}{(1-y_c)^2} = -\frac{(a+b)^3}{ab} < 0 \tag{1.20}$$

Por último, el discriminador no necesita estar definido fuera de $\text{supp}(p_{data}) \cup \text{supp}(p_g)$, ya que solo nos interesa distinguir entre elementos que provengan de p_{data} o p_g. □

Sean

$$\overline{p}_{data}(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

$$\overline{p}_g(\boldsymbol{x}) = \frac{p_g(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

Una vez hallado D_G^*, podemos reformular (1.10) como $\min_G C(G)$, donde:

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{\boldsymbol{X}\sim p_{data}}\left[\log D_G^*(\boldsymbol{X})\right] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\left[\log\left(1 - D_G^*(\boldsymbol{X})\right)\right] \\ &= \mathbb{E}_{\boldsymbol{X}\sim p_{data}}\left[\log \frac{p_{data}(\boldsymbol{X})}{p_{data}(\boldsymbol{X}) + p_g(\boldsymbol{X})}\right] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\left[\log \frac{p_g(\boldsymbol{X})}{p_{data}(\boldsymbol{X}) + p_g(\boldsymbol{X})}\right] \\ &= \mathbb{E}_{\boldsymbol{X}\sim p_{data}}\Big[\log \overline{p}_{data}(\boldsymbol{X})\Big] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\Big[\log \overline{p}_g(\boldsymbol{X})\Big] \end{aligned} \tag{1.21}$$

Teorema 5. *El mínimo global de $C(G)$ se alcanza si y solo si $p_g = p_{data}$. En ese punto, $C(G)$ toma el valor $-\log 4$ y $D_G^*(\boldsymbol{x}) = 0{,}5$.*

Demostración. $C(G)$ se puede escribir en términos de (1.9), la divergencia de Jensen-Shannon:

$$\begin{aligned} 2D_{JS}(p_{data}\|p_g) &= D_{KL}\left(p_{data}\middle\|\frac{p_{data}+p_g}{2}\right) + D_{KL}\left(p_g\middle\|\frac{p_{data}+p_g}{2}\right) \\ &= \int_{\mathbb{R}^n} \log\left(2\overline{p}_{data}(\boldsymbol{x})\right) p_{data}(\boldsymbol{x})d\boldsymbol{x} + \int_{\mathbb{R}^n} \log\left(2\overline{p}_g(\boldsymbol{x})\right) p_g(\boldsymbol{x})d\boldsymbol{x} \\ &= \log 4 + \mathbb{E}_{\boldsymbol{X}\sim p_{data}}\left[\log \overline{p}_{data}(\boldsymbol{X})\right] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\left[\log \overline{p}_g(\boldsymbol{X})\right] \end{aligned} \tag{1.22}$$

De manera que

$$C(G) = -\log 4 + 2D_{JS}(p_{data}\|p_g) \tag{1.23}$$

Por el Teorema 2, se sigue que el mínimo global se alcanza cuando $p_{data} = p_g$, donde $C(G)$ toma el valor $-\log 4$. □

1.4.2. Método del subgradiente

La convergencia del Algoritmo 1 depende del método de optimización por subgradientes, que generaliza el concepto de derivada en puntos no diferenciables.

Definición 1.4.1. *Sea $f : U \to \mathbb{R}$ una función convexa definida en un subcojunto convexo U de un espacio localmente convexo V. Un funcional v^* en el espacio dual V^* es el subgradiente en $x_0 \in U$ de f si para todo $x \in U$*

$$f(x) \geq f(x_0) + \langle x - x_0, v^* \rangle \tag{1.24}$$

El conjunto de todos los subgradientes en x_0 se conoce como el subdiferencial en

x_0, y se denota por $\partial f(x_0)$.

La minimización por subgradientes consiste en el siguiente la siguiente proceso [3][19]:

1. Escoger un punto de partida $x^{(0)} \in U$.

2. Actualizar la solución $x^(k)$ de acuerdo al criterio

$$x^{(k)} = x^{(k-1)} - t_k \cdot g^{(k-1)} \tag{1.25}$$

   donde $g^{(k-1)} \in \partial f(x^{(k-1)}$ y $t_k > 0$ es la magnitud del paso en cada iteración. Cuando f es diferenciable en $x^{(k-1)}$ el único posible valor de $g^{(k-1)}$ es $\nabla f(x^{(k-1)})$ y el método se reduce al método del gradiente descendiente, salvo la constante t_k.

3. Ya que el método no garantiza moverse en la dirección de máximo decrecimiento, tomar en cada paso la mejor solución:

$$f\left(x_{mejor}^{(k)}\right) = \min_{i=0,\ldots k} f\left(x^{(i)}\right) \tag{1.26}$$

La ventaja del método del subgradiente es que garantiza la convergencia del modelo dentro de un rango óptimo con un paso constante lo suficientemente pequeño [3][19].

Teorema 6. *Dado un paso constante (esto es, $t_k = h$, la convergencia del método*

está garantizada en un rango de la solución óptima $f^* = \text{ínf}_x f(x)$*:*

$$\lim_{k\to\infty} f\left(x_{mejor}^{(k)}\right) - f^* < \epsilon \tag{1.27}$$

donde $\epsilon = \epsilon(h)$ *es una función del paso* h *que decrece con él.*

1.4.3. Convergencia del algoritmo

Lema 7. *La función* $V(G, D) = U(p_g, D)$ *definida como*

$$U(p_g, D) = \mathbb{E}_{\boldsymbol{X}\sim p_{data}}\Big[\log D(\boldsymbol{x})\Big] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\Big[\log(1 - D(\boldsymbol{X}))\Big] \tag{1.28}$$

es convexa en p_g*.*

Demostración. Sea P_n el conjunto de funciones de densidad de probabilidad en $\mathbb{R}^n$. Es fácil ver que P_n es subconjunto del espacio vectorial de las funciones Lesbegue integrables en $\mathbb{R}^n$, L^1 (L^1, además es localmente convexo). Tomemos $p_1, p_2 \in P_n$ y $\theta \in [0, 1]$, y sea $p(\boldsymbol{x}) = \theta p_1(\boldsymbol{x}) + (1 - \theta)p_2(\boldsymbol{x})$. Entonces

$$\int_{\mathbb{R}^n} p(\boldsymbol{x})d\boldsymbol{x} = \theta \int_{\mathbb{R}^n} p_1(\boldsymbol{x})d\boldsymbol{x} + (1 - \theta) \int_{\mathbb{R}^n} p_2(\boldsymbol{x})d\boldsymbol{x} = 1 + (1 - \theta) = 1 \tag{1.29}$$

y por supuesto $p(\boldsymbol{x}) \geq 0$. Finalmente, P_n es un conjunto convexo.

Ahora, sea

$$J(p_g, D) = \mathbb{E}_{\boldsymbol{X}\sim p_g}\Big[\log(1 - D(\boldsymbol{X}))\Big]$$

Entonces

$$\begin{aligned} J(p, D) &= \int_{\mathbb{R}^n} \Big(\theta p_1(\boldsymbol{x}) + (1-\theta)p_2(\boldsymbol{x})\Big)\log\Big(1 - D(\boldsymbol{x})\Big)d\boldsymbol{x} \\ &= \theta \int_{\mathbb{R}^n} p_1(\boldsymbol{x})\log\Big(1 - D(\boldsymbol{x})\Big)d\boldsymbol{x} + (1-\theta)\int_{\mathbb{R}^n} p_2(\boldsymbol{x})\log\Big(1 - D(\boldsymbol{x})\Big)d\boldsymbol{x} \\ &= \theta J(p_1, D) + (1-\theta)J(p_2, D) \end{aligned} \tag{1.30}$$

de manera que (1.28) es convexa. □

Proposición 8. *Si en cada ciclo D alcanza su óptimo dado un p_g fijo, seguido de una actualización en p_g tal que haya una mejora en el criterio*

$$\mathbb{E}_{\boldsymbol{X}\sim p_{data}}\left[\log D_G^*(\boldsymbol{X})\right] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\left[\log\left(1 - D_G^*(\boldsymbol{X})\right)\right] \tag{1.31}$$

entonces p_g converge a p_{data}.

Demostración. Sea $\{f_\alpha\}_{\alpha\in A}$ una familia de funciones convexas definidas en un dominio convexo C, y consideremos $f = \sup_\alpha f_\alpha$. Dado $0 \leq \theta \leq 1$, $\alpha \in A$ y $x, y \in C$ tenemos

$$\begin{aligned} f_\alpha(\theta x + (1-\theta)y) &\leq \theta f_\alpha(x) + (1-\theta)f_\alpha(y) \\ &\leq \sup_{\alpha'\in A}\left(\theta f_{\alpha'}(x) + (1-\theta)f_{\alpha'}(y)\right) \\ &\leq \sup_{\alpha'\in A}\theta f_{\alpha'}(x) + (1-\theta)\sup_{\alpha'\in A} f_{\alpha'}(y) \\ &= \theta f(x) + (1-\theta)f(y) \end{aligned} \tag{1.32}$$

de manera que

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \tag{1.33}$$

es decir, el supremo f es en si mismo una función convexa.

Ahora, supongamos que para algún $x \in C$

$$\beta = \arg\sup_{\alpha} f_\alpha(x) \tag{1.34}$$

entonces $f(x) = f_\beta(x)$. Tomemos, $g \in \partial f_\beta(x)$, se sigue que para todo $y \in C$

$$f_\beta(y) \geq f_\beta(x) + \langle y - x, g \rangle \tag{1.35}$$

y ya que $f(y) \geq f_\beta(y)$ para todo $y \in D$

$$f(y) \geq f(x) + \langle y - x, g \rangle \tag{1.36}$$

y por tanto $g \in \partial f(x)$, o

$$\partial f_\beta(x) \subset \partial f(x) \tag{1.37}$$

Esto implica que podemos usar los subgradientes de $\partial f_\beta(x)$ en el método descrito en el Apartado 1.4.2 para optimizar f.

Finalmente, notemos que por lo anterior $\sup_D U(p_g, D)$ es una función convexa con mínimo global según el Teorema 5, y en consecuencia con pasos los suficientemente pequeños p_g converge a p_{data}. □

Aunque la proposición anterior garantiza la convergencia en el modelo no paramétrico, Goodfellow et al [9] advierten que en la práctica la familia de distribuciones p_g generadas a través de $G(\boldsymbol{Z}; \theta_g)$ es limitada, y la optimización se hace con

respecto a θ_g en lugar de p_g. Asimismo, expresar G como una red neuronal multicapa introduce varios puntos críticos en el espacio de parámetros. Sin embargo, podemos obviar la falta de garantías teóricas por el excelente desempeño mostrado en la práctica por las redes neuronales.

Por último, estudios posteriores revelaron que el rendimiento es superior al reformular la optimización de manera equivalente como un juego max-max[20], donde primero actualizamos el discriminador para maximizar de manera usual

$$\max_{D} \left(\mathbb{E}_{\boldsymbol{X}\sim p_{data}}[\log D(\boldsymbol{X})] + \mathbb{E}_{\boldsymbol{Z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{Z})))]\right) \tag{1.38}$$

y luego actualizamos el generador para maximizar

$$\max_{G} \left(\mathbb{E}_{\boldsymbol{Z}\sim p_{\boldsymbol{z}}}[\log D(G(\boldsymbol{Z}))]\right) \tag{1.39}$$

Capítulo 2

Experimentos

Para la implementación utilizamos la librería Pytorch, un framework para aprendizaje automático basado en la libería Torch [1], usando como guía en el notebook de la asignación 3 del curso CS231n: Deep Learning for Computer Vision [2]. Las dimensiones se dan en el formato (W, H, C), donde W es el ancho de la imagen, H la altura y C el número de canales. Entonces por ejemplo $5 \times 4 \times 3$ denota una imagen de 5 filas y 4 columnas con 3 canales.

[1] El código está disponible en `https://github.com/josedavid220/gans/blob/thesis/Generative_Adversarial_Networks.ipynb`

[2] Accesible en `http://cs231n.stanford.edu/`

2.1. Base de datos MNIST

Entrenamos nuestra red neuronal adversaria sobre la base de datos MNIST, que contiene un total de 70000 imágenes de dígitos del 0 al 9 escritos a mano en blanco y negro. Todas las imágenes tienen un tamaño estándar de 28×28 pixeles.

Figura 2.1: Ejemplos de la base de datos MNIST[6]

En la siguiente tabla se presenta un resumen de los hiperparámetros y parámetros del optimizador.

Optimizador: ADAM
Tasa de aprendizaje: 0.001
Betas: 0.5, 0.999
k (ver Algoritmo 1): 1

Cuadro 2.1: Hiperparámetros y parámetros del optimizador

2.2. Arquitectura de la red

La arquitectura del discriminador y del generador están inspiradas en las ideas de [17] para aprovechar los avances en visión computacional a través de redes neuronales convolucionales en modelos discriminativos.

Para el discriminador basamos la arquitectura en el tutorial de Keras para clasificación de imágenes de MNIST [1], que es capaz de clasificar con 99 % de precisión.

Conv2D: 32 filtros, kernel 5×5, stride 1
Leaky ReLU($\alpha = 0{,}01$)
Max Pool 2×2, stride 2
Conv2D: 64 filtros, kernel 5×5, stride 1
Leaky ReLU($\alpha = 0{,}01$)
Max Pool 2×2, stride 2
Aplanamiento (Convertir matriz a vector)
Fully Connected, salida de tamaño $4 \cdot 4 \cdot 64$
Leaky ReLU($\alpha = 0{,}01$)
Fully Connected, salida 1

Cuadro 2.2: Arquitectura del discriminador

En cambio, para el generador usamos la arquitectura sugerida en [5]. Las muestras de ruido son vectores 1×96 donde cada coordenada es una muestra aleatoria

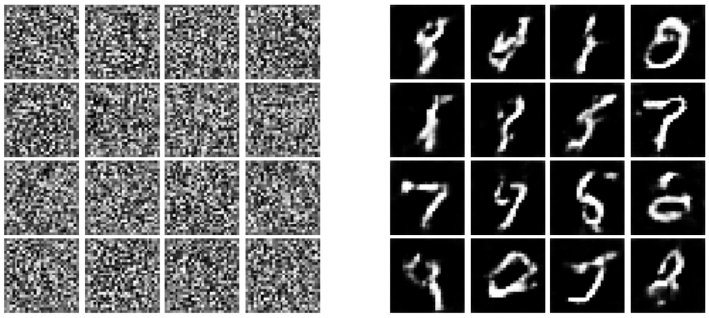
de la distribución uniforme entre -1 y 1.

Fully connected, salida de tamaño 1024
ReLU
BatchNorm1D (1024 canales)
Fully connected, salida de tamaño $7 \cdot 7 \cdot 128$
ReLU
BatchNorm1D (1024 canales)
Redimensionar vector a $7 \times 7 \times 128$
ConvTranspose2D: 64 filtros 4×4, stride 2, padding 1
ReLU
BatchNorm2D (64 canales)
ConvTranspose2D: 1 filtro 4×4, stride 2, padding 1
TanH
Aplanamiento (Convertir matriz a vector)

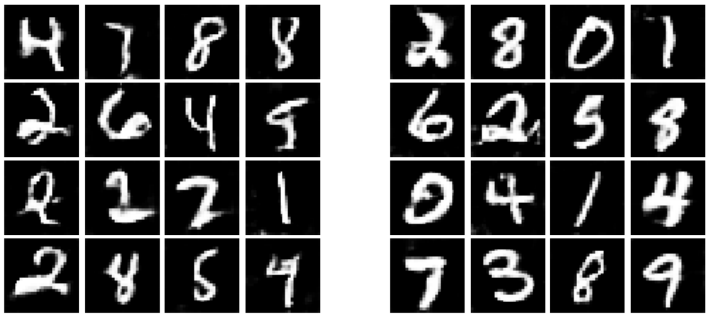
Cuadro 2.3: Arquitectura del generador

2.3. Resultados

Presentamos algunas imágenes generadas por G para diferentes iteraciones tomadas aleatoriamente. Claramente vemos que el generador es capaz de crear ejemplos cada vez más similares a los del conjunto original.



(a) Iteración 0 (b) Iteración 250



(c) Iteración 1500 (d) Iteración 2500

Figura 2.2: Imágenes generadas para diferentes iteraciones. Inicialmente en **(a)** las imágenes generadas son ruido puro, pero a medida que entrenamos la red **(b)**-**(d)** incrementamos la calidad de las imágenes al punto de ser casi indistinguibles de las imágenes reales.

Asimismo, observemos que al evaluar las imágenes generadas en el discriminador (esto es, $D(G(\boldsymbol{z}))$) en cada una de las iteraciones del modelo obtenemos valores altos y bajos de probabilidad en las primeras 500 iteraciones, pero luego hay una estabilización entre 0.45 y 0.55. Esto concuerda con las predicciones teóricas del Teorema 5, que establece que la probabilidad debería converger a 0.5.

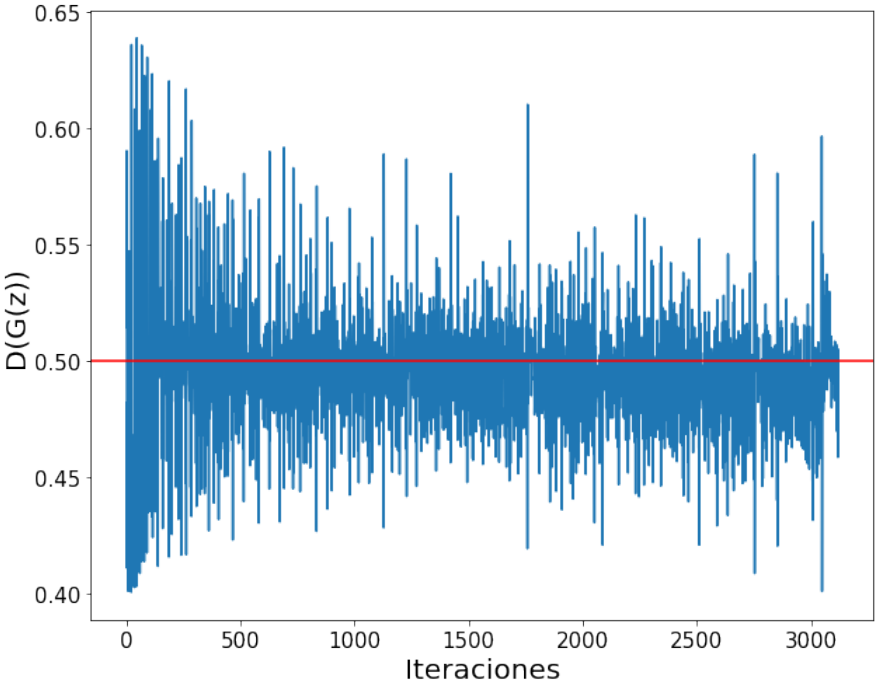


Figura 2.3: Probabilidad de pertenecer al conjunto de los datos reales para distintas iteraciones del modelo evaluada en las imágenes generadas por G.

Capítulo 3

Conclusiones

Las GANs son una manera efectiva de generar nuevos ejemplos de un conjunto de datos. Su potencial resulta de aprovechar los avances en modelos discriminativos para entrenar en un sistema adversario dos redes neuronales, y cuya garantía teórica deriva de adoptar un enfoque probabilístico sobre un conjunto de entrenamiento y luego generar distribuciones que se aproximen iterativamente a la distribución real de los datos. La convergencia está asegurada por la existencia de un mínimo global en la divergencia de Jensen-Shannon y la garantía de convergencia del método del subgradiente, que en la práctica significa que los elementos generados son muestras de la misma distribución de los datos reales.

La aplicación del modelo a través de dos redes neuronales convolucionales sobre la base de datos MNIST permitió obtener imágenes de dígitos que se asemejan en gran medida a los datos originales, confirmando la efectividad de la teoría. Adicionalmente, al evaluar los elementos generados en el discriminador obtenemos

resultados de probabilidad cercanos a 0.5 después de 500 iteraciones, confirmando de nuevo las predicciones teóricas.

Por otro lado, destacamos que algunas de las limitaciones del modelo son que no existe un criterio de parada claro para el entrenamiento, la evaluación de los elementos generados es complicada debido a la naturaleza subjetiva del criterio de ïndistinguible de los datos reales", y en contraste con métodos como el de máxima verosimilitud al final no tenemos una expresión analítica para la densidad real.

Bibliografía

[1] Simple mnist convnet. https://keras.io/examples/vision/mnist_convnet/. Accessed: 14-12-2022.

[2] S. Boyd, S. Boyd, L. Vandenberghe, and C. U. Press. *Convex Optimization*. Number pt. 1 in Berichte über verteilte messysteme. Cambridge University Press, 2004.

[3] S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. https://web.stanford.edu/class/ee392o/subgrad_method.pdf, 2003. [Online; accessed 18-November-2022].

[4] A. Brøndsted and R. T. Rockafellar. On the subdifferentiability of convex functions. *Proceedings of the American Mathematical Society*, 16(4):605–611, 1965.

[5] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets, 2016.

[6] W. Commons. Mnistexamples, 2017. File: MnistExamples.jpg.

[7] M. Elgendy. *Deep Learning for Vision Systems*. Manning Publications, 2020.

[8] D. Foster. *Generative deep learning : teaching machines to paint, write, compose, and play / David Foster.* 2019.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[10] V. Guruswam. Lecture 3: Kl-divergence and connections. `https://www.cs.cmu.edu/~venkatg/teaching/ITCS-spr2013/notes/lect-jan22.pdf`, 2013. [Online; accessed 7-November-2022].

[11] J. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Grundlehren Text Editions. Springer Berlin Heidelberg, 2004.

[12] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan, 2019.

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013.

[14] D. Kroese, Z. Botev, T. Taimre, and R. Vaisman. *Data Science and Machine Learning: Mathematical and Statistical Methods*. Chapman & Hall/CRC machine learning & pattern recognition. CRC Press, Boca Raton, 2019.

[15] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning. The MIT Press, 1 edition, 2012.

[16] I. Panageas. Lecture 7. introduction to min-max optimization. `https://panageas.github.io/_pages/GANs_scribe.pdf`, 2021. [Online; accessed 14-November-2022].

[17] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[18] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[19] R. Tibshirani. Lecture 7: Convex ptimization. `https://www.stat.cmu.edu/~ryantibs/convexopt-S15/scribes/07-sg-method-scribed.pdf`, 2015. [Online; accessed 18-November-2022].

[20] Y. Wang. A mathematical introduction to generative adversarial nets (gan), 2020.