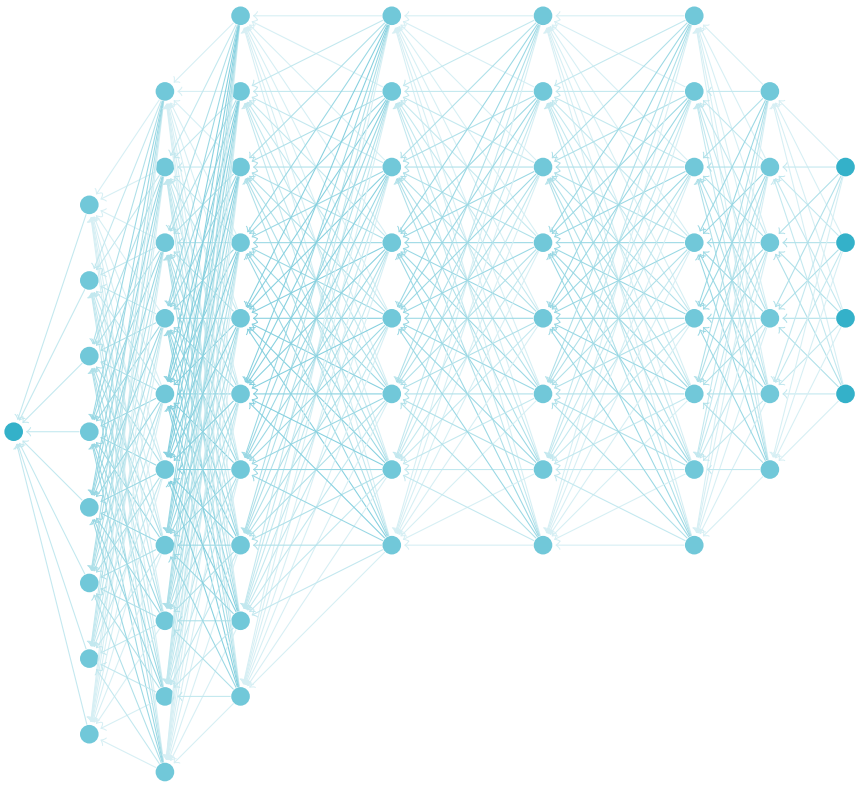
Teoría Matemática de Redes Neuronales

Notas de Clase



Seminario de Aplicaciones Matemáticas 2021-1, 2021-2,
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

https://github.com/danielrole/Notas-TMRN

La siguiente recopilación de notas están basadas en los Seminarios de Matemáticas Aplicadas 2021-1 y 2021-2, impartidos por el Dr. Miguel Arturo Ballesteros Montero.

First release, August 2020

Índice general

Prólogo

El presente documento es el resultado del esfuerzo e interés de estudiantes de semestres superiores, con el objetivo de aprender y facilitar la enseñanza para las generaciones próximas, al contar con un texto riguroso en el sentido matemático, que sirva como material de referencia, pero simple para adentrarse en el tema.

Los algoritmos basados en redes neuronales, subconjunto del aprendizaje automatizado, no son idealmente un punto de partida para introducirse en este mundo, sin embargo para estudiantes experimentados en el área del análisis funcional, puede resultar intuitivo, por lo cual se le facilitará la compresión de los conceptos presentados en el documento.

Por lo anteriormente explicado partiremos del supuesto que el lector posee conocimientos de análisis funcional, con el objetivo de no extendernos demasiado explicando estos conceptos, sin embargo se incluirá un anexo, que explique con mas detalle los mismos.

El enfoque principal del documento es el rigor matemático, el cual es fundamental para la comprensión y desarrollo de algoritmos implementados en librerías de lenguajes de programación como R, o Python.

Asimismo es de nuestro interés dar una introducción a su implementación en estos lenguajes (Principalmente Python), sin embargo, debido a como evolucionan estas librerías con sus respectivas actualizaciones, lo mas recomendable es siempre revisar su documentación. Al mismo tiempo prevenimos que el documento se vuelva obsoleto ya que la teoría se mantiene invariante.

Agradecimientos

Esto proyecto esta a cargo de Daniel Robles Leong y Mario Raúl Guzmán Gutiérrez, quienes agradecen el interés y esfuerzo, de todos los involucrados en el proyecto:

Ariadna Amayrani Ferrari Pardiño - Jorge Iván Reyes Hernández - Enrique Jiménez Villarreal - Noel De la Cruz Velasquez - Ángel Edmundo Hernández Martínez - Axel Francisco Leon Paloma - César Miguel Aguirre Calzadilla - Rodrigo Emanuel Albarrán Ruiz

Inteligencia Artificial a traves del tiempo:

- **1943-1955:** La gestación de la inteligencia artificial.
    - Modelo de neuronas artificiales de Warren McCulloch y Walter Pitts en 1943.
    - Turing dio conferencias sobre inteligencia artificial desde 1947.
- **1956:** El nacimiento de la inteligencia artificial.

- John MaCarthy convenció a Marvin Minsky, Claude Shannon y Nathaniel Rochester para que lo ayudaran a reunir investigadores estadounidenses interesados en la teoría de autómatas, las redes neuronales y el estudio de la inteligencia en un taller de dos meses en Dartmouth.

- **1952-1969:** Entusiasmo inicial, grandes exprectativas.
  - Primeros solucionadores y jugadores autómatas, algunos capaces de hacer demostraciones matemáticas.
  - John McCarthy se refirió a este periodo como el "¡Mira mama, sin manos!".
  - Creación de LISP.
  - *Perceptron* de Frank Rosenblatt en 1958.
  - Adalines (*Adaptive Linear Neuron*) de Bernie Widrow y Marcian Hoff en 1960.

- **1966-1973:** Una dosis de realidad.
  - Prueba y error: explosión combinatoria.
  - Falta de recursos computacionales.

- **1969-1979:** Sistemas basados en el conocimiento ¿La clave del poder?
  - Algoritmos que utilizan conocimientos específicos de dominio en lugar de solucionadores de propósito general.
  - Sistemas expertos para diagnóstico médico.
  - Incorporación de incertidumbre.

- **1980-presente:** La IA se convierte en industria.
  - Optimización de la logística.
  - Auge repentino, pero solo unos pocos proyectos estuvieron a la altura de las expectativas.
  - Invierno IA.

- **1986-presente:** El regreso de las redes neuronales.
  - El algoritmo de retropropagación (Backpropagation) para entrenar redes neuronales se reinventó en "Learning representation by back-propagating errors."por David E. Rumelhart, Geoffrey E. Hinton y Ronald J. Williams.

- **1987-presente:** La IA adopta el método científico.
  - Modelos ocultos de Markov.
  - Redes Bayesianas.

- **1995-presente.**
  - Internet impulsa el desarrollo de agentes inteligentes, por ejemplo:
    - Chatbots.
    - Sistemas de recomendación.
    - Recomendaciones de amistades.
  - Acceso a recursos de computación a suficiente velocidad.

- La era del *Big Data:* Gran cantidad de *label training data*, por ejemplo:
  - Diccionarios.
  - Wordnets.
  - Wikipedia.
  - Google.
- Los fundadores de la IA descontentos con su estado actual:
  - La IA debería volver a sus raíces de luchar por, en palabras de Herbert Simon, "Maquinas que piensan, que aprenden y crean".
- La IA en la cultura popular.
  - Lucha contra el spam.
  - Reconocimiento de voz: Siri, Alexa, Cortana.
  - Reconocimiento facial: Facebook, Apple Photos, Google Photos.
  - *Deep Blue vs Garry Kasparov*.
  - Planificación y programación autónoma: Mars rover de la NASA.
  - Vehículos robóticos: EL automóvil autónomo de Tesla.
  - Traducción automática: Traductor de Google.

- **Hoy** Tú empiezas a leer estas notas.

1 | Aprendizaje Automatizado

1.1. ¿Que es el Aprender?

> "El Aprendizaje Automatizado brinda a las computadoras la capacidad de aprender sin estar programadas explícitamente."
>
> Arthur L. Samuel, 1959

Un aspecto importante al definir la inteligencia es la capacidad de aprender, por ejemplo en los seres vivos esta directamente relacionado con nuestro instinto de supervivencia al poder identificar situaciones de peligro sin la necesidad de experimentarlas, a base de la observación o la comunicación. De este modo el aprendizaje a diferencia de la memorización, es la capacidad de inferir resultados a través de la experiencia previa.

Un ejemplo de esto es la inducción lógica en la cual, en base a un resultado trivial de un caso particular, esperamos inferir una regla general.

Entrada	**Implicación**	**Ejemplo**
CASO		El cielo estaba nublado
RESULTADO		llovió
	REGLA	Si el cielo está nublado, lloverá

De esta forma el objetivo de los algoritmos de aprendizaje es identificar patrones sobre las variables (Categóricas o Numéricas), que nos permitan inferir sobre información ajena a la experiencia previa, para predecir un resultado.En términos prácticos un algoritmo óptimo debería ser capaz de obtener generalizaciones a partir de casos individuales, a lo que se le conoce como razonamiento inductivo o inferencia inductiva.

Un ejemplo de estos algoritmos, sería idear una manera de identificar un correo spam en una bandeja de e-mail, como cualquier internauta con suficiente experiencia sabrá correos que contienen frases como "Usted ha ganado un premio" no son de confianza.

De esta manera podemos clasificar cada correo de esta manera:

$$F(x) = \begin{cases} 1 & \text{Contiene frases sospechosas} \\ 0 & \text{No contiene frases sospechosas} \end{cases}$$

Donde x es un elemento del conjunto de correos en una bandeja de entrada, en donde, podemos hacer inferencia sobre las siguientes casos:

Usted ha ganado **$10,000** haga click en el siguiente enlace	1
Usted ha ganado **un iPhone** haga click en el siguiente enlace	1
Usted ha ganado **un viaje** haga click en el siguiente enlace	1

Sobre la cual podemos inferir que cualquier correo que ofrezca **un premio**, dando un click en un enlace, ¡es un spam!.

Es importante mencionar que estas inferencias pueden llegar a ser erróneas, el factor humano es importante para identificar si un algoritmo es válido o no y existen diversos métodos que exploraremos mas adelante.

1.2. ¿Cuándo se requiere el Aprendizaje Automatizado?

Es importante tener claros las situaciones en las cuales se requiere de estos modelos, estos son la complejidad de la tarea y la necesidad de adaptabilidad.

Complejas de Programar

- Tareas Humanas:

  Existen numerosas tareas que desarrollamos diariamente sin darnos cuenta de la complejidad que representen, sin embargo aun tomando conciencia de ellas no existe una manera sencilla de programarlas.

  Algunos ejemplos serían: Manejar un auto, Reconocimiento del Lenguaje y Procesamiento de imágenes, estos algoritmos que aprenden de la experiencia, comúnmente llamados *"state of the art"*, son bastante efectivos si se tienen datos suficientes.

- Tareas Fuera de nuestras capacidades:

  Otra clase de tareas son aquellas que sobrepasan nuestras capacidades como humanos, algunos ejemplos son: patrones astronómicos, predicción del clima, análisis de millones de perfiles financieros, algoritmos de búsqueda en la web, etc.

  Con miles o millones de datos existentes y bases de datos actualizándose diariamente, se vuelve imposible para la capacidad humana trabajar con tal cantidad de información, es entonces cuando la capacidad de almacenar y procesar información se vuelve indispensable.

Adaptabilidad Un aspecto limitante a la hora de programar es la rigidez del proceso, un código una vez instalado se mantiene sin cambios. Esto representa un problema para tareas que varían de un usuario a otro, los algoritmos de aprendizaje automatizado permiten cambiar aspectos de su programación para ser óptimos, sin necesidad de intervención humana.

Algunos ejemplos son, reconocimiento de escritura a mano, que se debe adaptar de acuerdo a cada usuario, detección del spam y reconocimiento de voz.

1.3. Aprendizaje Supervisado

El aprendizaje supervisado se define conforme al uso realizado para entrenar algoritmos capaces de clasificar datos o realizar predicciones. Los datos de entrada alimentan el modelo y a su vez este ajusta los pesos de los datos de modo que el modelo se ajuste adecuadamente, el cual se produce como parte del proceso de validación cruzada.

El aprendizaje supervisado usa datos de entrenamiento, para poder educar al modelo de modo que se obtenga el resultado deseado. Dichos datos de entrenamiento, incluyen datos de entrada y datos de salida (correctos), el algoritmo mide la precisión de acuerdo a una función de perdida, ajustándose hasta que el error se minimice lo suficiente.

El aprendizaje supervisado se puede separar en dos tipos respecto a los desafíos de Data mining, clasificación y regresión:

Clasificación Se utiliza un algoritmo para asignar con precisión los datos de prueba en categorías específicas.Reconociendo entidades específicas dentro del conjunto de datos y realizando algunas conclusiones sobre cómo se definirse esas entidades. Los algoritmos de clasificación más comunes son los clasificadores lineales, las máquinas de vectores de apoyo (SVM), los árboles de decisión, los vecinos más cercanos y los bosques aleatorios, etc.

Regresión Se usa comúnmente para realizar proyecciones, las mas comunes suelen ser regresión lineal, regresión logística y regresión polinomios.

Distintos algoritmos y técnicas computacionales son usadas en procesos supervisados de machine learning, muchos de estos métodos son usados en Python o en R:

Redes Neuronales Son usadas principalmente en algoritmos de aprendizaje profundo, procesando datos de entrenamiento imitando la inter-conexión del cerebro humano a través de capas y nodos. Cada nodo se compone de entradas, pesos, un sesgo de umbral y una salida. Si el valor de salida supera un umbral determinado, se activa el nodo, pasando los datos a la siguiente capa de la red. Las redes neuronales aprenden esta función de mapeo a través del aprendizaje supervisado y ajustándose en base a la función de pérdida a través del proceso de descenso de gradiente. Cuando la función de coste es igual o cercana a cero, se puede confiar en la precisión del modelo.

Clasificador Bayesian Naive Bayes, es un enfoque de clasificación que adopta el principio de independencia condicional de las clases del Teorema de Bayes. Este indica que la presencia de una característica no influye en la presencia de otra en la probabilidad de un resultado determinado. Existen tres tipos de clasificadores Naïve Bayes: Naïve Bayes multinomial, Naïve Bayes Bernoulli y Naïve Bayes gaussiano. Esta técnica se utiliza principalmente en la clasificación de textos, la identificación de spam y los sistemas de recomendación.

Regresión Lineal y Logistica Esta tecnica se suele usar para identificar las relaciones de una variable dependiente y una o mas variables independientes, se suele usar para realizar predicciones sobre resultados futuros. Mientras que la regresión lineal se aprovecha cuando las variables dependientes son continuas, la regresión logística se selecciona cuando la variable dependiente es categórica, lo que significa que tienen salidas binarias. Aunque ambos modelos de regresión buscan comprender las relaciones entre las entradas de datos, la regresión logística se utiliza principalmente para resolver problemas de clasificación binaria, como la identificación del spam.

Support Vector Machine (SVM) Este es un modelo de aprendizaje supervisado que se utiliza tanto para la clasificación de datos como para la regresión. Normalmente se utiliza para problemas de clasificación, construyendo un hiperplano donde la distancia entre dos clases de puntos de datos es máxima. Este hiperplano se conoce como la frontera de decisión, que separa las clases de puntos de datos (por ejemplo, naranjas frente a manzanas) a ambos lados del plano.

K-nearest neighbor K-nearest neighbor es un algoritmo no paramétrico que clasifica los puntos de datos en función de su proximidad y asociación con otros datos disponibles. Este algoritmo asume que se pueden

encontrar puntos de datos similares cerca unos de otros. De modo que calcula la distancia entre los puntos de datos, normalmente a través de la distancia euclidiana, y luego asigna una categoría basada en la categoría más frecuente o en la media. Este algoritmo suele utilizar para los motores de recomendación y el reconocimiento de imágenes.

Algunos ejemplos de aprendizaje supervisado que pueden impulsar su uso en aplicaciones comerciales son: el análisis predictivo, el reconocimiento de imágenes y objetos, análisis sobre la opinión de los clientes, y en la detección de spam.

Actualmente existen algunos desafíos para la creación de modelos con aprendizaje supervisado, tales como: la necesidad de tener un cierto nivel de experiencia para estructurarlos con precisión, el tiempo de creación, el margen de error en la creación de los conjuntos de datos que dificultan el correcto aprendizaje de los modelos, y que el aprendizaje supervisado por si solo no clasifique los datos.

1.4. Aprendizaje No Supervisado

A diferencia del aprendizaje supervizado, el no supervizado usa datos sin etiquetar. A partir de ellos encuentra patrones que apoyan a la resolución de problemas de asociación y agrupación, sin la necesidad de intervención humana. Gracias a la capacidad de descubrir similitudes y diferencias en la información, el aprendizaje no supervizado se convierte en una gran herramienta para el análisis de datos exploratorios, estrategias de venta cruzada, y reconocimiento de imágenes. Los algoritmos de agrupación más comunes son: los modelos jerárquicos, de k-medias y de mezcla gaussiana.

Los modelos de aprendizaje no supervisado se ocupan principalmente en tres tareas: el agrupamiento, la asociación y la reducción de dimensional.

El aprendizaje no supervizado proporciona una ruta exploratoria para ver datos, lo cual permite a las empresas identificar patrones en grandes volúmenes de datos más rápidamente. Algunos ***ejemplos*** del uso del aprendizaje no supervizado son: secciones de noticias (categorizar artículos), visión por computadora (algoritmos para la percepción visual, reconocimiento de objetos), imágenes médicas (detección, segmentación y clasificación de imágenes para diagnostico más rápido y preciso), detección de anomalías (analizar grandes cantidades de datos y descubrir puntos de datos atípicos).

Los retos a superar en la implementación de los modelos de aprendizaje no supervizado, ya que no hay intervención humana tienen que ver con la complejidad computacional debido a la gran cantidad de datos de entrenamiento, el largo tiempo de formación, un mayor margen de error, la necesidad de intervenir para validar variables de salida y no tener un claro entendimiento sobre la agrupación de los datos en la base.

1.5. Supervisado VS No Supervisado

Como el aprendizaje involucra una interacción entre el algoritmo y el entorno, uno puede dividir las tareas de acuerdo a la naturaleza de la interacción y a traves de ello elegir un método que mejor se adapte a los resultados esperados.

Retomando el ejemplo de identificar spam una forma de ilustrar las diferencias entre los algoritmos supervisados de los no supervisados, es la diferencia entre clasificar entre spam y no spam, a detectar correos inusuales. Son dos aproximaciones a un mismo problema, en donde cada procedimiento representa una aproximación diferente.

De forma mas abstracta, viendo al aprendizaje como el proceso de ”Utilizar la experiencia para ganar perspica-

cia", el aprendizaje supervisado representa un escenario donde las datos utilizados para entrenar el algoritmo contienen información a través de la cual podemos "validar / supervisar " la precisión de nuestro modelo, en el caso del aprendizaje no supervisado, no contamos con este método de validación.

Para modelos de aprendizaje supervisado dispondremos de métodos como *cross validation* en el cual podemos dividimos nuestros datos entre un conjunto de prueba y conjunto de entrenamiento, a través de los cuales podremos establecer si nuestras inferencias son independientes del conjunto de entrenamiento que tomemos. esto al validar el modelo con datos con los que no fue entrenado.

Por otro lado para modelos no supervisados, existen métodos como *Dimensional Reduction* para descartar variables que no favorecen a la clasificación o *Empirical Risk Minimization* para establecer los límites de la clasificación.

Cabe mencionar que existe un tipo de aprendizaje intermedio llamado **Aprendizaje por Refuerzo**

2 | Neuronas y Redes Neuronales

En este curso estudiaremos objetos matemáticos llamados redes neuronales. Estos son elementos centrales en Inteligencia Artificial (IA). El principal objetivo al momento de usar Inteligencia artificial es que las computadoras puedan razonar de manera similar al humano. Una manera de lograr reflejar dicho comportamiento es mediante redes neuronales, dichas redes permiten a los programas computacionales reconocer patrones y resolver problemas de clasificación y regresión.

2.1. Desarrollo Histórico

En un contexto histórico, las computadoras digitales han evolucionado del modelo de Von Neumann, y opera mediante una ejecución de instrucciones explicitas por medio de un acceso a la memoria con un determinado numero de procesadores. Mientras que por otro lado, el origen de las redes neuronales se basa en los esfuerzos de modelar la información procesada en sistemas biológicos. A diferencia del modelo de von Neuman, las redes neuronales no requieren una arquitectura en la que la memoria este separada del procesador.

La base preliminar para desarrollar la teoría de redes neuronales fue propuesta por Alexandeer Bain y William James al modelar el comportamiento de la neuronas humanas. Posteriormente a mediados del siglo XX, McCulloch y Pitts crearon un modelo computacional para las redes neuronales basados en algoritmos matemáticos. El llamado modelo Threshold logic, sento las bases en la investigación y el desarrollo de redes neuronales. Tiempo después Rosenblatt, desarrollo la idea del perceptron en su trabajo, un algoritmo de aprendizaje basado en dos capas de aprendizaje y dicho desarrollo continuo de manera consecutiva.

2.2. Neuronas Artificiales

Perceptrón Históricamente, la primera neurona en el contexto del cerebro fue el perceptrón propuesto por McCulloch y Pitts en 1943

$$\widehat{x} \to \chi_{\mathbb{R}^+}\left(\sum w_i x_i - \theta\right),$$

en donde, $\widehat{x} \in \mathbb{R}^d$ con $d \in \mathbb{N}$, para todo i, los *pesos* $w_i \in \mathbb{R}$, el umbral o sesgo $\theta \in \mathbb{R}$, $\mathbb{R}^+ = [0, \infty]$ y la función de activación $\chi_{\mathbb{R}^+}$ se define, para cada conjunto A, como:

$$\chi_A(y) = \begin{cases} 1 & \text{si } y \in A, \\ 0 & \text{si } y \notin A. \end{cases}$$

En otras palabras, un perceptrón es un tipo de neurona artificial que toma varias entradas reales $x_1, x_2, \cdots, x_n$ y produce una única salida binaria y. Para calcular la salida se introducen pesos $w_1, w_2, \cdots, w_n$, números reales que expresan la importancia de cada entrada para la salida. La salida de la neurona es 0 ó 1, y esta determinada

si la suma ponderada con el sesgo b, $\sum_j w_j x_j + b$ es menor o mayor que 0. El sesgo expresa que tan fácil el perceptrón tiene como salida un 1, la función:

$$\widehat{x} \to \sum x_i w_i - \theta$$

es una transformación afín (una transformación lineal compuesta con una traslación) y la función $\chi_{\mathbb{R}^+}$ es una función no lineal.

Si reescribimos la suma con el producto punto, entonces $w \cdot x = \sum_j w_j x_j$ (donde w y x son los vectores de las entradas y los pesos, respectivamente), podríamos expresar esto de forma algebraica del siguiente modo,

$$y = \begin{cases} 0 \quad si \quad w \cdot x + b \leq c \\ \\ 1 \quad si \quad w \cdot x + b > c \end{cases}$$

Una implementación en Python de la función perceptrón se muestra a continuación;

```
#Libreria requerida
import numpy as np

#Definimos nuestra funcion caracteristica
def char(z):
    if z < 0:
        return 0
    else:
        return 1

#Definimos nuestro perceptron
def perceptron(a,x,b):
    x = np.matmul(a, x) + b

    return [char(i) for i in x]
```

Neurona Sigmoidal Las neuronas sigmoidales son bastante similares a los perceptrones, pero la modificamos de tal modo que pequeños cambios en su peso y su sesgo causa sólo pequeños cambios en la salida.

Al igual que el perceptrón, las neuronas sigmoidales tiene entradas $x_1, x_2, \cdots, x_n$, pero en vez de que sólo tomen valores 0 ó 1, estas entradas pueden tomar cualquier valor entre 0 y 1. De igual modo que un perceptrón, las neuronas sigmoidales tienen pesos asociados a cada entrada $w_1, w_2, \cdots, w_n$ y un sesgo general, b. Pero su salida no es 0 o 1, en cambio es $\sigma(wx + b)$, donde σ se llama función sigmoidal y un ejemplo de este tipo de funciones haciéndo un poco más explicita, la salida de la neurona sigmoidal con entradas $x_1, x_2, \cdots, x_n$, pesos $w_1, w_2, \cdots, w_n$ y sesgo b es:

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + exp(-\sum_j w_j x_j - b)}$$

Para poder entender las similaridades entre los perceptrones y las neuronas sigmoidales, supongamos que $z = w \cdot x + b$ es un número positivo bastante grande, entonces $e^z \approx 0$ y por lo tanto $\sigma(z) \approx 1$. De igual modo, si

$z = w \cdot x + b$ es un número negativo bastante grande, entonces $e^z \to \infty$ y de ahí que $\sigma(z) \approx 0$. Es decir,cuando z se aproxima a ∞ entonces la salida es aproximadamente 1, y cuando z se aproxima a $-\infty$ se aproxima a 0. De aquí podemos concluir que el comportamiento de las neuronas sigmoidales en sus valores límites es bastante parecido a la de los perceptrones.

Una implementación en Python de la función sigmoidal se muestra a continuación;

```
#Libreria requerida
import numpy as np

#Definimos nuestra funcion signodal
def sigma(z):
    return 1/(1+(np.exp(-z)))

#Definimos nuestra neurona sigmoidal
def sigmund_neuron(a,x,b):
    x = np.matmul(a, x) + b

    return [sigma(i) for i in x]
```

Neurona Artificial General Podemos definir de manera general a las neuronas artificiales de la siguiente forma, donde la función ρ es la función de activación y la transformación afín T contiene a los pesos y al umbral (de forma abstracta), una red neuronal es una composición de neuronas. En nuestro caso consideramos una única función de activación para todas las neuronas de la red.

Definición 2.1 (Neurona)

Sean $T : \mathbb{R}^m \to \mathbb{R}^n$ una transformación afín y $\rho : \mathbb{R} \to \mathbb{R}$ una función no lineal. Definimos una neurona como la composición:

$$\rho(T(\widehat{x})) := (\rho(T(\widehat{x})_1), \cdots, \rho(T(\widehat{x})_n))^t$$

en donde:

$$\widehat{x} = (x_1, \cdots, x_m)^t \in \mathbb{R}^m \qquad T(\widehat{x}) = (T(\widehat{x})_1, \cdots, T(\widehat{x})_n)^t \in \mathbb{R}^n$$

2.3. Redes Neuronales Artificiales

También conocidas como redes neuronales artificiales (ANN) o redes neuronales simuladas (SNN), son un subconjunto del aprendizaje automático y están en el corazón de los algoritmos de aprendizaje profundo. La estructura y nombre de las redes neuronales están inspirados por la imitación en que las neuronas biológicas se transmiten entre sí para reflejar el comportamiento del cerebro humano, Estas permiten que los programas informáticos reconozcan patrones y resuelvan problemas en el campo dela inteligencia artificial.

Las ANN están compuestas por capas de nodos, que contienen una capa de entreda, una de salida y una o más capas ocultas. Cada nodo, o neurona artificial, se conecta a otro y tiene un peso y un umbral asociados. Si la salida de cualquier nodo individual está por encima del valor de umbral especificado, entonces se activa y envía

datos a la siguiente capa. De lo contrario, no se transmiten datos a la siguiente capa de la red, a su vez esto da como resultado que la salida de un nodo se convierta en la entrada del siguiente. A este proceso de pasar datos de una capa a la siguiente define esta red neuronal como una red de alimentación. Estas se entran a través de algoritmos como **Backprpagation** el cual veremos más adelante.

Definición 2.2 (Red Neuronal)

Una red neuronal de L niveles consiste de L transformaciones afines, $T_1, T_2, ..., T_L$ y una función de activación $\rho : \mathbb{R} \to \mathbb{R}$. Definimos la red neuronal como la composición

$$F = T_L \circ \rho \circ T_{L-1} \circ \rho \circ ... \circ \rho \circ T_1$$

en donde estamos identificando a la función ρ con funciones $\rho : \mathbb{R}^n \to \mathbb{R}^n$ de tal forma que

$$\rho \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} := \begin{pmatrix} \rho(Y_1) \\ \vdots \\ \rho(Y_n) \end{pmatrix}.$$

(Esto es un abuso de notación que siempre se usa)

A la función F se le nombra un "Multilayer Perceptron"(MLP) de dimensión d, L niveles y función de activación ρ; donde el dominio de T_1 es $\mathbb{R}^d$. Las redes neuronales feedforward, o perceptrones multicapa (MLP), se componen de una capa de entrada, una o más capas ocultas y una capa de salida. Si bien estas redes neuronales también se conocen comúnmente como "Multilayer Perceptron"(MLP), es importante tener en cuenta que en realidad están compuestas por neuronas sigmoides, no perceptrones, ya que la mayoría de los problemas del mundo real no son lineales.

De forma general, se considera el siguiente modelo:

Tranformación afín Sea $T : \mathbf{R}^m \longrightarrow \mathbf{R}^n$ una transformación afín. la cual se define como una transformación lineal compuesta con una traslación: $T(\theta) = Ax + \theta$, con A una transformación lineal y $\theta \in \mathbf{R}^n$. A se puede representar como una matriz, normalmente se identifica con las bases canónicas de $\mathbf{R}^m$ y $\mathbf{R}^n$ y las componentes de A, A_j, se nombran pesos.

Función de activación $\rho : \mathbf{R} \longrightarrow \mathbf{R}$ es una función de activación, con ρ cualquier función. Usando ρ se puede definir funciones que denotamos temporalmente por $\rho^n : \mathbf{R}^n \longrightarrow \mathbf{R}^n$, definida por

$$\rho^n \begin{pmatrix} z_1 \\ z_2 \\ . \\ . \\ z_n \end{pmatrix} := \begin{pmatrix} \rho(z_1) \\ \rho(z_2) \\ . \\ . \\ \rho(z_n) \end{pmatrix}, \text{ con } z_i \in \mathbf{R}, \forall i.$$

La capa más a la izquierda de una red neuronal es llamada la *capa de entrada*, y las neuronas en esa capa se conocen como *neuronas de entrada*. La capa más a la derecha es conocida como *capa de salida* y contiene las *neuronas de salida*. Las capas de en medio son llamadas *capas ocultas*, estas capas no son de entrada ni de salida.

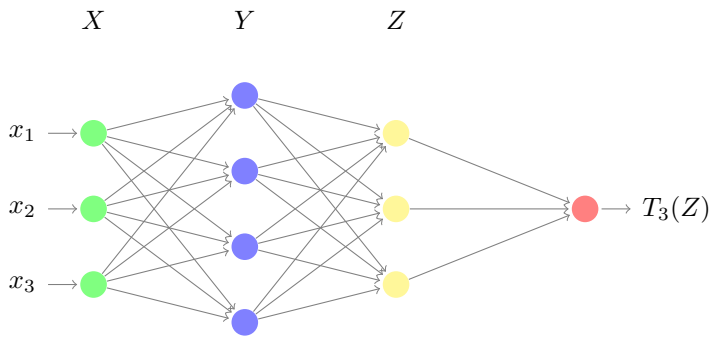


Figura 2.1: Ejemplo de Red Neuronal.

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \rho(T_1(X)) \quad Z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \rho(T_2(Y))$$

Para su implementación en Python, se recomienda el uso de la librería Torch, a través de la cual podemos el siguiente ejemplo, para la ANN descrita anteriormente.

```
#Libreria requerida
import torch
import torch.nn as nn

#Definimos la ANN

#nn.Module se utiliza para heredar las propiedades de la clase nn
class ANN(nn.Module):
#Primero definimos las transformaciones afines
  def __init__(self):
        #La funcion super se utiliza para heredar las propiedades de otra clase
    super(ANN, self).__init__()
        #La funcion nn.Linear aplica una transformacion lineal a los datos entrantes
        #de la forma xA^t + b
        #Es recomendable utilizar varias capas en vez de multiplicar los tensores
        #esto por motivos  de optimalidad
    self.fc1 = nn.Linear(3,4)
    self.fc2 = nn.Linear(4,3)
    self.fc3 = nn.Linear(3,1)
#Posteriormente definimos la aplicacion de las funciones de activacion
  def forward(self, X):
    l1 = torch.sigmoid(self.fc1(X))
    l2 = torch.sigmoid(self.fc2(l1))
    l3 = torch.sigmoid(self.fc3(l2))

    return l3
```

La gran ventaja es que una vez se ajusta su precisión se vuelven poderosas herramientas en ciencias de la computación e IA, lo cual nos permite clasificar y agrupar datos a alta velocidad. Una forma matemática de

comprender el funcionamiento de las ANN, es considerar a cada nodo individual como su propio modelo de regresión lineal

Dado que las redes neuronales se comportan de manera similar a los árboles de decisión, tener valores de x entre 0 y 1 reducirá el impacto de cualquier cambio dado de una sola variable en la salida de cualquier nodo dado, y posteriormente, la salida de la red neuronal.

A medida que vamos pensando en casos de uso más prácticos para las redes neuronales, como el reconocimiento o la clasificación de imágenes, aprovecharemos el aprendizaje supervisado o conjuntos de datos etiquetados para entrenar el algoritmo. Y mientras estamos entrenando el modelo, querremos evaluar su precisión usando una función de costo (o pérdida). Esto también se conoce comúnmente como el error cuadrático medio (ECM).

El objetivo es minimizar la función de costos para garantizar la corrección del ajuste para cualquier observación. A medida que el modelo ajusta sus pesos y sesgos, utiliza la función de costo y el aprendizaje por refuerzo para alcanzar el punto de convergencia, o el mínimo local. El proceso para que el algoritmo ajuste sus pesos es a través del descenso de gradiente, lo que permite que el modelo determine la dirección a seguir para reducir los errores o minimizar la función de costo. Con cada ejemplo de entrenamiento, los parámetros del modelo se ajustan cada vez más para converger gradualmente al mínimo.

A las redes neuronales cuya estructura en que la salida de una capa es usada como la entrada de la siguiente capa, es decir que no tiene bucles, se les conoce como *redes neuronales de pre-alimentación*. Por otro lado, las redes neuronales que tienen bucles de retroalimentación se les conoce como *redes neuronales recurrentes*.

2.4. Aprendizaje Profundo

El aprendizaje profundo es un subconjunto del aprendizaje automático , esencialmente es una red neuronal con tres o más capas. Estas redes neuronales se inspiraron en simular el comportamiento del cerebro humano, por lo que para ”aprender” se requiere una gran cantidad de datos. Si bien una red neuronal con una sola capa aún puede realizar predicciones aproximadas, las capas ocultas adicionales pueden ayudar a optimizar y refinar la precisión.

Los algoritmos de aprendizaje profundo son increíblemente complejos y existen diferentes tipos de redes neuronales para abordar problemas o conjuntos de datos específicos.

Por ejemplo:

Redes neuronales convolucionales (CNN) Son utilizadas principalmente en aplicaciones de clasificación de imágenes y visión por computadora, pueden detectar características y patrones dentro de una imagen, lo que permite tareas como la detección o el reconocimiento de objetos. Un dato interesante es que en 2015, una CNN superó a un humano en un desafío de reconocimiento de objetos por primera vez.

Redes neuronales recurrentes (RNN) Son utilizadas normalmente en aplicaciones de reconocimiento de voz y lenguaje natural, ya que aprovechan los datos secuenciales o de series de tiempo.

2.5. Aprendizaje Profundo VS Redes Neuronales Artificiales

El aprendizaje profundo y las redes neuronales tienden a usarse indistintamente en una conversación, lo que puede resultar confuso. Como resultado, vale la pena señalar que lo ”profundo” en el aprendizaje profundo se

refiere simplemente a la profundidad de las capas en una red neuronal. Una red neuronal que consta de más de tres capas, que incluirían las entradas y la salida, se puede considerar un algoritmo de aprendizaje profundo. Una red neuronal que solo tiene dos o tres capas es solo una red neuronal básica.

Algunas aplicaciones del aprendizaje profundo son parte de nuestra vida cotidiana, están tan bien integradas en productos y servicios, que en la mayoria de los casos, los usuarios no son conscientes del complejo procesamiento de datos que se lleva a cabo en segundo plano. Ejemplo de ello es que los algoritmos de aprendizaje profundo pueden analizar y aprender de los datos transaccionales para identificar patrones peligrosos que indican una posible actividad fraudulenta o delictiva que vaya en contra del **cumplimiento de la ley**. También para **servicios financieros,** las instituciones utilizan regularmente el análisis predictivo para impulsar la negociación algorítmica de acciones, evaluar los riesgos comerciales para la aprobación de préstamos, detectar fraudes y ayudar a administrar las carteras de crédito e inversión de los clientes. Otros ejemplos son aplicaciones pensadas para el **servicio al cliente** o el **cuidado de la salud.**

Un punto importante sobre el aprendizaje profundo es que requiere una enorme cantidad de potencia informática. Las unidades de procesamiento gráfico (GPU) de alto rendimiento son ideales porque pueden manejar un gran volumen de cálculos en múltiples núcleos con abundante memoria disponible. Pero administrar varias GPU en las instalaciones puede crear una gran demanda de recursos internos y ser increíblemente costoso su uso.

3 | Resultados de Análisis Funcional

3.1. Espacios de Banach y Espacios de Hilbert

Definición 3.1 (Norma)

Sea V un espacio vectorial. Una norma en V es una asignación $\widehat{x} \to \|\widehat{x}\|$ para todo $\widehat{x}$ en V, tal que:

1. $\|\widehat{x}\| \geq 0$,
2. $\|\widehat{x}\| = 0 \Leftrightarrow \widehat{x} = \widehat{0}$,
3. $\|\alpha \cdot \widehat{x}\| = |\alpha|\|\widehat{x}\| \ \forall \alpha \in \mathbb{R}$,
4. $\|\widehat{x} + \widehat{y}\| \leq \|\widehat{x}\| + \|\widehat{y}\|$.

Definición 3.2 (Producto Interior)

Sea V un espacio vectorial. Un producto interior en el espacio de los números reales es una asignación $\widehat{x}, \widehat{y} \to \langle \widehat{x}, \widehat{y} \rangle$ para cualesquiera $\widehat{x}, \widehat{y}$ en V, tal que:

1. $\langle \widehat{u} + \widehat{v}, \widehat{w} \rangle = \langle \widehat{u}, \widehat{w} \rangle + \langle \widehat{v}, \widehat{w} \rangle$,
2. $\langle \lambda \widehat{v}, \widehat{w} \rangle = \lambda \langle \widehat{v}, \widehat{w} \rangle \ \forall \lambda \in \mathbb{R}$,
3. $\langle \widehat{u}, \widehat{v} \rangle = \langle \widehat{v}, \widehat{u} \rangle$,
4. $\langle \widehat{v}, \widehat{v} \rangle \geq 0, \langle \widehat{v}, \widehat{v} \rangle = 0 \Leftrightarrow \widehat{v} = 0$.

Notación 3.1

Todo producto interior induce una norma de la forma

$$\|\widehat{x}\| = \sqrt{\langle \widehat{x}, \widehat{x} \rangle}.$$

Definición 3.3 (Completitud)

Decimos que un espacio vectorial normado V es completo si toda sucesión de Cauchy converge, es decir: Sea $(\widehat{v}_n)_{n \in \mathbb{N}}$ una sucesión de elementos en V tal que

$$\forall \varepsilon > 0, \exists N \in \mathbb{N}, \text{ tal que } \forall n, m \geq N, \|\widehat{v}_n - \widehat{v}_m\| < \varepsilon,$$

entonces existe

$$\widehat{v} = \lim_{n \to \infty} \widehat{v}_n.$$

Definición 3.4 (Espacios de Banach y de Hilbert)

- Un espacio de Banach es un espacio vectorial normado y completo.
- Un espacio de Hilbert es una espacio vectorial con producto interior y completo.

Notación 3.2

Sea V un espacio vectorial normado. Se usará la siguiente notación:

- Bola:
  Dado $\widehat{x} \in V$ y $r > 0$, denotamos a la bola con centro en $\widehat{x}$ y radio r por
  $$B(\widehat{x}, r) := \{y \in V \mid \|\widehat{y} - \widehat{x}\| < r\}.$$
- Conjunto Abierto:
  Dado $A \subset V$, decimos que A es abierto si para todo $\widehat{a} \in A$, existe $r > 0$ tal que $B(\widehat{a}, r) \subset A$.
- Conjunto Cerrado:
  Dado $B \subset V$, decimos que B es cerrado si B^c es abierto o equivalentemente si la siguiente proposición es verdadera:
  $$\text{Si } \widehat{y} \in V \text{ es tal que, para todo } \epsilon > 0 \text{ tal que } B(\widehat{y}, \epsilon) \cap B \neq \emptyset, \text{ entonces } \widehat{y} \in B.$$
- Cerradura:
  La cerradura de un conjunto C es el conjunto cerrado mas chico que contiene a C. Denotamos con $\bar{C}$ a la cerradura de C.
  El conjunto $\bar{C}$ se caracteriza con la siguiente propiedad:
  $$\widehat{y} \in \overline{C} \Leftrightarrow \forall \epsilon > 0, B(\widehat{y}, \epsilon) \cap C \neq \emptyset.$$
  Notemos que C es cerrado $\Leftrightarrow C = \overline{C}$.
- Interior:
  El interior de un conjunto A es el conjunto abierto mas grande contenido en A. Denotamos con A° al interior de A.

Definición 3.5 (Densidad)

Sean V un espacio vectorial normado y $M \subset V$. Decimos que M es denso si se cumple alguno de los siguientes puntos:

1. $\overline{M} = V$.
2. $\forall\, \widehat{v} \in V, \forall\, \varepsilon > 0$ tal que $B(\widehat{v}, \varepsilon) \cap M \neq \emptyset$.

Definición 3.6 (Norma de una transformación lineal continua)

Sean V y W espacios vectoriales normados y $\Lambda : V \to W$ una transformación lineal. Definimos:

$$\|\Lambda\| := \sup\{\|\Lambda(\widehat{v})\| \ : \ \widehat{v} \in V \ y \ \|\widehat{v}\| \leq 1\}.$$

Note que Λ es continua si y sólo si $\sup\{\|\Lambda(\widehat{v})\| \ : \ \widehat{v} \in V \ y \ \|\widehat{v}\| \leq 1\} < \infty$.

Teorema 3.1 (Hahn-Banach)

Sean V un espacio vectorial normado y $M \subset V$ un sub-espacio vectorial. Supongamos $h : M \to \mathbb{R}$ es continua y lineal, entonces existe una extensión $g : V \to \mathbb{R}$ (lineal y continua) tal que:

1. $g(m) = h(m)\ \forall m \in M$,
2. $\|h\| = \|g\|$.

Corolario 3.1

Sean V un espacio vectorial normado y $M \subset V$ un sub-espacio vectorial. Suponemos que M no es denso (i.e. $\overline{M} \neq V$), entonces existe una función lineal $g : V \to \mathbb{R}$ continua tal que:

1. $g|_{\overline{M}} = 0$,
2. $g \neq 0$.

Demostración. Como $\overline{M} \neq V$, entonces existe $\widehat{x} \in V \setminus \overline{M}$. Sea W el espacio vectorial generado por $\overline{M}$ y $\widehat{x}$, entonces todo elemento de W se puede escribir de manera única de la siguiente manera:

$$\widehat{w} = \alpha\widehat{x} + \widehat{m}$$

en donde $\alpha \in \mathbb{R}$ y $m \in \overline{M}$; es decir

$$W = \{\alpha\widehat{x} + \widehat{m} \mid \alpha \in \mathbb{R}, \widehat{m} \in \overline{M}\}.$$

Veamos que los elementos de W se pueden escribir de forma única como se describe antes.

Supongamos que existen α_1, α_2 en $\mathbb{R}$ y $\widehat{m}_1$, $\widehat{m}_2$ en $\overline{M}$ tales que:

$$\begin{aligned} & \widehat{w} = \alpha_1\widehat{x} + \widehat{m}_1 = \alpha_2\widehat{x} + \widehat{m}_2, \\ \Rightarrow \quad & (\alpha_1 - \alpha_2)\widehat{x} = \widehat{m}_2 - \widehat{m}_1 \in \overline{M}. \end{aligned}$$

(Pues como M es espacio vectorial, entonces $\overline{M}$ también lo es).

Concluimos que $(\alpha_1, \alpha_2)\widehat{x} \in \overline{M}$ pues si $\alpha_1 - \alpha_2 \neq 0$, entonces tendriamos que existe $\widehat{y} \in \overline{M}$ tal que $\widehat{x} = (\frac{1}{\alpha_1 - \alpha_2})\widehat{y} \in \overline{M}$, lo que contradice que $\widehat{x} \notin \overline{M}$.

Entonces $\alpha_1 - \alpha_1 = 0$, por lo que $\alpha_1 = \alpha_2$. Como $(\alpha_1 - \alpha_2)\widehat{x} = m_2 - m_1$, concluimos que $m_1 = m_2$.

Definimos a $h : W \to \mathbb{R}$ como $h(\alpha\widehat{x} + \widehat{m}) = \alpha\|\widehat{x}\|$, que es lineal, pues:

$$\begin{aligned} h(\lambda(\alpha_1\widehat{x} + \widehat{m}) + \alpha_2\widehat{x} + \widehat{m}_2) &= h(\lambda\alpha_1 + \alpha_2)\widehat{x} + \lambda\widehat{m}_1 + \widehat{m}_2 \\ &= (\lambda\alpha_1 + \alpha_2)\|x\| = \lambda h(\alpha_1\widehat{x} + \widehat{m}_1) + h(\alpha_2\widehat{x} + \widehat{m}_2), \end{aligned}$$

en donde $\lambda\widehat{m}_1 + \widehat{m}_2 \in \overline{M}$.

Para toda $\lambda \in \mathbb{R}$, $\alpha_1, \alpha_2 \in \mathbb{R}$, $\widehat{m}_1, \widehat{m}_2 \in \overline{M}$.
Además $h(m) = 0$, $\forall\, m \in \overline{M}$.
Veamos que $h : W \to \mathbb{R}$ es continua para lo cual verificamos que

$$\|h\| = \sup\{|h(\widehat{w}) \mid \widehat{w} \in W, \|\widehat{w}\| \leq 1|\} \leq \infty.$$

Como $\widehat{x} \in V \setminus \overline{M}$, entonces

$$\exists \varepsilon > 0 \text{ tal que } B(\widehat{x}, \varepsilon) \cap \overline{M} = \emptyset.$$

Sea $\widehat{w} = \alpha\widehat{x} + \widehat{m} \in W$ tal que $\|\widehat{w}\| \leq 1$, tenemos que

$$\|\alpha\widehat{x} + \widehat{m}\| = |\alpha|\|\widehat{x} + \frac{1}{\alpha}\widehat{m}\| \leq 1$$

y además como $\frac{1}{\alpha}\widehat{m} \in \overline{M}$, se obtiene que

$$\|\widehat{x} - (-\frac{1}{\alpha}\widehat{m})\| = \|\widehat{x} + \frac{1}{\alpha}\widehat{m}\| \geq \varepsilon.$$

Concluimos que

$$|\alpha|\|\widehat{x} + \frac{1}{\alpha}\widehat{m}\| \leq 1 \ \Rightarrow |\alpha| \leq \frac{1}{\|\widehat{x} + \frac{1}{\alpha}\widehat{m}\|} \leq \frac{1}{\varepsilon}.$$

Se tiene que

$$|\alpha| \leq \frac{1}{\varepsilon}, \forall\, \widehat{w} = \alpha\widehat{x} + \widehat{m} \in W \text{ tal que } \|\widehat{w}\| \leq 1,$$

de lo que obtenemos

$$|h(\widehat{w})| = |\alpha|\|\widehat{x}\| \leq \frac{1}{\varepsilon}\|\widehat{x}\| \qquad \forall\, \widehat{w}, \text{ con } \|\widehat{w}\| \leq 1, \text{ con } w = \alpha\widehat{x} + \widehat{m}.$$

Concluimos que

$$sup\{\, \|h(\widehat{w})\| \mid \widehat{w} \in W, |\widehat{w}| \leq 1 \,\} = |h| \leq \frac{1}{\varepsilon}\|\widehat{x}\|$$

y por lo tanto es continua.

Así, h es una función continua tal que $h \neq 0$, puesto que $h(\widehat{x}) = \|\widehat{x}\|$ y además $h(\widehat{m}) = 0$, $\forall\, \widehat{m} \in \overline{M}$.

Por el Teorema de Hahn-Banach, existe g lineal y continua tal que:

- $g(\widehat{w}) = h(\widehat{w})$ $\forall \widehat{w} \in W$,
- $\|g\| = \|h\|$ (en particular g es continua).

g es la función que buscamos, pues:

- $g \neq 0$ (ya que $h \neq 0$),
- $g(\widehat{m}) = h(\widehat{m}) = 0$ $\forall \widehat{m} \in \overline{M}$.

□

3.2. El Dual de las Funciones Continuas e Integrables

Definición 3.7 (Dual de un espacio vectorial)

Sea V un espacio vectorial normado. Denotamos por $V^\bullet$ al conjunto de funciones lineales y continuas de V en $\mathbb{R}$.

Definición 3.8

Sea $K \subset \mathbb{R}$ un conjunto compacto (cerrado y acotado). Denotamos por

$$C(K) := \{f : K \to \mathbb{R} : \text{ f es continua } \}$$

al espacio vectorial de las funciones continuas en K con valores en $\mathbb{R}$, dotado de la norma.

$$\|f\|_\infty := sup\{|f(\widehat{x}| \ : \ \widehat{x} \in K\}.$$

Recordemos que $C(k)^\bullet$ es el dual de $C(K)$ el cual es el conjunto de funciones lineales y continuas de $C(K)$ en $\mathbb{R}$ y que la función lineal $h : C(K) \to \mathbb{R}$ es continua si y sólo si

$$\|h\| := sup\{|h(\widehat{x}| : \widehat{x} \in C(K)\} < \infty.$$

Definición 3.9 (Teorema de Representación de Riez)

Decimos que una función f acotada, $f : K \to \mathbb{R}$, es integrable respecto a $h \in C(K)$ si existe una sucesión de funciones $\{f_n\}_{n\in\mathbb{N}}$ en $C(K)$ tal que:

1. Existe $c \in \mathbb{R}$ tal que
$$|f_n(\widehat{x})| \leq c \ \ \forall\, \widehat{x}\,, \forall\, n;$$
2. $\lim\limits_{n\to\infty} f_n(\widehat{x}) = f(\widehat{x}) \ \forall\, \widehat{x} \in K.$

Notación 3.3

Denotaremos por $L^1_a(h)$ al espacio de funciones integrables y acotadas con respecto a h.

Notemos que en está definición $L^1_a(h)$ no depende de h, pero mantenemos h en nuestra notación ya que se utilizará posteriormente.

Teorema 3.2 (Convergencia Dominada de Lebesgue 1)

Sean $f \in L^1_a(h)$ para algún $h \in C(K)^\bullet$ y $\{f_n\}_{n\in\mathbb{N}}$ como en 3.9, entonces $\lim_{n\to\infty} h(f_n)$, existe y si $\{g_n\}_{n\in\mathbb{N}}$ es otra sucesión que cumple 1 y 2 de la definición 3.9, entonces

$$\lim_{n\to\infty} h(f_n) = \lim_{n\to\infty} h(g_n).$$

Es fácil ver que $L^1_a(h)$ es un espacio vectorial y que la función $f \to \int f\, dh$ es lineal.

Teorema 3.3 (Convergencia Dominada de Lebesgue 2)

Sea $h \in C(K)^\bullet$. Supongamos que $g \in L^1_a(h)$ es tal que existe una sucesión $\{g_n\}_{n\in\mathbb{N}}$ de elementos en $L^1_a(h)$, que cumple lo siguiente:

1. Existe $G \in L^1_a(h)$ tal que
$$|g_n(\widehat{x})| \leq G(\widehat{x}) \ \ \forall\, \widehat{x} \in K, \ \forall\, n \in \mathbb{N};$$
2. $\lim\limits_{n\to\infty} g_n(\widehat{x}) = g(\widehat{x}) \ \forall\, \widehat{x} \in K,$

entonces

$$\lim_{n\to\infty} \int g_n\, dh = \int g\, dh.$$

Definición 3.10

Sea $K \subset \mathbb{R}^d$ compacto. Dado $x \in \mathbb{R}^d$ y $a \in \mathbb{R}^d$, denotamos por

$$\langle \widehat{a}, \widehat{x} \rangle = \sum_{i=1}^{d} \widehat{a}_i \widehat{x}_i$$

en donde

$$\widehat{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_d \end{pmatrix}, \widehat{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}.$$

Lema 3.1

Dados $b_1, b_2 \in \mathbb{R}$ con $b_1 < b_2$ y $f \in C(K)$, se cumple que

$$\chi_{[b_1,b_2]} \circ f \in L_a^1(h),$$

en donde $f : K \subset \mathbb{R}^d \to \mathbb{R}$ y $\chi_{[b_1,b_2]} : \mathbb{R} \to \mathbb{R}$ es tal que

$$\chi_{[b_1,b_2]}(y) = \begin{cases} 1 & y \in [b_1, b_2], \\ 0 & y \notin [b_1, b_2]. \end{cases}$$

Teorema 3.4

Dado $\widehat{a} \in \mathbb{R}^d$ definimos la función $f_{\widehat{a}} : \mathbb{R}^d \to \mathbb{R}$ dada por $f_{\widehat{a}}(\widehat{x}) = \langle \widehat{a}, \widehat{x} \rangle$.
Sea $h \in C(K)^\bullet$. Si para cualesquiera $a \in \mathbb{R}^d$ y $b_1, b_2 \in \mathbb{R}$ con $b_1 < b_2$ se cumple que

$$\int \chi_{[b_1,b_2]} \circ f_{\widehat{a}} \, dh = 0,$$

entonces $h = 0$.

4 | Aproximación por medio de Redes Neuronales

4.1. Universalidad

Recordemos que una función F de la forma

$$F = T_L \circ \rho \circ T_{L-1} \circ \rho \cdots \circ \rho \circ T_1,$$

en donde $T_1, \ldots, T_L$ son transformaciones afines y $\rho : \mathbb{R} \to \mathbb{R}$ es una función de activación, es nombrada "multilayer perceptron"(MLP) con dimensión de entrada d, L niveles y función de activación ρ; en donde el dominio de T_1 es $\mathbb{R}^d$.

Definición 4.1 (MLP)

Sean $\rho : \mathbb{R} \to \mathbb{R}$ continua, $d, L \in \mathbb{N}$ y $K \subset \mathbb{R}^d$ compacto. Denotamos por $MLP(\rho, d, L)$ al conjunto de MLP´s con función de activación ρ, L niveles y dimensión de entrada d tales que T_L toma valores en $\mathbb{R}$; es decir, si $f \in MLP(\rho, d, L)$, entonces

$$f : \mathbb{R}^d \to \mathbb{R}.$$

Definición 4.2 (Universalidad)

Decimos que $MLP(\rho, d, L)$ es universal si las restricciones de las funciones $f \in MLP(\rho, d, L)$ son densas en $C(K)$ para cualquier compacto $K \subset \mathbb{R}^d$.

Normalmente se identifican las funciones $f : \mathbb{R}^d \to \mathbb{R}$ con sus restricciones a K, $f|_K \equiv f : K \to \mathbb{R}$.

Una manera de interpretar que el conjunto de MLP´s sea denso en el conjunto de funciones continuas, es que para cada función continua, existe un MLP que lo aproxima.

Definición 4.3 (Discriminante)

Sean $d \in \mathbb{N}$ y $K \subset \mathbb{R}^d$ compacto. Una función $f : \mathbb{R} \to \mathbb{R}$ continua es llamada discriminante si dada $h \in C(K)^\bullet$ se cumple la siguiente proposición para toda $\widehat{a} \in \mathbb{R}^d$, $b \in \mathbb{R}$:

$$h(t_{\widehat{a},b}(f)) = 0 \Rightarrow h = 0.$$

En donde denotamos por $t_{\widehat{a},b}(f) \in C(K)$ a la función dada por $t_{\widehat{a},b}(f)(\widehat{x}) = f(\langle \widehat{a}, \widehat{x} \rangle - b)$.

Proposición 4.1

$MLP(\rho, d, 2)$ es un espacio vectorial.

Demostración. Sean $F, G \in MLP(\rho, d, 2)$ y $\alpha \in \mathbb{R}$, veamos que

$$\alpha F + G \in MLP(\rho, d, 2).$$

Como $F, G \in MLP(\rho, d, 2)$, entonces existen transformaciones afines T_1, T_2, V_1, V_2 tales que:

- $F = T_2 \circ \rho \circ T_1$,
- $G = V_2 \circ \rho \circ V_2$;

en donde T_1 y V_1 tiene dominio $\mathbb{R}^d$ y T_2, V_2 toman valores en $\mathbb{R}$.
Definimos

$$U_1(\widehat{x}) = \begin{pmatrix} T_1(\widehat{x}) \\ V_1(\widehat{x}) \end{pmatrix}, \ \forall\, \widehat{x} \in \mathbb{R}^d;$$

U_1 es una transformación afín.

Dados y, z con y en el dominio de T_2 y z en el dominio de V_2, definimos

$$U_2 \begin{pmatrix} y \\ z \end{pmatrix} = \alpha T_2(y) + V_2(z).$$

Tenemos que

$$\begin{aligned} U_2 \circ \rho \circ U_1(\widehat{x}) &= U_2 \ \rho \begin{pmatrix} T_1(\widehat{x}) \\ V_1(\widehat{x}) \end{pmatrix} \\ &= \alpha T_2(\rho(T_1(\widehat{x})) + V_2(\rho(V_1(\widehat{x}))) \\ &= \alpha F(\widehat{x}) + G(\widehat{x}). \end{aligned}$$

Entonces $\alpha F + G = U_2 \circ \rho \circ U_1 \in MLP(\rho, d, 2)$, de lo que concluimos que $MLP(\rho, d, 2)$ es un espacio vectorial. □

Teorema 4.1 (Aproximación Universal, Gybenko 1989)

Sean $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compacto y $\rho : \mathbb{R} \to \mathbb{R}$ una función discriminante. Entonces $MLP(\rho, d, 2)$ es denso en $C(K)$.

Demostración. Procederemos por contradicción:

Supongamos que $MLP(\rho, d, 2)$ no es denso en $C(K)$, entonces $\overline{MLP(\rho, d, 2)} \subsetneq C(K)$ por lo que existe $f \in C(K) \setminus \overline{MLP(\rho, d, 2)}$. Por el Corolario 3.1 existe una función $h \in C(K)^\bullet$ tal que:

- $h \neq 0$,
- $h|_{\overline{MLP(\rho,d,2)}} = 0$.

Sea $T_{\widehat{a},b} : \mathbb{R}^d \to \mathbb{R}$, dada por $T_{\widehat{a},b} = \langle \widehat{a}, \widehat{x} \rangle - b$, en donde $a \in \mathbb{R}^d$, $b \in \mathbb{R}$.
Se tiene que

$$\rho \circ T_{\widehat{a},b} = Id \circ \rho \circ T_{\widehat{a},b} \in MLP(\rho, d, 2),$$

en donde $Id : \mathbb{R} \to \mathbb{R}$ es la función identidad.

Como $h|_{\overline{MLP(\rho,d,2)}} = 0$, concluimos que

$$h(\rho \circ T_{\widehat{a},b}) = 0, \ \forall \widehat{a} \in \mathbb{R}^d, \forall b \in \mathbb{R}.$$

Notemos que

$$\rho \circ T_{\widehat{a},b}(\widehat{x}) = \rho(\langle \widehat{a}, \widehat{x} \rangle - b) = t_{\widehat{a},b}(\rho)(\widehat{x}),$$

así,

$$h(t_{\widehat{a},b}(\rho)) = 0, \ \forall \widehat{a} \in \mathbb{R}^d, \ \forall b \in \mathbb{R}.$$

Como supusimos que ρ es discriminante, esto implica que $h = 0$.

Esto es una contradicción pues $h \neq 0$. La contradicción se obtiene por suponer que $MLP(\rho, d, 2)$ no es denso por lo que concluimos que $MLP(\rho, d, 2)$ es denso en $C(K)$. □

Definición 4.4 (Sigmoidal)

Una función $f : \mathbb{R} \to \mathbb{R}$ continua tal que:

$$\lim_{r \to \infty} f(r) = 1, \ \lim_{r \to -\infty} f(r) = 0,$$

se llama sigmoidal.

Proposición 4.2

Sean $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compacto. Toda función sigmoidal $\rho : \mathbb{R} \to \mathbb{R}$ es discriminante (y por lo tanto $\overline{MLP(\rho, d, 2)}$ es denso en $C(K)$, por el Teorema 4.1).

Demostración. Sea ρ sigmoidal, entonces dados λ, b, $\theta \in \mathbb{R}$ y $\widehat{a}, \widehat{x} \in \mathbb{R}^d$, tenemos que:

$$\lim_{\lambda \to \infty} \rho(\lambda(\langle \widehat{a}, \widehat{x} \rangle - b) + \theta) = \begin{cases} 1 & \text{si } \langle \widehat{a}, \widehat{x} \rangle - b > 0, \\ \rho(\theta) & \text{si } \langle \widehat{a}, \widehat{x} \rangle - b = 0, \\ 0 & \text{si } \langle \widehat{a}, \widehat{x} \rangle - b < 0. \end{cases}$$

Sea $h \in C(K)^\bullet$ y sean:

$$H_{\widehat{a},b,>} := \{\widehat{x} \in K \ : \ \langle \widehat{a}, \widehat{x} \rangle - b > 0\},$$
$$H_{\widehat{a},b,=} := \{\widehat{x} \in K \ : \ \langle \widehat{a}, \widehat{x} \rangle - b = 0\},$$
$$H_{\widehat{a},b,<} := \{\widehat{x} \in K \ : \ \langle \widehat{a}, \widehat{x} \rangle - b < 0\}.$$

Sea $g_{\lambda,\widehat{a},b,\theta}(\widehat{x}) = \rho(\lambda(\langle \widehat{a}, \widehat{x} \rangle - b+)\theta)$, entonces

$$\lim_{\lambda \to \infty} g_{\lambda,\widehat{a},b,\theta}(\widehat{x}) = X_{H\widehat{a},b,>}(\widehat{x}) + \rho(\theta) X_{H\widehat{a},b,=}(\widehat{x}).$$

Ahora veamos que ρ es discriminadora, es decir, si $h \in C(K)^\bullet$ es tal que $h(t_{\widehat{a},b}(\rho)) = 0$ para cualesquiera $\widehat{a} \in \mathbb{R}^d$ y $b \in \mathbb{R}$, únicamente tenemos que ver $h = 0$.

Notemos que:

$$g_{\lambda,\widehat{a},b,\theta} = t_{\lambda\widehat{a},\lambda b - \theta}(\rho),$$

entonces

$$h(g_{\lambda,\widehat{a},b,\theta}) = 0 \ \forall \lambda, \ \forall \widehat{a}, \ \forall b, \ \forall \theta.$$

Por el teorema de convergencia dominada de Lebesgue 2, obtenemos que

$$\begin{aligned} 0 = \lim_{\lambda \to \infty} h(g_{\lambda,\widehat{a},b,\theta}) &= \int \lim_{\lambda \to \infty} g_{\lambda,\widehat{a},b,\theta} \, dh \\ &= \int (\chi_{H\widehat{a},b,>} + \rho(\theta) \chi_{H\widehat{a},b,=}) \, dh \\ &= \int \chi_{H\widehat{a},b,>} \, dh + \rho(\theta) \int \chi_{H\widehat{a},b,=} \, dh. \end{aligned}$$

Así,

$$\int \chi_{H\widehat{a},b,>} \, dh + \rho(\theta) \int \chi_{H\widehat{a},b,=} \, dh = 0 \ \forall \widehat{a}, \ \forall b, \ \forall \theta.$$

1. Haciendo θ tender a ∞ obtenemos

$$\int \chi_{H\widehat{a},b,>} \, dh + \int \chi_{H\widehat{a},b,=} \, dh = 0.$$

2. Haciendo θ tender a $-\infty$ obtenemos

$$\int \chi_{H\widehat{a},b,>} \, dh = 0.$$

Sea $f_{\widehat{a}}(\widehat{x}) := \langle \widehat{a}, \widehat{x} \rangle \ \forall \widehat{x} \in K$.

$$\begin{aligned} H_{\widehat{a},b,>} &= \{\widehat{x} \in K : \langle \widehat{a}, \widehat{x} \rangle - b > 0\} = f_{\widehat{a}}^{-1}((b, \infty)) \\ &\Rightarrow H_{\widehat{a},b,=} = f^{-1}(b) \\ &\Rightarrow \chi_{H\widehat{a},b,>} + \chi_{H\widehat{a},b,=} = \chi_{f_{\widehat{a}}^{-1}([b,\infty))}. \end{aligned}$$

Dados $b_1, b_2 \in \mathbb{R}$ con $b_1 < b_2$, obtenemos que

$$\chi_{H\widehat{a},b_1,>} + \chi_{H\widehat{a},b,=} - \chi_{H\widehat{a},b_2,>} = \chi_{f_{\widehat{a}}^{-1}([b_1,b_2])}.$$

De (1) y (2) obtenemos que

$$\int \chi_{f_{\widehat{a}}^{-1}([b_1,b_2])} \, dh = 0.$$

Notemos que

$$\chi_{f_{\hat{a}}^{-1}([b_1,b_2])} = \chi_{[b_1,b_2]} \circ f_{\hat{a}}$$

así, para cualesquiera $b_1, b_2 \in \mathbb{R}$ con $b_1 < b_2$ y $\widehat{a} \in \mathbb{R}^d$,

$$\int \chi_{[b_1,b_2]} \circ f_{\,\hat{a}} \; dh = 0.$$

Usamos el Teorema 3.4 para concluir que h=0. Por lo tanto ρ es discriminante. □

4.2. Redes Neuronales

Recordemos que las transformaciones afines consisten de transformaciones lineales compuestas con traslaciones. Dada una transformación afín T, existe una matriz A y vector $\widehat{b}$ tales que

$$T(\widehat{x}) = A\widehat{x} + \widehat{b};$$

de manera que identificamos

$$T \equiv (A, \widehat{b}).$$

Dada un red neuronal

$$F = T_L \circ \rho \circ T_{L-1} \circ \rho \circ \cdots \circ \rho \circ T_1$$

en donde $T_1, T_2, \cdots, T_L$ son transformaciones afines y ρ una función de activación, identificamos

$$F_\rho \equiv (T_1, T_2, \cdots, T_L);$$

en caso de no ser necesario resaltar la función de activación, simplemente identificamos

$$F \equiv (T_1, T_2, \cdots, T_L).$$

Convengamos de ahora en adelante que a $(T_1, T_2, \cdots, T_L)$ se le nombra red neuronal o neural network (NN) y F es su realización. Así, dada una red neuronal

$$\phi = (T_1, T_2, \cdots, T_L),$$

para denotar su realización utilizamos

$$F_\rho = R_\rho(\phi);$$

en caso de no ser necesario resaltar la función de activación, simplemente denotamos

$$F = R(\phi).$$

Otra notación que se usa es:

- $x_0 = \widehat{x}$,
- $x_1 = \rho \circ T_1(x_0)$,

$\vdots$

- $x_p = \rho \circ T_p \circ x_{p-1}$,

$\vdots$

- $x_L = T_L \circ x_{L-1}$.

Definición 4.5

Sea $\phi = (T_1, T_2, \cdots, T_L)$ una red neuronal donde $T_p : \mathbb{R}^{N_{p-1}} \to \mathbb{R}^{N_p}$ y $N_0 = d$.

Definimos los siguientes conceptos:

- La dimensión de entrada de ϕ es d.
- El número de niveles o capas de ϕ es L, también denotado por $L(\phi) = L$.
- El número de neuronas está dado por $\sum_{j=0}^{L} N_j$.
- Dada una matriz A y un vector $\widehat{b}$, denotamos por:

$$\|A\|_0 \text{ al número de entradas no cero de } A,$$
$$\|\widehat{b}\|_0 \text{ al número de entradas no cero de } \widehat{b}.$$

- Dada una transformación afín $T \equiv (A, \widehat{b})$, definimos $\|T\|_0 := \|A\|_0 + \|\widehat{b}\|_0$.
- Denotamos por:

$$M(\phi) = \sum_{j=1}^{L} M_j(\phi)$$

donde

$$M_j(\phi) = \|T_j\|_0.$$

4.3. Operaciones Básicas con Redes Neuronales

Definición 4.6 (Concatenación)

Sean ϕ_1, ϕ_2 redes neuronales tales que

$$\phi_1 = (T_1, T_2, \cdots, T_L),$$
$$\phi_2 = (V_1, V_2, \cdots, V_M).$$

Definimos la concatenación de ϕ_1 y ϕ_2 como

$$\phi_1 \bullet \phi_2 := (V_1, V_2, \cdots, V_{M-1}, T_1 \circ V_M, T_2, \cdots, T_L).$$

La cual es una red neuronal de $L + M - 1$ niveles.

Es directo ver que

$$R(\phi_1 \bullet \phi_2) = R(\phi_1) \circ R(\phi_2).$$

Definición 4.7

Dados T y V transformaciones afines, definimos las siguientes transformaciones afines:

- $\begin{pmatrix} T \\ V \end{pmatrix}(\widehat{x}) = \begin{pmatrix} T(\widehat{x}) \\ V(\widehat{x}) \end{pmatrix}$
  donde T y V tienen el mismo dominio.
- $T \oplus V \begin{pmatrix} \widehat{x} \\ \widehat{y} \end{pmatrix} = \begin{pmatrix} T(\widehat{x}) \\ V(\widehat{y}) \end{pmatrix}$
  donde T y V no necesariamente tienen el mismo dominio.

Definición 4.8

Sean ϕ_1, ϕ_2 redes neuronales tales que

$$\phi_1 = (T_1, T_2, \cdots, T_L),$$
$$\phi_2 = (V_1, V_2, \cdots, V_L).$$

- Definimos la paralelización con entrada compartida de ϕ^1 y ϕ^2 como

$$P(\phi_1, \phi_2) = \left(\begin{pmatrix} T_1 \\ V_1 \end{pmatrix}, T_2 \oplus V_2, \cdots, T_L \oplus V_L \right)$$

  donde las dimensiones de entrada de ϕ_1 y ϕ_2 son la misma.

- También definimos a la paralelización con entradas no compartidas de ϕ_1 y ϕ_2 como

$$FP(\phi_1, \phi_2) = (T_1 \oplus V_1, T_2 \oplus V_2, \cdots, T_L \oplus V_L)$$

  donde las dimensiones de entrada de ϕ_1 y ϕ_2 no necesariamente son la misma.

Proposición 4.3

1. $M(P(\phi_1, \phi_2)) = M(FP(\phi_1, \phi_2)) = M(\phi_1) + M(\phi_2)$.
2. $R(P(\phi_1, \phi_2))(\widehat{x}) = \begin{pmatrix} R(\phi_1(\widehat{x})) \\ R(\phi_2(\widehat{x})) \end{pmatrix}$.

Proposición 4.4

Sean $d \in \mathbb{N}$ y $K \subseteq \mathbb{R}^d$ compacto. Supongamos que $\rho : \mathbb{R} \to \mathbb{R}$ es no constante y diferenciable en un abierto, entonces, para todo $\varepsilon > 0$, existe una red neuronal ϕ tal que

$$\phi = (T_1, T_2)$$

donde $T_1, T_2 : \mathbb{R}^d \to \mathbb{R}^d$, $M(\phi) \leq 4d$ y $\|R(\phi)(\widehat{x}) - \widehat{x}\|_{\mathbb{R}^d} < \varepsilon$ para todo $\widehat{x} \in K$.

Demostración. Sean $d = 1$, $x^\bullet \in \mathbb{R}$ tal que ρ es diferenciable en una vecindad alrededor de $x^\bullet$ y supongamos que

$$\rho'(x^\bullet) = \theta \neq 0.$$

Dada $\lambda > 0$ definimos:

$$T_1(\widehat{x}) = \frac{1}{\lambda}\widehat{x} + x^\bullet,$$
$$T_2(\widehat{x}) = \frac{\lambda}{\theta}\widehat{x} - \lambda\rho\frac{(x^\bullet)}{\theta}.$$

Se tiene que

$$|\underbrace{R(\phi)(\widehat{x})}_{T_2(\rho\circ T_1(\widehat{x}))} - \widehat{x}| = \left|\frac{\lambda}{\theta}(\rho(\overbrace{\frac{\widehat{x}}{\lambda} + x^\bullet}^{T_1(\widehat{x})}) - \rho(x^\bullet)) - \widehat{x}\right|.$$

Es claro que $|R(\phi)(0) - 0| = 0$ y para $\widehat{x} \neq 0$, se tiene que

$$\begin{aligned}|R(\phi)(\widehat{x}) - \widehat{x}| &= \frac{|\widehat{x}|}{|\theta|}\left|\frac{\rho(\frac{\widehat{x}}{\lambda} + x^\bullet) - \rho(x^\bullet)}{\frac{\widehat{x}}{\lambda}} - \theta\right| \\ &= \frac{|\widehat{x}|}{|\theta|}\left|\frac{\rho(x^\bullet + h_\lambda) - \rho(x^\bullet)}{h_\lambda} - \rho'(x^\bullet)\right|.\end{aligned}$$

Así,

$$|R(\phi)(\widehat{x}) - \widehat{x}| = \frac{|\widehat{x}|}{|\theta|}\left|\frac{\rho(x^\bullet + h_\lambda) - \rho(x^\bullet)}{h_\lambda} - \rho'(x^\bullet)\right|.$$

Sea $\alpha = max\{|\widehat{x}| : \widehat{x} \in K\}$. Como ρ es diferenciable en un intervalo y que contiene a $x^\bullet$ entonces dada $\varepsilon > 0$ existe $\delta > 0$ tal que si $|y| < d$, entonces

$$\left|\frac{\rho(x^\bullet + y) - \rho(x^\bullet)}{y} - \rho'(x^\bullet)\right| < \frac{|\theta|}{M}\varepsilon.$$

Como $h_\lambda = \frac{\widehat{x}}{\lambda}$ tomamos λ tal que $\frac{\alpha}{\lambda} < \delta$, así,

$$|h_\lambda| = \frac{|\widehat{x}|}{\lambda} < \delta.$$

En este caso obtenemos que, para todo $\widehat{x} \in K$,

$$|R(\phi)(\widehat{x}) - \widehat{x}| = \frac{|\widehat{x}|}{|\theta|}\left|\frac{\rho(x^\bullet + y) - \rho(x^\bullet)}{y} - \rho'(x^\bullet)\right| < \frac{|\theta|}{M}\frac{|\widehat{x}|}{|\theta|}\varepsilon \leq \varepsilon;$$

de lo que concluimos que

$$|R(\phi)(\widehat{x}) - \widehat{x}| \leq \varepsilon \ \ \forall\, \widehat{x} \in K.$$

□

Teorema 4.2 (Universalidad, Caso $L \geq 2$)

Sean $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compacto y $\rho : \mathbb{R} \to \mathbb{R}$ una función discriminante tal que es diferenciable y no constante en un abierto. Entonces $MLP(\rho, d, L)$ es universal para $L \in \mathbb{N}$ con $L \geq 2$, es decir, $MLP(\rho, d, L)$ es denso en $C(K)$.

Demostración. Considere $f \in C(K)$ y supongamos que $(\phi_n)_{n\in\mathbb{N}}$ es una sucesión de redes neuronales tal que

$$\lim_{n\to\infty} \phi_n = f$$

donde $\phi_n \in MLP(\rho, d, L)$ para todo $n \in \mathbb{N}$.

Por el teorema de universalidad para el caso $L = 2$, existe una red neuronal φ tal que $R(\varphi^n) \in MLP(\rho, d, 2)$ satisface que

$$\|R(\varphi^n) - f\| < \frac{1}{n} \ \ \forall n \in \mathbb{N}. \tag{1}$$

Como f es continua, entonces $f(K)$ es compacto (cerrado y acotado).

Sea $\tilde{K} = \overline{B(0,r)} \subseteq \mathbb{R}$ tal que $r > L$ y $f(K) \subseteq B(0, r - L)$.

Usamos la proposición 4.4 para encontrar redes neuronales ψ^n tales que la profundidad de $R(\psi^n)$ es 2 y

$$|R(\psi^n)(y) - y| < \frac{1}{n} \ \ \forall y \in \tilde{K}. \tag{2}$$

Definimos

$$\phi_n = \underbrace{\psi^n \bullet \psi^n \bullet \cdots \bullet \psi^n \bullet \varphi^n}_{L-2 \ veces}$$

donde ϕ_n tiene profundidad L.

Recordemos que

$$R(\phi_n) = R(\psi^n) \circ R(\psi^n) \circ \cdots \circ R(\psi^n) \circ R(\varphi^n).$$

Así,

$$\begin{aligned} |R(\phi_n)(\widehat{x}) - f(\widehat{x})| \leq |R(\psi^n[R(\psi^n) \circ \cdots \circ R(\varphi^n)(\widehat{x})] - \overbrace{R(\psi^n) \circ \cdots \circ R(\varphi^n)(\widehat{x}))}^{n-1 \ veces}| \\ + |R(\psi^n)[\overbrace{R(\psi^n) \circ \cdots \circ R(\varphi^n)(\widehat{x})}^{n-2 \ veces}] - \overbrace{R(\psi^n) \circ \cdots \circ R(\varphi^n)(\widehat{x})]}^{n-2 \ veces}| \\ + \ \cdots \ + |R(\phi^n)[R(\varphi^n)(\widehat{x})] - R(\varphi^n(\widehat{x}))| + |R(\varphi^n)(\widehat{x}) - f(\widehat{x})|. \end{aligned}$$

Luego, por la ecuación 2,

$$|R(\phi_n)(\widehat{x}) - f(\widehat{x})| < \overbrace{\frac{1}{n} + \frac{1}{n} + \cdots + \frac{1}{n}}^{L-2 \ veces} + |R(\varphi^n)(\widehat{x}) - f(\widehat{x})|$$

y por la ecuación 1

$$|R(\phi_n)(\widehat{x}) - f(\widehat{x})| < (L-1)\frac{1}{n}.$$

De lo que concluimos que

$$\lim_{n\to\infty} R(\phi_n) = f.$$

□

Esto último demuestra el teorema, sin embargo, para poder emplear la ecuación 2 usamos que

$$\underbrace{R(\psi^n) \circ \cdots \circ R(\varphi^n)(\widehat{x})}_{m-veces} \in \tilde{K},\ \forall \widehat{x} \in K,\ \forall m \leq L-2.$$

Veamos porqué pasa esto:

Notemos que por la ecuación 1

$$|R(\varphi^n)(\widehat{x}) - f(\widehat{x})| < \frac{1}{n},$$

por lo que la distancia

$$dist(R(\varphi^n)(x), F(K)) \leq \frac{1}{n}.$$

Luego, por la ecuación 2,

$$|R(\psi^n) \circ R(\varphi^n)(\widehat{x}) - R(\varphi^n(\widehat{x}))| < \frac{1}{n},$$

por lo que

$$dist(R(\psi^n) \circ \varphi(\varphi^n)(\widehat{x}), (R \circ \varphi^n)(K)) \leq \frac{1}{n}.$$

Así,

$$dist(R(\psi^n) \circ R(\varphi^n)(\widehat{x}), F(K)) \leq \frac{2}{n}.$$

De esta forma obtenemos que

$$dist[R(\psi^n) \circ \cdots \circ R(\varphi^n)(\widehat{x}), f(K)] \leq \frac{L}{n}\ \ \forall\, \widehat{x} \in K.$$

Esto implica que $R(\psi^n) \circ \cdots \circ R(\varphi^n)(\widehat{x}) \in \tilde{K}$ para todo $\widehat{x} \in K$.

4.4. Reaproximación de Diccionarios

Definición 4.9

Sean H un espacio normado y $(A_n)_{n\in\mathbb{N}}$ una colección anidada de subconjuntos, es decir, $A_n \subseteq A_{n+1}$ para todo $n \in \mathbb{N}$. Dado $C \subseteq H$, definimos

$$\sigma(A_n, C) := \sup_{f\in C} \inf_{g\in A_n} \|f \cdot g\|.$$

Sea $h : \mathbb{N} \to \mathbb{R}^+$. Si se cumple que, cuando $n \to \infty$

$$\sigma(A_n, C) = \mathcal{O}(h(n))$$

entonces decimos que $(A_n)_{n\in\mathbb{N}}$ tiene una tasa de aproximación h para C.

Recordemos que la expresión $\sigma(A_n, C) = \mathcal{O}(h(n))$ significa que existe una constante α tal que

$$\sigma(A_n, C) \leq \alpha \cdot h(n)\ \ \forall\, n.$$

Definición 4.10

Dadas funciones f y g, decimos que

$$f \lesssim g$$

si existe una constante α tal que $f \leq \alpha \cdot g$ puntualmente.

De lo anterior tenemos que

$$\sigma(A_n, C) = \mathcal{O}(h(n)) \Leftrightarrow \sigma(A_n, C) \lesssim h(n).$$

Definición 4.11 (Diccionario)

Sea $D = (f_i)_{i=1}^{\infty} \subseteq H$ una sucesión (a la cual llamaremos Diccionario).
Definimos los espacios

$$A_N := \left\{ \sum_{i=1}^{\infty} C_i f_i \ : \ C_i \in \mathbb{R} \text{ y } \|C\|_0 \leq N \right\},$$

en donde $\|C\|_0 = \#\{i \in \mathbb{N} \ : \ C_i \neq 0\}$.
Es claro que $A_N \subseteq A_{N+1}$ para todo N.

Teorema 4.3

Sean $d \in \mathbb{N}$, $H \subseteq \left\{ f : K \subseteq \mathbb{R}^d \to \mathbb{R} \right\}$ un espacio normado, $\rho : \mathbb{R} \to \mathbb{R}$ y $D := (f_i)_{i=1}^{\infty} \subseteq H$ un diccionario. Supongamos que existen $L, C \in \mathbb{N}$ tales que, para todo $i \in \mathbb{N}$ y para todo $\varepsilon > 0$, existe una red neuronal ϕ_i^{ε} tal que:

$$L(\phi_i^{\varepsilon}) = L,$$

$$M(\phi_i^{\varepsilon}) \leq C,$$

$$\|R(\phi_i^{\varepsilon}) - f_i\|_H \leq \varepsilon.$$

Para todo $C \subseteq H$, definimos A_N como en la definición 4.11 y definimos

$$B_N := \{R(\phi) \ : \ \phi \text{ es una red neuronal, } L(\phi) = L,\ M(\phi) \leq N\},$$

entonces, para todo $\zeta \subseteq H$,

$$\sigma(B_{CN}, \zeta) \leq \sigma(A_N, \zeta).$$

Demostración. Sea $a \in A_N$, entonces

$$a = \sum_{j=1}^{n} C_{i(j)} f_{i(j)}.$$

Sea $\varepsilon > 0$, entonces por hipótesis existen redes neuronales $(\phi)_{j=1}^{n}$ tales que:

- $L(\phi_j) = L$,
- $M(\phi_j) \leq C$,
- $\|R(\phi_j) - f_{i(j)}\| \leq \dfrac{\varepsilon}{N \cdot \text{máx}(\{|C_{i(j)}|\})}$.

Definimos

$$\phi^{(a,\varepsilon)} = \phi^c \cdot P[\phi_1, \phi_2, \cdots, \phi_N],$$

en donde $\phi^c = ([C_{i(1)}, C_{i(2)}, \cdots, C_{i(N)}], 0)$, entonces

$$R(\phi^{(a,\varepsilon)}) = \sum_{j=1}^{n} C_{i(j)} R(\phi_j).$$

Por lo que

$$\begin{aligned}\|R(\phi^{(a,\varepsilon)}) - a\| &= \|\sum_{j=1}^{n} C_{i(j)} R(\phi_j) - \sum_{j=1}^{n} C_{i(j)} f_{i(j)}\| \\ &\leq \sum_{j=1}^{n} |C_{i(j)}| \|R(\phi_j) - f_{i(j)}\| \\ &\leq N \cdot \text{máx}(\{|C_{i(j)}|\}) \frac{\varepsilon}{N \cdot \text{máx}(\{|C_{i(j)}|\})} \\ &\leq \varepsilon.\end{aligned}$$

Así, para cualesquiera $\varepsilon > 0$ y $a \in A_N$, existe $\phi^{(a,\varepsilon)}$ tal que:

- $L(\phi^{(a,\varepsilon)} = L$,
- $M(\phi^{(a,\varepsilon)} \leq C \cdot N$,
- $\|R(\phi^{(a,\varepsilon)}) - a\| \leq \varepsilon$.

En otras palabras, para cualesquiera $\varepsilon > 0$ y $a \in A_N$, existe $R(\phi^{(a,\varepsilon)}) \in B_{CN}$ tal que

$$\|R(\phi^{(a,\varepsilon)}) - a\| \leq \varepsilon. \tag{1}$$

Recordemos que:

$$\begin{aligned}\sigma(A_N, \zeta) &= \sup_{f \in \zeta} \inf_{a \in A_N} \|f - a\|, \\ \sigma(B_{CN}, \zeta) &= \sup_{f \in \zeta} \inf_{b \in B_{CN}} \|f - b\|.\end{aligned}$$

Sean $f \in \zeta$ y $a \in A_N$. Por la ecuación 1 encontramos $b_\varepsilon \in B_{CN}$ tal que $\|a - b_\varepsilon\| \leq \varepsilon$.
Así, para todo $a \in A_N$,

$$\begin{aligned}\inf_{b \in B_{CN}} \|f - b\| \leq \|f - b_\varepsilon\| &= \|f - b_\varepsilon + a - a\| \\ &\leq \|f - a\| + \|a - b_\varepsilon\| \\ &\leq \|f - a\| + \varepsilon.\end{aligned}$$

Por lo que

$$\inf_{b \in B_{CN}} \|f - b\| \leq \inf_{a \in A_N} \|f - a\| + \varepsilon \ \ \forall \varepsilon > 0.$$

Luego, haciendo tender $\varepsilon \to 0$, obtenemos

$$\inf_{b \in B_{CN}} \|f - b\| \leq \inf_{a \in A_N} \|f - a\| \ \ \forall\, f \in \zeta.$$

Tomando el supremo en la desigualdad, obtenemos que

$$\sigma(B_{CN}, \zeta) = \sup_{f \in \zeta} \inf_{b \in B_{CN}} \|f - b\| \leq \sup_{f \in \zeta} \inf_{a \in A_N} \|f - a\| = \sigma(A_N, \zeta),$$

de lo que concluimos que

$$\sigma(B_{CN}, \zeta) \leq \sigma(A_N, \zeta).$$

□

4.5. Aproximación de Funciones Suaves

Definición 4.12 (B-Spline)

Definimos el B-Spline de orden $k \in \mathbb{N}$ de la siguiente manera:

$$\mathcal{N}_k(x) := \frac{1}{(k-1)!} \sum_{s=0}^{k} (-1)^s \binom{k}{s} (x-s)_+^{k-1},$$

en donde $x \in \mathbb{R}$, $0^0 = 0$, $0! = 1$ y $t_+ = \begin{cases} y & \text{si } y \geq 0, \\ 0 & \text{si } y \leq 0. \end{cases}$

Definición 4.13 (B-Spline Multivariado)

Dados $t \in \mathbb{R}$ y $l \in \mathbb{N}$, definimos

$$\mathcal{N}_{l,t,k}(x) := \mathcal{N}_k(2^l(x-t)),$$

donde $x \in \mathbb{R}$.

Dados $d, l \in \mathbb{N}$ y $t = (t_1, t_2, \cdots, t_d) \in \mathbb{R}^d$, definimos el B-Spline multivariado como

$$\mathcal{N}^d_{l,t,k}(\widehat{x}) := \prod_{j=1}^{d} \mathcal{N}_{l,t_j,k}(x_j),$$

donde $\widehat{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$.

Definición 4.14

Definimos

$$\mathcal{B}^k := \left\{ \mathcal{N}^d_{l,t_l,k} \ : \ l \in \mathbb{N}, t_l \in 2^{-l}\mathbb{Z}^d \right\}.$$

Teorema 4.4

Sean $d, k \in \mathbb{N}$ y $s \in (0, k)$. Entonces existe $c > 0$ tal que para toda $f \in C^s([0,1]^d)$ tenemos que, para toda $\delta > 0$ y todo $N \in \mathbb{N}$, existen $C_i \in \mathbb{R}$ y $B_i \in \mathcal{B}^k$ tales que:

- $|C_i| \leq C\|f\|$,
- $f - \sum_{i=1}^{N} C_i B_i \lesssim N^{\frac{\delta - s}{d}} \|f\|_{C^s([0,1]^d)}$,

en donde $i \in \{1, 2, \cdots N\}$ y

$$\|f\|_{C^s([0,1]^d)} := \sum_{J_i + \cdots + J_d \leq s} \| \frac{\partial^{J_i}}{\partial x_i} \cdots \frac{\partial^{J_d}}{\partial x_d} f \|.$$

Definición 4.15 (Sigmoidal)

Una función $\rho : \mathbb{R} \to \mathbb{R}$ es llamada sigmoidal de orden $q \in \mathbb{N}$ si cumple:

1. $\rho \in C^{q-1}(\mathbb{R})$,
2. $\frac{\rho(x)}{x^q} \to 0$ como $x \to -\infty$,
3. $\frac{\rho(x)}{x^q} \to 1$ como $x \to \infty$,
4. $|\rho(x)| \lesssim (1 + |x|)^q \ \ \forall\, x \in \mathbb{R}$.

Definición 4.16 (Función Sigmoidal Estandar)

El ejemplo estándar de función sigmoidal de orden q es $\rho_s : \mathbb{R} \to \mathbb{R}$ tal que

$$\rho_s(x) = x_+^q.$$

Frecuentemente usaremos la función de activación $\rho_s = x_+^q$ para facilitar las demostraciones, aunque éstas son válidas para cualquier función sigmoidal de orden q.

Lema 4.1

Dados $\varepsilon > 0$ y $p \in \mathbb{N}$, existe una red neuronal ϕ tal que:

- $L(\phi) = \lceil \text{máx}\{\log_q(p-1), 0\} \rceil + 1$,
- $\|R(\phi) - x_+^{p-1}\|_{[-z,z]} \leq \varepsilon$,

para cualquier intervalo de la forma $[-z, z]$ donde ϕ depende de z.

Demostración. Sea $\lambda := \lceil \text{máx}(log_q(p-1), 0) \rceil$, se tiene que

$$\underbrace{\rho_s \circ \rho_s \circ \cdots \circ \rho_s(x)}_{\lambda - veces} = x_+^{q^\lambda} = x_+^b,$$

donde $b = q^\lambda$.

Así,

$$b = q^\lambda \geq q^{\log_q(p-1)} \geq p - 1.$$

Denotamos por $g(x) = x_+^b$. Luego

$$\rho_s \circ \rho_s \circ \cdots \circ \rho_s = g.$$

Tomamos, dada $\delta \in \mathbb{R}$,

- $T_1^\delta(x) = \begin{pmatrix} x + \delta \\ x \end{pmatrix} \ \ \forall x \in \mathbb{R}$,
- $T_l^\delta \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \ \ \forall x, y \in \mathbb{R},\ \forall l \in \{2, 3, \cdots, \lambda - 1\}$,
- $T_{\lambda+1}^\delta \begin{pmatrix} x \\ y \end{pmatrix} = \frac{x-y}{\delta} \ \ \forall x, y \in \mathbb{R}$.

Sea $\phi^\delta = (T_1^\delta, T_2^\delta, \cdots, T_{\lambda+1}^\delta)$. Tenemos que

$$\begin{aligned} R(\phi^\delta)(x) &= T_{\lambda+1}^\delta \circ \rho_s \circ T_{\lambda-1}^\delta \circ \rho_s \circ \cdots \circ \rho_s \circ T_1(x) \\ &= T_{\lambda+1}^\delta \circ \rho_s \circ \cdots \circ T_2 \begin{pmatrix} \rho_s(x+\delta) \\ \rho_s(x) \end{pmatrix} \\ &= T_{\lambda+1}^\delta \begin{pmatrix} \rho_s \circ \rho_s \circ \cdots \circ \rho_s(x+\delta) \\ \rho_5 \circ \rho_s \circ \cdots \circ \rho_s(\ \ x\ \ ) \end{pmatrix} \\ &= T_\lambda^\delta \begin{pmatrix} g(x+\delta) \\ g(x) \end{pmatrix} \\ &= \frac{g(x+\delta) - g(x)}{\delta}, \end{aligned}$$

esto implica que, cuando $\delta \to 0$, $|R(\phi^\delta)(x) - g'(x)| \to 0$ uniformemente en $[-z, z]$.

Concluimos que dado $\varepsilon > 0$ existe una red neuronal ϕ^δ tal que

$$\|R(\phi^\delta) - g'\|_{[-z,z]} < \varepsilon \ \text{ y } \ L(\phi^\delta) = \lambda + 1.$$

El siguiente paso es aproximar $g''(x) = b(b-1)x_+^{b-2}$ usando redes neuronales.

Tomamos $h \in (0, 1)$ suficientemente chico tal que

$$\left| \frac{g(x+h) - g(x) - h \cdot g'(x) - \frac{h^2}{3} g''(x)}{h^2} \right| < \varepsilon \ \ \forall\, x \in [-z, z],$$

(usamos el Teorema de Taylor: $g(x+h) = g(x) + h \cdot g'(x) + \frac{h^2}{2} g''(x) + \mathcal{O}(h^3)$).

Encontramos $\delta > 0$ tal que, de manera uniforme,

$$|R(\phi^\delta)(x) - g'(x)| < h^2 \cdot \varepsilon.$$

Así, de manera uniforme,

$$\begin{aligned} &\left| \frac{g(x+h) - g(x) - h \cdot R(\phi^d)(x) - \frac{h^2}{2} g''(x)}{h^2} \right| \\ &\leq \left| \frac{g'(x) - R(\phi^d)(x)}{h} \right| + \left| \frac{g(x+h) - g(x) - h \cdot g'(x) - \frac{h^2}{2} g''(x)}{h^2} \right| \\ &< 2 \cdot \varepsilon. \end{aligned} \tag{1}$$

Recordemos que

$$\phi^d = (T_1^\delta, \cdots, T_{\lambda+1}^\delta),$$

tomemos ahora

$$\phi^{\varepsilon,\delta} = (T_1^{\varepsilon,\delta}, T_2^{\varepsilon,\delta}, \cdots, T_{\lambda+1}^{\varepsilon,\delta}),$$

en donde:

- $T_1^{\varepsilon,\delta}(x) = \begin{pmatrix} x+h \\ x \\ T_1^\delta(x) \end{pmatrix}$,

- $T_l^{\varepsilon,\delta}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ T_l^{\delta}(y_3) \end{pmatrix} \quad \forall l \in \{1, 2, ..., \lambda\},$
- $T_{\lambda+1}^{\varepsilon,\delta}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \dfrac{y_1 - y_2 - h \cdot T_{\lambda+1}^{\delta}(y_3)}{h^2}.$

Se tiene que

$$\begin{aligned} R(\phi^{\varepsilon,\delta})(x) &= T_{\lambda+1} \circ \rho_s \circ \cdots \circ \rho_s \circ T_1(x) \\ &= T_{\lambda+1}^{\varepsilon,\delta}\begin{pmatrix} \rho_s \circ \cdots \circ \rho_s(x+h) \\ \rho_s \circ \cdots \circ \rho_s(x) \\ \rho_s \circ T_\lambda^\delta \circ \cdots \circ \rho_s \circ T_1^\delta(x) \end{pmatrix} \\ &= T_{\lambda+1}^{\varepsilon,\delta}\begin{pmatrix} g(x+h) \\ g(x) \\ \rho_s \circ T_\lambda^\delta \circ \cdots \circ \rho_s \circ T_1^\delta(x) \end{pmatrix} \\ &= \frac{g(x+h) - g(x) - T_{\lambda+1}^\delta \circ \cdots \circ T_1^\delta(x)}{h^2} \\ &= \frac{g(x+h) - g(x) - h \cdot R(\phi^\delta)}{h^2}, \end{aligned}$$

así,

$$R(\phi^{\varepsilon,\delta})(x) = \frac{g(x+h) - g(x) - h \cdot R(\phi^\delta)}{h^2}$$

y de la ecuación (1) obtenemos que

$$|R(\phi^{\varepsilon,\delta})(x) - \frac{1}{2}g''(x)| < 2 \cdot \varepsilon$$

Por lo tanto converge uniformemente.

De la misma forma se demuestra que para toda n y para toda $\epsilon > 0$ existe una constante $c \neq 0$ y una red neuronal ϕ con $L(\phi) = \lambda + 1$ tal que, de manera uniforme,

$$|R(\phi)(x) - c \cdot \frac{d^n}{dx^n}g(x)| < \epsilon\ ;$$

siempre y cuando se pueda aplicar el teorema de Taylor.

Sea m tal que

$$\frac{d^m}{dx^m}g(x) = \alpha \cdot x_+^{p-1}$$

en donde α es una constante. Por lo anterior, existe una red neuronal ψ con $L(\psi) = \lambda + 1$ tal que, de manera uniforme,

$$|R(\psi)(x) - c \cdot x_+^{p-1}| < |c| \cdot \epsilon$$

en donde c es una constante no cero.

Sean

$$\begin{aligned} \psi &= (\tilde{T}_1, \tilde{T}_2, \cdots, \tilde{T}_{\lambda+1}), \\ \phi &= (\tilde{T}_1, \tilde{T}_2, \cdots, \tilde{T}_\lambda, \frac{1}{c} \cdot \tilde{T}_{\lambda+1}), \end{aligned}$$

se tiene que

$$R(\phi) = \frac{1}{c}R(\psi).$$

Así, de manera uniforme,

$$\begin{aligned} |R(\phi)(x) - x_+^{p-1}| &= |\frac{1}{c}R(\psi)(x) - x_+^{p-1}| \\ &= \frac{1}{|c|}|R(\psi)(x) - c \cdot x_+^{p-1}| \\ &< \frac{1}{|c|} \cdot |c| \cdot \epsilon = \epsilon. \end{aligned}$$

Hemos demostrado que, para toda $\epsilon > 0$, existe ϕ tal que, de manera uniforme,

$$|R(\phi)(x) - x_+^{p-1}| < \epsilon.$$

Lo que implica el resultado del lema 4.1. □

Corolario 4.1

Sea $t \in \mathbb{R}$, $p \in \mathbb{N}$, $z > 0$. Si que ρ es una función sigmoidal de orden q, entonces existe una red neuronal ϕ^ϵ tal que:

$$\begin{aligned} &\|R(\phi) - \mathcal{N}_{l,t,\rho}\|_{[-z,z]} < \epsilon, \\ &L(\phi) = \lceil max\{log_q(p-1), 0\} \rceil + 1. \end{aligned}$$

Demostración. Por el lema 4.1, dada $s > 0$ y $A > 0$, existe una red neuronal ϕ^s tal que:

$$\begin{aligned} &\|R(\phi^s) - x_+^{p-1}\|_{[-A,A]} < s, \\ &L(\phi^s) = \lceil max\{log_q(p-1), 0\} \rceil + 1 = L. \end{aligned}$$

Supongamos que

$$\phi^s = (T_1, T_2, \cdots, T_L)$$

y sean, para toda $l \in \{0, 1, \cdots, p\}$,

$$T_{1,l}(x) = T_1(2^l(x-t) - l).$$

Definimos las transformaciones afines:

- $V_1(x) = \begin{pmatrix} T_{1,0}(x) \\ T_{1,1}(x) \\ \vdots \\ T_{1,p}(x) \end{pmatrix}$,
- $V_j \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \end{pmatrix} = \begin{pmatrix} T_j(x_0) \\ T_j(x_1) \\ \vdots \\ T_j(x_p) \end{pmatrix} \ \forall j \in \{2, 3, \cdots, L-1\}$,
- $V_j \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \end{pmatrix} = \frac{1}{(p-1)!} \sum_{l=0}^{p} (-1)^l \binom{p}{l} T_L(x_l)$.

Definimos también

$$\psi^s = (V_1, V_2, \cdots, V_L),$$

de manera que

$$R(\psi^s) = \frac{1}{(p-1)!}\sum_{i=0}^{p}(-1)^i \binom{p}{i} R(\psi^s)(2^i(x-t)-i).$$

Obtenemos que

$$|R(\psi^s)(x) - \mathcal{N}_{i,t,p}(x)| = \left|\frac{1}{(p-1)!}\sum_{i=0}^{p}(-1)^i \binom{p}{i}\left[R(\psi^5)(2^i(x-t)-i) - (2^i(x-t)-i)_+^{p-1}\right]\right|,$$

entonces

$$\begin{aligned}|R(\psi^s(x)) - \mathcal{N}_{l,t,p}(x)| &\leq \frac{1}{(p-1)!}\sum_i \binom{p}{i}\left|R(\phi^s)(2^l(x-t)-i) - (2^l(x-t)-i)_+^{p-1}\right| \\ &\leq \left[\frac{1}{(p-1)!}\sum_i \binom{p}{i}\right]\cdot s.\end{aligned}$$

Tomamos

$$s = \frac{1}{\frac{1}{(p-1)!}\sum_i \binom{p}{i}}\cdot\epsilon$$

y elegimos A adecuadamente para que finalmente obtengamos

$$\|R(\psi^s) - \mathcal{N}_{l,t,p}(x)\|_{[-z,z]} < \epsilon.$$

□

Corolario 4.2 (De la demostración del Lema 4.1)

Sean $K \subset \mathbb{R}^2$ compacto y $\epsilon > 0$. Si ρ es una función sigmoidal de orden $q \geq 2$, entonces existe una red neuronal ϕ_ϵ con $L(\phi_\epsilon) = 2$ tal que

$$|\phi_\epsilon(x) - (x_1 + x_2)_+^2| < \epsilon \ \ \forall\, x = \begin{pmatrix}x_1\\x_2\end{pmatrix} \in K$$

y existe otra red neuronal ψ_ϵ con $L(\psi_\epsilon) = 2$ tal que

$$|\psi_\epsilon(x) - x_1x_2| < \epsilon \ \ \forall\, x = \begin{pmatrix}x_1\\x_2\end{pmatrix} \in K.$$

Corolario 4.3

Dado $K \subset \mathbb{R}^d$ y $\epsilon > 0$, existe una red neuronal $\phi_\epsilon^{mult.d}$ tal que

$$|\phi_\epsilon^{mult.d}(x) - x_1x_2\cdots x_d| < \epsilon\ \forall x \in K,$$

donde

$$L(\phi_\epsilon^{mult.d}) = \lceil log_2(d)\rceil + 1.$$

Demostración. Haremos la demostración para el caso en el que $d = 2^\alpha$ para alguna $\alpha \in \mathbb{N}$.

Por el corolario 4.2, para cada ε existe ψ_ε tal que, de manera uniforme,

$$\left|\psi_\varepsilon\left(\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right) - y_1 y_2\right| < \varepsilon.$$

Luego, definimos

$$\phi_\varepsilon^{mult,d} = \underbrace{FP(\psi_\varepsilon, \psi_\varepsilon)}_{2^{\alpha-(\alpha-1)} \text{ - veces}} \bullet FP(\psi_\varepsilon, \psi_\varepsilon, \psi_\varepsilon, \psi_\varepsilon) \bullet \cdots \bullet \underbrace{FP(\psi_\varepsilon, ..., \psi_\varepsilon)}_{2^{\alpha-2} \text{ - veces}} \bullet \underbrace{FP(\psi_\varepsilon, \psi_\varepsilon, ..., \psi_\varepsilon)}_{2^{\alpha-1} \text{ - veces}}$$

en donde

$$FP(\psi_\varepsilon, \psi_\varepsilon, ..., \psi_\varepsilon)\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} \psi_\varepsilon(z_1) \\ \psi_\varepsilon(z_2) \\ \vdots \\ \psi_\varepsilon(z_n) \end{pmatrix}.$$

Visualicemos de forma gráfica lo que está sucediendo:
en donde $w_1 \sim x_1, ..., x_{\frac{d}{2}}$ y $w_2 \sim x_{\frac{d}{2}+1}, ..., x_d$.

Es fácil ver que, de manera uniforme,

$$\lim_{\varepsilon \to 0} \left|\Phi_\varepsilon^{mult,d}(x) - x_1 x_2, ..., x_d\right| = 0.$$

Claramente la profundidad de $\Phi_\varepsilon^{mult,d}$ es α.
En el caso de que d no sea de la forma $d = 2^\alpha$, hacemos algo similar a lo que se muestra en el siguiente ejemplo con $d = 7$:
Esto demuestra el resultado.

□

Teorema 4.5

Sean $d, p \in \mathbb{N}$, $q \geq 2$, $z > 0$ y $\varepsilon > 0$. Dada $f \in B^p$, existe una red neuronal ϕ_ε tal que

$$\|f - R(\phi_\varepsilon)\|_{[-z,z]} d \leq \varepsilon \tag{0}$$

en donde la función de activación es ρ_s de orden $q \geq 2$ y

$$L(\phi_\varepsilon) = \lceil log_2(d) \rceil + \max\left\{\lceil log_q(p-1) \rceil, 0\right\} + 1.$$

Ademas existe una constante C tal que

$$M(\phi_\varepsilon) \leq C;$$

donde C no depende de f.

Demostración. Sea $f(x) = \prod_{i=1}^d N_{i,t_i,p}(x_i)$, con $x = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$.
Usamos el corolario 4.1 para encontrar redes neuronales ψ_i^n tales que

$$\|R(\psi_i^n) - N_{i,t_i,p}\|_{[-z,z]} \leq \frac{1}{n}. \tag{1}$$

Sea $\phi^{mult,d}$ como en el corolario 4.3 tal que

$$\|R(\phi^{mult,d}_{\frac{1}{n}})(x) - x_1 x_2 \cdots x_d\|_{[-z,z]^d} \leq \frac{1}{n}. \tag{2}$$

Tomamos

$$\phi^n = \phi_{\frac{1}{n}^{mult,d}} \cdot FP(\psi_1^n, \psi_2^n, \cdots, \psi_d^n),$$

así,

$$\begin{aligned} R(\phi^n)(x) &= R(\phi^{mult,d}_{\frac{1}{n}}) \circ R(FP(\psi_1^n, \psi_2^n, \cdots, \psi_d^n))(x) \\ &= R(\phi^{mult,d}_{\frac{1}{n}}) \begin{pmatrix} R(\psi_1^n)(x_1) \\ \vdots \\ R(\psi_d^n)(x_d) \end{pmatrix}. \end{aligned}$$

Por la ecuación 2, tenemos que

$$\|R(\phi^n) - \prod_{i=1}^{d} \psi_i^n\| \to_{n\to\infty} 0. \tag{3}$$

Además, por la ecuación 1, tenemos que

$$\|\prod_{i=1}^{d} \psi_i^n - \prod_{i=1}^{d} N_{i,t_i,p}\| \to_{n\to\infty} 0. \tag{4}$$

De las ecuaciones 3 y 4 obtenemos que

$$\|R(\theta^n) - f\| \to_{n\to\infty} 0,$$

lo que demuestra la ecuación 0 para n suficientemente grande.

Además, de los corolarios 4.1 y 4.3, obtenemos que

$$L(\phi^n) = \lceil log_2(d) \rceil + \text{máx}\{\lceil log_q(p-1) \rceil, 0\} + 1$$

y también que existe una constante C tal que

$$M(\phi^n) \leq C \quad \forall f \in B^p.$$

□

Teorema 4.6

Sean $d, p, s \in \mathbb{N}$, $s > \delta > 0$ y $\rho : \mathbb{R} \to \mathbb{R}$ una función sigmoidal de orden $q \geq 0$. Entonces existe $\tilde{\tilde{C}} > 0$ tal que, para cualesquiera $f \in C^s([0,1]^d)$ y $\varepsilon \in (0, \frac{1}{2})$, existe una red neuronal ϕ tal que:

- $\|f - R(\phi)\| \leq \varepsilon$,
- $M(\phi) \leq \tilde{\tilde{C}} \left(\frac{\varepsilon}{\|f\|_{C^s}}\right)^{\frac{d}{s-\delta}}$,
- $L(\phi) = \lceil log_2(d) \rceil + \text{máx}\{\lceil log_q(p-1) \rceil, 0\} + 1$.

Demostración. Sean $\delta \in (0, s)$ y $N \in \mathbb{N}$ tales que

$$CN^{\frac{\delta - s}{d}} \|f\|_{C^s([0,1]^d)} < \frac{\varepsilon}{2}$$

en donde C es como en el teorema 4.4, con $k = P$.

Luego

$$\|f - \sum_{i=1}^{N} CiBi\| < CN^{\frac{\delta - s}{d}} \|f\|_{C^s([0,1]^d)}.$$

Se tiene que

$$\left(\frac{C\|f\|_{C^s}}{\varepsilon/2}\right) < N^{\frac{s-\delta}{d}},$$

de lo que se obtiene que

$$\left(\frac{C\|f\|_{C^s}}{\varepsilon/2}\right)^{\frac{d}{s-\delta}} < N.$$

Así, por el teorema 4.4, se tiene que

$$\sigma(A_N, |f|) \leq \frac{\varepsilon}{2}$$

con

$$A_N = \left\{ \sum_{i=1}^{\infty} CiBi : Ci \in \mathbb{R}, \|C\|_0 \leq N \right\},$$

en donde $\|C\|_0 = \#\{i \in \mathbb{N} : C_i \neq 0\}$ y $B_l \in B^P$ (ver la definición 4.11 con $D = B^P$).

Se sigue de los Teorema 4.4 y 4.5 que existe una constante $\tilde{C}$ tal que

$$\sigma\left(B_{\tilde{C}N}, |f|\right) \leq \sigma\left(A_N, |f|\right) \leq \frac{\varepsilon}{2},$$

con

$$B_{\tilde{C}N} = \{R(\phi) : L(\phi) = L \text{ y } M(\phi) \leq \tilde{C}N\}.$$

Así,

$$\sigma\left(B_{\tilde{C}N}, |f|\right) \leq \frac{\varepsilon}{2},$$

lo que implica que existe ϕ con $L(\phi) = L$ y $M(\phi) \leq \tilde{C}N$ tal que

$$\|f - R(\phi)\| < \varepsilon.$$

En este caso podemos tomar cualquier N tal que

$$\left(\frac{C\|f\|_{C^s}}{\varepsilon/2}\right)^{\frac{d}{s-\delta}} < N,$$

por ejemplo

$$N = \left\lceil \left(\frac{C\|f\|_{C^s}}{\varepsilon/2}\right)^{\frac{d}{s-\delta}} \right\rceil,$$

entonces existe otra constante $\tilde{\tilde{C}}$ tal que

$$M(\phi) \leq \tilde{\tilde{C}} \left(\frac{\varepsilon}{\|f\|_{C^s}}\right)^{\frac{d}{s-\delta}}.$$

□

5 | Análisis de Datos con Redes Neuronales

5.1. Algoritmos de Aprendizaje

Dentro del Machine Learning, ¿Qué se considera un algoritmo de aprendizaje?. Un algoritmo de aprendizaje es aque que es capaz de aprender a partir de los datos con los que se pretende trabajar. El principal problema a resolver con los algoritmos de aprendizaje involucra la optimización de algún modo. Matemáticamente hablando, optimizar es minimizar o maximizar alguna función $f(\widehat{x})$ llamada *función de objetivo*, esto se logra modificando el valor de $\widehat{x}$. Para el aprendizaje de una Red Neuronal, el problema será minimizar a $f(\widehat{x})$, nombrando ahora a $f(\widehat{x})$ la *función de costo*. Lo que se muestra en el capítulo actual es como minimizar tales funciones además de las demostraciones matemáticas correspondientes.

5.2. Ejemplo

Suponga que tenemos un rectángulo $R \subset \mathbb{R}^2$ y que a cada punto $\widehat{x} \in R$ se le asigna una etiqueta de color rojo o azul (en este ejemplo tendremos solamente diez puntos).

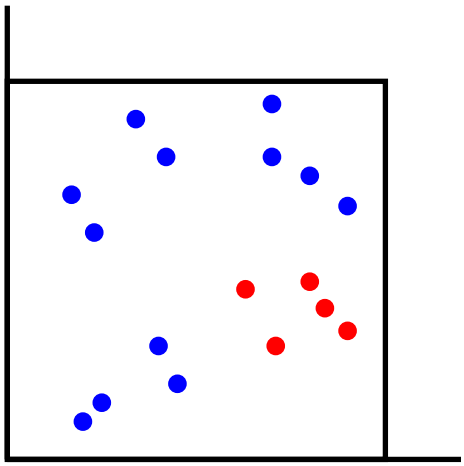


Figura 5.1

En principio no sabemos que puntos tienen asignada cual etiqueta, luego, mediante un muestreo obtenemos esa información para algunos puntos (ver figura 5.1).
Para los puntos de la imagen ya conocemos su etiqueta correspondiente.

Sea $R = [0,1] \times [0,1]$ y considere el conjunto de los puntos sobre los cuales se realizó el muestreo

$$\{\widehat{x}^{(i)} : \widehat{x}^{(i)} \in R, i = 1, ..., 10\}.$$

Sea $y : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ tal que

$$y(\widehat{x}) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{si la etiqueta de } x \text{ es de color rojo,} \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{si la etiqueta de } x \text{ es de color azul.} \end{cases}$$

No conocemos a la función y, pero quisiéramos encontrar una función (red neuronal) que la aproxime. Lo único que conocemos de la función y es su valor en los puntos $\widehat{x}^{(i)}$; a partir de esta información queremos obtener una red neuronal que aproxime a la función y de una forma "satisfactoria".

Planteamiento del Problema:
Queremos determinar, para cada punto del cuadrado R, si su etiqueta es de color rojo o azul usando redes neuronales.
Considere una función de activación sigmoidal $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ dada por

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

Podemos aproximar a la función y en los puntos conocidos $\widehat{x}^{(i)}$ con una red neuronal

$$F = R(T_1, ..., T_n),$$

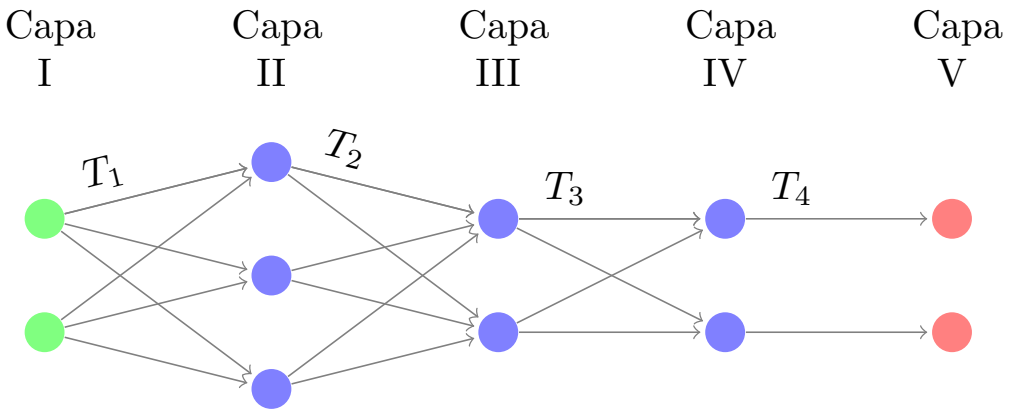
donde

$$F(\widehat{x}) = T_n \circ \sigma \circ T_{n-1} \circ \sigma \circ \cdots \circ \sigma \circ T_1(\widehat{x})$$

y definimos una función de costo como sigue:

$$cost(T_1, \ldots, T_n) := \frac{1}{10} \sum_{i=1}^{10} \frac{1}{2} \|y(\widehat{x}^{(i)}) - F(\widehat{x}^{(i)})\|^2.$$

Considere la siguiente red neuronal con 4 capas



donde:

- $T_1 : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$,
- $T_2 : \mathbb{R}^3 \longrightarrow \mathbb{R}^2$,
- $T_3 : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ y

- $T_4 : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ está dada por $T_4(\widehat{x}) = \widehat{x}$.

Así,

$$F = \sigma \circ T_3 \circ \sigma \circ T_2 \circ \sigma \circ T_1.$$

Para cada transformación afín T_j, con $j \in \{1, 2, 3\}$, existen matrices W_j y vectores b_j tales que

$$T_j(\widehat{z}) = W_j(\widehat{z}) + b_j,$$

donde $W_1 \in \mathcal{M}_{3\times 2}(\mathbb{R})$, $b_1 \in \mathbb{R}^3$, $W_2 \in \mathcal{M}_{2\times 3}(\mathbb{R})$, $b_2 \in \mathbb{R}^2$, $W_3 \in \mathcal{M}_{2\times 2}(\mathbb{R})$ y $b_3 \in \mathbb{R}^2$.

Tenemos que F depende W_1, W_2, W_3, b_1, b_2 y b_3, los cuales tienen 17 parámetros que deben elegirse de tal manera que se minimice la función de costo

$$cost(T_1, \ldots, T_4).$$

Para encontrar el mínimo de la función de costo normalmente se utilizan programas de cómputo; por ejemplo, en MATLAB se usa el 'optimization toolbox' sobre los parámetros de las transformaciones afines (específicamente se usa 'non-linear least-squares solver is q nonlin').

Sea $F : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ dada por

$$F(\widehat{x}) = \begin{pmatrix} F_1(\widehat{x}) \\ F_2(\widehat{x}) \end{pmatrix} \in \mathbb{R}^2$$

la red neuronal que obtenemos después de encontrar el mínimo de la función de costo.

Si $F_1(x) > F_2(x)$ interpretamos que la etiqueta de $\widehat{x}$ es de color azul y
si $F_1(x) \leq F_2(x)$ interpretamos que la etiqueta de $\widehat{x}$ es de color rojo.
Con esta condición obtenemos la siguiente gráfica:

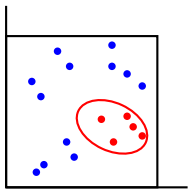


Figura 5.2

De elegir otra red, obtendríamos otro resultado; por ejemplo:

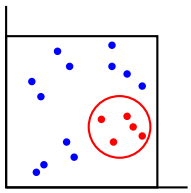


Figura 5.3

5.3. Esquema General

En 1847 el Barón Augustin-Louis Cauchy publicó un artículo llamado *Méthode générale pour la résolution des systèmes d'équations simultanées*. Motivado por la posibilidad de realizar cálculos astronómicos, los cuales son

bastante voluminosos, Cauchy propuso un método para resolver grandes conjuntos de ecuaciones simultáneas, lo que él buscaba era no centrarse en las ecuaciones diferenciales obtenidas para calcular las órbitas de los cuerpos celestes, estaba más interesado en resolver las ecuaciones algebráicas que representaban ese tipo de movimientos.

Durante la época de Cauchy, era común que se intentara resolver los sistemas de ecuaciones tratando de eliminar ecauciones para reducir el sistema lo más posible. Sin embargo, no siempre era posible eliminar ecuaciones y de ser el caso, a veces resultaban en una ecuación aún más complicada que el sistema entero. Para ello, dedujo que se podría resolver de manera iterativa un sistema de ecuaciones si se hacían pequeñas variaciones en cada una de las ecuaciones y se les restaba a la ecuación original. Veamos como funciona la idea de Cauchy en Machine Learning.

Antes de continuar con la explicación teórica, es buena idea tener un poco de visón sobre lo que estamos buscando. Observando la figura 2, tenemos un paraboloide que representa el aprendizaje de nuestra red, mientras más cerca estemos del punto B, mejor aprenderá la red ya que la función de costo se minimiza. Lo que se busca con el descenso del gradiente es llegar al fondo del paraboloide.

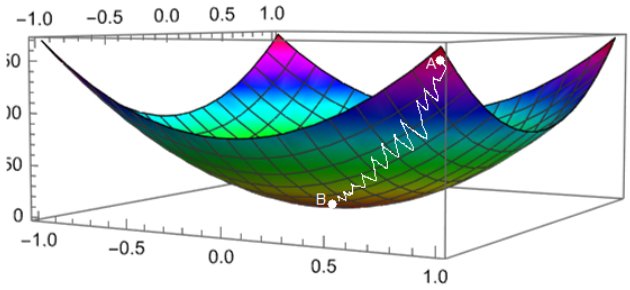


Figura 5.4: Recorrido del gradiente

Comenzando del punto A, el algoritmo de aprendizaje por descenso de gradiente itera pequeños saltos para llegar al mínimo global de la función. Al llegar al mínimo, el algoritmo debe parar y quedarse con los weights y bias que le dan el camino más corto en su aprendizaje. Es por ello que se llama descenso del gradiente, pues se busca hacer diferencias mínimas en la pendiente del aprendizaje para optimizarlo hasta llegar al costo más efectivo.

Matematicamente lo anterior se ve así:

Supongamos que tenemos N muestras de un fenómeno. Estas muestras se obtienen en puntos $(\widehat{x}_i)_{i=1}^N$ con resultados $(y(\widehat{x}_i))_{i=1}^n$. Construimos una red neuronal

$$\phi = (T_1, T_2, \cdots, T_n)$$

con

$$F = R(\phi).$$

La función de costo correspondiente es

$$cost(T_1, T_2, \cdots, T_n) = \frac{1}{n} \sum_{i=1}^{N} \frac{1}{2} \|y(\widehat{x}_i) - f(\widehat{x}_i)\|^2.$$

La idea es encontrar los parámetros $T_1, T_2, \cdots, T_n$ que determinan el mínimo de la función de costo.

5.4. Método de Descenso por el Gradiente (Convergencia)

Dada una función continuamente diferenciable

$$f : \mathbb{R}^n \to \mathbb{R}$$

queremos encontrar

$$\min_{\widehat{x}\in\mathbb{R}^n} f(\widehat{x}).$$

Definición 5.1

Una función de forzamiento es una función continua $\sigma : [0, \infty) \to (-\infty, 0]$ tal que:

- $\sigma(0) = 0$,
- $\sigma(t) > 0$ para todo $t > 0$ y
- si (t_n) es una sucesión en $\mathbb{R}$ tal que, cuando $n \to \infty$,

$$\sigma(t_n) \to 0,$$

entonces

$$t_n \to 0.$$

Por ejemplo, para $c > 0$, $\sigma(t) = ct$ y $\sigma(t) = ct^2$ son funciones de forzamiento.

Definición 5.2

Denotamos por $LC^1_L(\mathbb{R}^n)$ al conjunto de funciones $f \in C^1(\mathbb{R}^n)$ tales que, para cualesquiera $\widehat{x}, \widehat{y} \in \mathbb{R}^n$,

$$\|\nabla f(\widehat{x}) - \nabla f(\widehat{y})\| \leq L\|\widehat{x} - \widehat{y}\|.$$

Lema 5.1

Sea $f \in LC^1_L(\mathbb{R}^n)$, entonces, para cualesquiera $\widehat{x}, \widehat{y} \in \mathbb{R}^n$,

$$|f(\widehat{x} + \widehat{y}) - f(\widehat{x}) - \langle \nabla f(\widehat{x}), \widehat{y}\rangle| \leq \frac{L}{2}\|\widehat{y}\|^2.$$

Demostración. Escribimos

$$f(\widehat{x} + \widehat{y}) = f(\widehat{x}) + \int_0^1 \langle \nabla f(\widehat{x} + z\widehat{y}), \widehat{x}\rangle dz.$$

Tenemos que

$$\begin{aligned} f(\widehat{x} + \widehat{y}) &= f(\widehat{x}) + \int_0^1 \langle \nabla f(\widehat{x} + z\widehat{y}), \widehat{y}\rangle \, dz \\ &= f(\widehat{x}) + \int_0^1 \langle \nabla f(\widehat{x}), \widehat{y}\rangle \, dz + \int_0^1 \langle \nabla f(\widehat{x} + z\widehat{y}) - \nabla f(\widehat{x}), \widehat{y}\rangle \, dz \\ &= f(\widehat{x}) + \langle \nabla f(\widehat{x}), \widehat{y}\rangle + \int_0^1 \langle \nabla f(\widehat{x} + z\widehat{y}) - \nabla f(\widehat{x}), \widehat{y}\rangle \, dz. \end{aligned}$$

Ahora usamos el hecho de que $f \in LC^1_L(\mathbb{R}^n)$ para obtener que, para todo $z \geq 0$,

$$\|\nabla f(\widehat{x} + z\widehat{y}) - \nabla f(\widehat{x})\| \leq Lz\|\widehat{x}\|.$$

Así,

$$|f(\widehat{x} + \widehat{y}) - f(\widehat{x}) - \langle \nabla f(\widehat{x}), \widehat{y}\rangle| \leq \int_0^1 |\langle \nabla f(\widehat{x} + z\widehat{y}) - \nabla f(\widehat{x}), \widehat{y}\rangle|$$

$$\leq \int_0^1 \|\nabla f(\widehat{x}+z\widehat{y}) - \nabla f(\widehat{x})\|\|\widehat{y}\| \leq \int_0^1 Lz\|\widehat{y}\|^2 dz = \frac{L}{2}\|\widehat{y}\|^2.$$

□

Lema 5.2

Sean $(a_n)_{n\in\mathbb{N}}$ y $(\epsilon_n)_{n\in\mathbb{N}}$ sucesiones positivas tales que, para toda $n \in \mathbb{N}$,

$$0 \leq a_{n+1} \leq a_n + \epsilon_n \qquad \sum_{i=1}^{\omega} \epsilon_n < +\infty$$

entonces $(a_n)_{n\in\mathbb{N}}$ converge.

Demostración.

De las hipótesis se obtiene que

$$0 \leq a_{n+m} \leq a_n + \sum_{j=n}^{n+m} \epsilon_j.$$

Luego,

$$a_{n+m} - a_n \leq \sum_{j=n}^{n+m} \epsilon_j$$

$$\Rightarrow a_n - a_{n+m} \geq -\sum_{j=n}^{n+m} \epsilon_j.$$

Así, cuando $m, n \to \infty$,

$$|a_{n+m} - a_n| \leq \sum_{j=n}^{m+n} \epsilon_j \to 0;$$

es decir, $(a_n)_{n\in\mathbb{N}}$ es de Cauchy y por tanto converge.

□

Teorema 5.1 (Descenso de gradiente)

Sea $f \in C^1(\mathbb{R}^n)$ una función acotada inferiormente. Construimos una sucesión como sigue:

$\widehat{x}_0 \in \mathbb{R}^n$ y

$\widehat{x}_{i+1} = \widehat{x}_i - h_i d_i$,

donde *el tamaño* $h_i > 0$ y *la dirección del paso* $d_i \in \mathbb{R}^n$.

Se cumple que $\sum_i h_i = \infty$ y existen sucesiones $\{\lambda_i\}_{i\in\mathbb{N}\cup\{0\}}, \{\nu_i\}_{i\in\mathbb{N}\cup\{0\}}$ en $[0,\infty)$ tales que

$$\sum \lambda_i h_i < \infty,\ \sum \nu_i < \infty$$

y se satisface que

$$\langle \nabla f(\widehat{x}_i), d_i\rangle \geqslant \sigma(\|f(\widehat{x}_i)\|) - \lambda_i, \tag{1}$$

$$f(\widehat{x}_i) - f(\widehat{x}_{i+1}) \geqslant h_i\langle \nabla f(\widehat{x}_i), d_i\rangle - \nu_i, \tag{2}$$

en donde σ es una función de forzamiento.

En caso de que $\nabla f(\widehat{x}_i) = 0$ para alguna i, truncamos la sucesión (con la primera i que satisfaga esto). Entonces la sucesión $\{f(\widehat{x}_i)\}_{i\in\mathbb{N}}$ converge y para toda $b \in \mathbb{R}$

$$\inf_{i\geqslant b}\|\nabla f(\widehat{x}_i)\| = 0.$$

Además si $f \in LC^1_L(\mathbb{R}^n)$ y la sucesión $\{\|d_i\|\}_{i\in\mathbb{N}}$ es acotada, entonces

$$\lim_{i\longrightarrow\infty} \nabla f(\widehat{x}_i) = 0$$

y para cualquier punto de acumulación $\widehat{x}$ de $\{\widehat{x}_i\}_{i\in\mathbb{N}}$ se cumple que $\nabla f(\widehat{x}) = 0$.

Demostración. De existir un índice r tal que $\nabla f(\widehat{x}_r) = 0$, entonces $\widehat{x}_r$ ya es un punto crítico y truncamos la sucesión.
Suponga que, para toda $i \in \mathbb{N}$, $\nabla f(\widehat{x}_i) \neq 0$.

De las ecuaciones 1 y 2 obtenemos que

$$f(\widehat{x}_i) - f(\widehat{x}_{i+1}) \geqslant h_i\sigma(\|\nabla f(\widehat{x}_i)\|) - \lambda_i h_i - \nu_i. \tag{3}$$

Sea $a = \text{ínf}_{\widehat{y}\in\mathbb{R}^n}\{f(\widehat{y})\}$. Usando la ecuación 3, obtenemos que

$$0 \leq f(\widehat{x}_{i+1}) - a \leq f(\widehat{x}_i) - a + \lambda_i h_i + \nu_i.$$

Como $\sum \lambda_i h_i < \infty,\ \sum \nu_i < \infty$, y si $\epsilon_i = h_i\lambda_i + \nu_i$, entonces $\sum_i \epsilon_i < \infty$.
Usando el lema 5.2, con $a_n = f(\widehat{x}_n) - a$, se tiene que $\{a_n\}_n$ converge; por lo tanto, $\{f(\widehat{x}_i)\}_{i\in\mathbb{N}}$ converge.

Aplicando 3 obtenemos que, para $i > b$,

$$\begin{aligned} f(\widehat{x}_b) - f(\widehat{x}_i) &= \sum_{j=b}^{i-1}(f(\widehat{x}_i) - f(\widehat{x}_{i+1})) \\ &\geqslant \sum_{j=b}^{i-1} h_j\sigma(\|\nabla f(\widehat{x}_j)\|) - \sum_{j=0}^{i-1}(\lambda_j h_j + \nu_i). \end{aligned}$$

Luego,

$$f(\widehat{x}_b) - f(\widehat{x}_i) \geqslant \inf_{b\leq j\leq i-1} \sigma(\|\nabla f(\widehat{x}_j)\|) \sum_{j=0}^{i-1} h_j - \sum_{j_0}^{i-1} (\lambda_j h_j + \nu_j).$$

Por lo que, para toda $i > b$,

$$f(\widehat{x}_b) - a \geqslant \inf_{j\geqslant b} \sigma(\|\nabla f(\widehat{x}_j)\|) \sum_{j=0}^{i-1} h_j - \sum_{j=0}^{\infty} (\lambda_j h_j + \nu_j),$$

entonces

$$f(\widehat{x}_b) - a \geqslant \inf_{j\geqslant b} \sigma(\|\nabla f(\widehat{x}_j)\|) \sum_{j=0}^{\infty} h_j - \sum_{j=0}^{\infty} (\lambda_j h_j + \nu_j).$$

Como por hipótesis $\sum_{j=0}^{\infty} h_j = \infty$, entonces, para todo $b \geqslant 0$

$$\inf_{j\geqslant b} \sigma(\|\nabla f(\widehat{x}_j)\|) = 0.$$

Por definición de función de forzamiento tenemos que, para todo $b \geqslant 0$

$$\inf_{j\geqslant b} \|\nabla f(\widehat{x}_j)\| = 0 \tag{4}$$

Ahora supongamos que $f \in LC^1_L(\mathbb{R}^n)$ y, para toda $i \in \mathbb{N}$, $\|d_i\| \leq c$.

Demostraremos que $\|\nabla f(\widehat{x}_i)\| \longrightarrow 0$ cuando $i \longrightarrow \infty$.

Supongamos lo contrario. Entonces existe $\epsilon > 0$ y una subsucesión $\{\widehat{x}_{i_j}\}_{j\in\mathbb{N}}$ tal que, para toda j,

$$\|\nabla f(\widehat{x}_{i_j})\| \geqslant \epsilon.$$

Además de la ecuación 4 se sigue que para toda j existe otro índice L, con $L > i_j$, tal que

$$\|\nabla f(\widehat{x}_L)\| \leq \frac{\epsilon}{2}.$$

Denotaremos por $L(j)$ al número entero más pequeño que cumple que:

$$L(j) > i_j \qquad \|\nabla f(\widehat{x}_{L(j)})\| \leq \tfrac{\epsilon}{2} \qquad \|\nabla f(\widehat{x}_s)\| > \tfrac{\epsilon}{2}$$

para todo $s \in \{i_j, ..., L(j) - 1\}$

Usamos la desigualdad del triángulo y el hecho de que $f \in LC^1_L(\mathbb{R}^n)$ para obtener

$$\begin{aligned}
\frac{\epsilon}{2} &\leq \|\nabla f(\widehat{x}_{i_j})\| - \|\nabla f(\widehat{x}_{L(j)})\| \\
&\leq \|\nabla f(\widehat{x}_{i_j}) - \nabla f(\widehat{x}_{L(j)})\| \\
&\leq L\|\widehat{x}_{i_j} - \widehat{x}_{L(j)}\| = L\|\sum_{t=i_j}^{L(j)-1} (\widehat{x}_t - \widehat{x}_{t+1})\| \\
&\leq L \sum_{t=i_j}^{L(j)-1} \|\widehat{x}_t - \widehat{x}_{t+1}\| = L \sum_{t=i_j}^{L(j)-1} h_t\|d_t\| \\
&\leq LC \sum_{t=i_j}^{L(j)-1} h_t.
\end{aligned}$$

Entonces obtenemos

$$\frac{\epsilon}{2} \leq LC \sum_{t=i_j}^{L(j)-1} h_t,$$

por lo que

$$\sum_{t=i_j}^{L(j)-1} h_t \geqslant \frac{\epsilon}{2LC} := H. \tag{5}$$

Por las ecuaciones 3 y 5, obtenemos que

$$\begin{aligned} f(\widehat{x}_{i_j}) - f(\widehat{x}_{L(j)}) &= \sum_{t=i_j}^{L(j)-1} (f(\widehat{x}_t) - f(\widehat{x}_{t+1})) \\ &\geqslant \sum_{t=i_j}^{L(j)-1} h_t \sigma(\|\nabla f(\widehat{x}_t)\|) - \sum_{t=i_j}^{L(j)-1} (\lambda_t h_t + \nu_t) \\ &\geqslant \inf_{i_j \leq t \leq L(j)-1} \sigma(\|\nabla f(\widehat{x}_t)\|) H - \sum_{t=i_j}^{\infty} (\lambda_t h_t + \nu_t). \end{aligned}$$

Como la sucesión $\{f(\widehat{x}_i)\}_{i\in\mathbb{N}}$ converge y $\lim_{j\longrightarrow\infty} \sum_{t=i_j}^{\infty} (\lambda_t h_t + \nu_t) = 0$, obtenemos que

$$\lim_{j\longrightarrow\infty} \inf_{i_j \leq t \leq L(j)-1} \sigma(\|\nabla f(\widehat{x}_t)\|) = 0. \tag{6}$$

Como $\{f(\widehat{x}_i)\}_{i\in\mathbb{N}}$ es una sucesión de Cauchy, entonces $\lim_{j\longrightarrow\infty} |f(\widehat{x}_{i_j}) - f(\widehat{x}_{L(j)})| = 0$.

Note que por la elección de i_j y $L(j)$ tenemos que, para toda $t \in \{i_j, i_{j+1}, ..., L(j)-1\}$,

$$\|\nabla f(\widehat{x}_t)\| > \frac{\epsilon}{2} \tag{7}$$

pues $L(j)$ es el primer natural mayor que i_j tal que $\|\nabla f(\widehat{x}_{L(j)})\| \leq \frac{\epsilon}{2}$.

Las ecuaciones 6 y 7 son contradictorias porque σ es una función de forzamiento. Por lo tanto

$$\lim_{i\longrightarrow\infty} \|\nabla f(\widehat{x}_i)\| = 0.$$

Para terminar la demostración sólo falta probar que, para toda $\widehat{x} \in \{\widehat{x}_i\}_{i\in\mathbb{N}}$, si $\widehat{x}$ es un punto de acumulación, entonces

$$\nabla f(\widehat{x}) = 0.$$

Sea $\widehat{x}$ un punto de acumulación de $\{\widehat{x}_i\}_{i\in\mathbb{N}}$. Entonces existe una subsucesión $\{\widehat{x}_{s_n}\}_{n\in\mathbb{N}}$ tal que

$$\lim_{n\longrightarrow\infty} \widehat{x}_{s_n} = \widehat{x}.$$

Como $\{\|\nabla f(\widehat{x}_i)\|\} \longrightarrow 0$, entonces:

$$\lim_{n\longrightarrow\infty} \nabla f(\widehat{x}_{s_n}) = 0$$

Como ∇f es una función continua, entonces:

$$0 = \lim_{n\longrightarrow\infty} \nabla f(\widehat{x}_{s_n}) = \nabla f(\lim_{n\longrightarrow\infty} \widehat{x}_{s_n}) = \nabla f(\widehat{x}).$$

□

Corolario 5.1

Sean $f \in LC^1_L(\mathbb{R}^n)$ una función acotada inferiormente y $M \in \mathbb{R}$ tal que, para todo $\widehat{x} \in \mathbb{R}^n$,

$$\|\nabla f(\widehat{x})\| \leq M. \tag{1}$$

Construimos una sucesión de la siguiente forma:

$$\lambda_0 \in \mathbb{R}^n \text{ arbitrario y}$$

$$\lambda_{i+1} = \lambda_i - h_i d_i, \tag{2}$$

en donde

$$di = \nabla f(\widehat{x}_i) - \widehat{e}_i \tag{3}$$

y si $\nabla f(\widehat{x}_i) = 0$ paramos.

Si $\widehat{e}_i \in \mathbb{R}^n$ y $h_i \in (0, \infty)$ son tales que, para toda i,

$$\sum h_i = \infty, \quad \sum h_i^2 < \infty, \quad \sum h_i \|\widehat{e}_i\| < \infty \quad \text{y para algún } \gamma \in \mathbb{R} \quad \|\widehat{e}_i\| \leq \gamma,$$

entonces las conclusiones del Teorema 5.1 se cumplen, es decir, $(f(\widehat{x}_i))_{i\in\mathbb{N}}$ converge,

$$\lim_{n\to\infty} \nabla f(\widehat{x}_i) = 0$$

y para cualquier punto de acumulación $\widehat{x}$ de $(\widehat{x}_i)_{i\in\mathbb{N}}$, se tiene que $\nabla f(\widehat{x}) = 0$.

Demostración. Demostraremos las hipótesis del Teorema 5.1, es decir, veamos que:

$$\langle \nabla f(\widehat{x}_i), d_i \rangle \geq \sigma\left(\|\nabla f(\widehat{x}_i)\|\right) - \lambda_i \qquad \langle \nabla f(\widehat{x}_i), d_i \rangle \geq \sigma\left(\|\nabla f(\widehat{x}_i)\|\right) - \lambda_i$$

$$f(\widehat{x}_i) - f(\widehat{x}_{i+1}) \geq h_i \langle \nabla f(\widehat{x}_i), d_i \rangle - v_i$$

en donde tenemos que encontrar $\lambda_i \geq 0$, $v_i \geq 0$ tales que $\sum \lambda_i h_i < \infty$, $\sum v_i = \infty$ y ademas queremos ver que, para toda i, $\|d_i\| \geq c$.

Primero veamos que $\|d_i\| \leq c$, para alguna c.
De las ecuaciones 1, 2 y 3 tenemos que, para toda i,

$$\|d_i\| \leq \|\nabla f(\widehat{x}_i)\| + \|\widehat{e}_i\| \leq M + \gamma := c.$$

Ahora veamos como encontramos λ_i, v_i.
Utilizamos la desigualdad de Cauchy y tomamos $\sigma(s) = s^2$. Calculamos

$$\begin{aligned}
\langle \nabla f(\widehat{x}_i), d_i \rangle &= \langle \sigma\left(f\left(\widehat{x}_i\right)\right), \sigma\left(f\left(\widehat{x}_i\right)\right) - \widehat{e}_i \rangle \\
&= \|\nabla f(\widehat{x}_i)\|^2 - \langle \nabla f(\widehat{x}_i), \widehat{e}_i \rangle \\
&\geq \|\nabla f(\widehat{x}_i)\|^2 - \|\nabla f(\widehat{x}_i)\| \|\widehat{e}_i\| \\
&\geq \sigma\left(\sigma\left(f\left(\widehat{x}_i\right)\right)\right) - M\|\widehat{e}_i\| \\
&= \sigma\left(\sigma\left(f\left(\widehat{x}_i\right)\right)\right) - \lambda_i.
\end{aligned}$$

Con $\lambda_i = M\|\widehat{e}_i\|$, cumple

$$\sum h_i \lambda_i < \infty, \quad \lambda_i \geq 0,$$

entonces

$$\langle \nabla f(\widehat{x}_i), d_i \rangle \geq \sigma\left(\|f(\widehat{x}_i)\|\right) - \lambda_i.$$

Ahora calculamos, usando el Lema 5.1,

$$|f(\widehat{x}_i) - f(\widehat{x}_{i+1}) - \langle \nabla f(\widehat{x}_i), \widehat{x}_i - \widehat{x}_{i+1} \rangle| \leq \frac{L}{2} \|\widehat{x}_{i+1} - x_i\|^2.$$

Por lo que, usando la ecuación 1, obtenemos

$$\begin{aligned} f(\widehat{x}_i) - f(\widehat{x}_{i+1}) &\geq \langle \nabla f(\widehat{x}_i), \widehat{x}_i - \widehat{x}_{i+1} \rangle - \frac{L}{2} \|\widehat{x}_{i+1} - \widehat{x}_i\|^2 \\ &= \langle \nabla f(\widehat{x}_i), h_i d_i \rangle - \frac{L}{2} \|h_i d_i\|^2 \\ &\geq h_i \langle \nabla f(\widehat{x}_i), d_i \rangle - \frac{L}{2} h_i^2 c^2. \end{aligned}$$

Ahora tomamos $v_i = \frac{L}{2} h_i^2 c^2$ con lo que tenemos

$$f(x_i) - f(x_{i+1}) \geq h_i \langle \nabla f(x_i), d_i \rangle - v_i,$$

con $v_i \geq 0$ y $\sum v_i < \infty$. Con esto terminamos de demostrar las hipótesis del Teorema 5.1 y, por lo tanto, se cumplen sus conclusiones.

□

5.5. Aplicación del Descenso del Gradiente

Ya sabemos que el aprendizaje de una Red Neuronal está enfocado en minimizar una *función de costo*. Sin embargo, ¿Por qué necesitamos una función de costo?. Necesitamos esa función ya que se requiere fabricar un algoritmo que permita encontrar los *weights* y *bias* para que la red aprenda de la mejor manera posible. Tal que teniendo lo anterior, el output $y(\widehat{x})$ de la red se aproximará para cada input $\widehat{x}$ de nuestro conjunto de datos para entrenamiento.

Conociendo los *inputs* y *outputs* para la Red, podemos elegir la función de costo que mejor se ajuste a nuestro problema. En este ejemplo utilizaremos la *Quadratic Cost Function*, ya definida anteriormente:

$$cost(T_1, T_2, \cdots, T_n) = C(w_i, b_i) = \frac{1}{2n} \sum_{i=1}^{N} \|y(\widehat{x}_i) - f(\widehat{x}_i)\|^2$$

En este caso los $(T_1, T_2, \cdots, T_n)$ corresponden a los *weights* y *bias* de la red llamados (w_i, b_i). Lo que se espera es que C se acerque lo más posible a 0, encontrando un conjunto de (w, b) que vuelvan al costo lo más pequeño posible. Estudiando la función, lo que estamos haciendo es comparar los valores que ya sabemos que son correctos $y(\hat{x})_i$ con los valores estimados por la red $f(\hat{x})_i$. De esa manera se mide que tan efectivas son las predicciones. Para lograr lo anterior, aplicamos *El Descenso del Gradiente*.

Queremos minimizar la función de costo $C(\hat{v})$ con $\widehat{v} = (w_i, b_i)$. En el caso general tenemos $\hat{v} = v_1, v_2, ..., v_n$. Por practicidad nos quedaremos con $\hat{v} = v_1, v_2$. Debemos encontrar el mínimo global de una función de dos variables.

$$\Delta C \approx \frac{\partial C}{\partial v_1} \Delta v_1 + \frac{\partial C}{\partial v_2} \Delta v_2$$

Queremos que $\Delta C < 0$, entonces encesitamos un par de Δv_1 y Δv_2 que nos funcionen. Definimos:

$$\Delta v = \begin{cases} \Delta v_1 \\ \Delta v_2 \end{cases}$$

Entonces:

$$\nabla C = \begin{cases} \frac{\partial C}{\partial v_1} \\ \frac{\partial C}{\partial v_2} \end{cases}$$

$$\Delta C \approx \nabla C \cdot \Delta v$$

Ya definido el gradiente, es necesario asegurarnos de que siempre vaya en descenso, pues lo que queremso es minimizar la función C. Por lo tanto, hay que escoger una Δv tal que logremos dicho propósito:

$$\Delta v \equiv -h \, \nabla C$$

A h se le conoce como *learning rate* y es un hiperparámetro que puede modificarse manualmente. Este hiperparámetro describe que tan corto va a ser el trayecto del algoritmo, si η es muy pequeña, podemos quedarnos atrapados en un mínimo local y jamás llegar al global. Por otro lado, si es demasiado grande, podemos pasarnos de nuestro mínimo global y nunca encontralo.

Ya conocido el *learning rate* podemos regresar a la ecuación anterior y asegurar que $\Delta C \leq 0$, pues:

$$\Delta C \approx -h \, \nabla C \cdot \nabla C = -h||\nabla C||^2$$

Pese a todo lo anterior, aún no tenemos descrito ningún algoritmo para asegurarnos que, paso a paso, podemos encontrar el mínimo global de una función. Entonces, el método quedaría así:

$$v \to v' = v - h \, \nabla C$$

Cada que iteremos en el vector v, a la siguiente iteración ser realizará el cambio propuesto por $v' = v - \eta \, \nabla C$. La idea es usarlo para encontrar los weights y bias que minimicen al costo. Hay que restar nuestro Descenso del Gradiente con los weights y bias:

$$\nabla C \equiv \left(\frac{\partial C}{\partial w_k}, \frac{\partial C}{\partial b_l} \right)$$

$$w_k \to w_k' = w_k - h \, \frac{\partial C}{\partial w_k}$$

$$b_l \to b_l' = b_l - h \, \frac{\partial C}{\partial b_l}$$

De esta manera usamos el descenso del gradiente de forma algoritmica para llegar al mínimo global de la función de costo C. Pese a todo, en la vida real es muy difícil llegar al mínimo global de las funciones utilizadas en Machine Learning, ya que esto consume bastante poder de computo, además de algunos problemas usuales del proceso como el *desvanecimiento del gradiente*. Lo anterior ocurre cuando el proceso iterativo es tan largo y tardado que el gradiente comienza a desaparecer provocando que el aprendizaje se pause.

6 | Glosario

- **Clasificación Binaria**
  Claisficación de datos en dos clases, normalmente descritos por las etiquetas '0' y '1'.
  - **Verdadero Positivo**
    Valor predecicho para el elemento de la muestra:1
    Valor del elemento de la muestra en la población: 1
  - **Verdadero Negativo**
    Valor predecicho para el elemento de la muestra:0
    Valor del elemento de la muestra en la población: 0
  - **Falso Positivo**
    Valor predecicho para el elemento de la muestra : 1
    Valor del elemento de la muestra en la población: 0
  - **Falso Negativo**
    Valor predecicho para el elemento de la muestra: 0
    Valor del elemento de la muestra en la población: 1
- **Tasa de verdaderos positivos (Sensibilidad)**
  Porción de los casos positivos (etiqueta 1) que son predichos correctamente.

$$\frac{Verdaderos\ Positivos}{Falsos\ Negativos + Verdaderos\ Positivos}$$

- **Tasa de verdaderos negativos (Especificidad)**
  Porción de los casos negativos (etiqueta 0) que son predichos correctamente.

$$\frac{Verdaderos\ Negativos}{Falsos\ Positivos + Verdaderos\ Negativos}$$

- **Tasa de falsos positivos**
  Porción de los casos negativos que son predichos como positivos

$$\frac{Falsos\ Positivos}{Falsos\ Positivos + Verdaderos\ Negativos}$$

  Notar que $Tasa\ de\ falsos\ positivos = 1 - Especificidad$

- **Función de pérdida/costo**

  Función que calcula la diferencia entre los valores reales y los predichos por el modelo.

- **Métrica**
  Función que evalua el desempeño del modelo

- **Exactitud (Accuracy)**
  Clasificación binaria.
  Cociente del numero de predicciones correctas entre el total de prediciones hechas.

$$\frac{Verdaderos\ Positivos + Verdaderos\ Negativos}{Tamaño\ de\ la\ muestra}$$

- **Precisión (Precision)**

$$\frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos}$$

- **Memoria (Recall)**

$$\frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$

- **Área bajo la curva (AUC)**
  Tomemos un caso de clasificación binaria. En entonces el modelo tendrá como valores crudos valores en el intervalo [0,1], de modo que para clasificarlos se definirá un umbral, de modo que los valores mayores o iguales a este umbral se etiquetan como positivos (1), y los que están por debajo como negativos (0).
  - **Curva ROC**
    Mide la capacidad del modelo para distinguir entre 2 clases.
    Cada umbral posible nos define una sensibilidad y una especificidad concretas. Graficando la relación de 1-especificidad contra la sensibilidad obtenemos la curva ROC. La gráfica se halla en la celda $[0,1] \times [0,1]$

  El área bajo la curva ROC es un valor en el intervalo [0,1], y entre más próximo sea este valor a 1 mejor será el modelo.

- **Matriz de confusión**

  En la clasificación binaria, toma la la siguiente forma:

	1	0
1	$Verdaderos\ Positivos$	$Falsos\ Negativos$
0	$Falsos\ Positivos$	$Verdaderos\ Negativos$

  Conforme los datos se concentren más en la diagonal mejor será el modelo. A partir de esta matriz se puede calcular la exactitud sumando los valores de la diagonal y dividiendo entre el tamaño de la muestra.

  La matriz se puede generalizar del siguiente modo:

	L_1	L_2	...	L_n
L_1	a_{11}	a_{12}	...	a_{1n}
L_2	a_{21}	a_{22}	...	a_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
L_n	a_{n1}	a_{n2}	...	a_{nn}

  Donde la entrada a_{ij} es el numero de datos cuyo valor real es L_i y L_j es su valor predicho.

- **F1 Score**
  Media armónica entre precision y recall.
  Entre mayor sea la media major será el modelo.

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

- **Raiz del error cuadrático medio(RMSE)**
  Rregresión
  Medida de rendemiento que evalua la distancia, con la métrica euclideana, entre el vector de valores predichos y el vector de valores reales, entre la raiz cuadrada del tamaño de la muestra. Al dividir estamos normalizando nuestro resultado.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_j - \hat{y}_j)^2}$$

- **Error absoluto medio (MAE)**
  Rregresión
  Medida de rendemiento que evalua la distancia, con la métrica 1, entre el vector de valores predichos y el vector de valores reales, entre el tamaño de la muestra. Al dividir estamos normalizando nuestro resultado.

$$\frac{1}{n}\sum_{i=1}^{n}|y_j - \hat{y}_j|$$

- **Error cuadrático medio**
  Rregresión
  Cuadrado de **RMSE**.

$$\frac{1}{n}\sum_{i=1}^{n}(y_j - \hat{y}_j)^2$$

- **Sobreajuste (Overfitting)**
  Conflicto derivado del ajuste exacto del modelo con los datos de entrenamiento, que lleva a una rigidez capaz ignorar tendencias en nuestro fenómeno de estudio, y entorpecer prediccion para datos ajenos a nuestro conjunto de entrenamiento.

- **Underfitting**
  Conflicto que deriva de contar con un conjunto de entrenamiento insuficiente en tamaño como para reconocer tendendecias en nuestro fenómeno de estudio.

- **Épocas**
  Número de veces que un algoritmo procesa al total del conjunto de entrenamiento.

- **Iteraciones**
  Número de veces que el algoritmo procesa un lote (batch) del conjunto de entrenamiento.

- **Batches**
  Subconjuntos del conjunto de datos de entrenamiento. Los batches sustituyen al conjunto total de datos de entrenamiento en la ejecución del algoritmo, a fin de evitar iteraciones con cantidades de datos muy grandes.

- **Batch size**
  Cantidad de datos por batch.

- **Mini-batch**
  Subconjunto aleatorio del conjunto de entrenamiento, en ocasiones nombrada como 'estocástica'.