UNIVERSIDAD DE GRANADA

HISTORIA DE LAS REDES NEURONALES PROFUNDAS

Blanca Cano Camarero, Manuel Gachs Ballegeer y Jose Luis Ruiz Benito

13 de diciembre de 2021

Introducción

En nuestros días, el aprendizaje profundo o «Deep Learning» es un campo en continua expansión. Desde procesamiento de imágenes hasta predicciones de acciones en bolsa, las redes neuronales se van implantando en todos los campos del conocimiento. Sin embargo, siempre son tratadas desde un punto de vista práctico, como herramientas que resuelven un problema. Aunque es cierto que sirven de herramientas, la motivación inicial para su creación es muy distinta.

El concepto de neurona artificial nace en el primer tercio del siglo XX, como intento de modelar el pensamiento humano en forma de procesos lógicos. Esta primera etapa de investigación culmina con la invención del perceptrón en 1958 por Frank Rosenblatt. Es importante notar que todas estas investigaciones tenían un objetivo más allá de la resolución de problemas complejos utilizando máquinas u ordenadores: comprender el proceso de aprendizaje y cognición del ser humano.

Empero, estos primeros modelos tenían sus carencias y reservas por parte de la comunidad científica. En 1969, la publicación de la demostración de que los modelos eran incapaces de resolver problemas lógicos simples hizo que el campo casi muriera, hasta el descubrimiento de un modelo en 1986, más complejo, que superaba estas limitaciones.

Este modelo, no obstante, usaba un método que sus autores eran incapaces de relacionar con el mundo de la cognición. Este podría decirse que fue el inicio de la separación de las redes neuronales con el campo de la psicología y la neurociencia y el inicio del enfoque actual de resolución de problemas. Estos nuevos descubrimientos, sumados a una mejora en las capacidades computacionales de los ordenadores, provocaron un auge en el interés acerca de las redes, que perdura hasta hoy.

El objetivo de este trabajo es dar a conocer estos aspectos tan interesantes y poco tratados, al menos en el ámbito popular, acerca de las redes neuronales.

Índice

Figura 1: Arthur L. Samuel (1901-1990) [30] se graduó en Ingeniería Electrónica en el MIT. Trabajó en los laboratorios Bell, la Universidad de Illinois, en IBM y en la Universidad de Stanford (1966). Fue un pionero de los videojuegos y la inteligencia artificial. Popularizó el término «Machine Learning» en 1959. Implementó una IA para jugar a las damas, el primer caso de éxito en aprendizaje automático. Contribuyó notablemente en el desarrollo de TeX.

Figura 2: Tom M. Mitchell (1951) estudió Ingeniería Electrónica en el MIT, se doctoró en la Universidad de Stanford y ejerce en la Univesidad de Carnegie Mellon. Es conocido por su libro de texto Machine Learning, una obra fundamental del campo del mismo nombre. Ha recibido numerosas condecoraciones y participa en asociaciones que promueven la ciencia y la ingeniería.

1 El Aprendizaje Automático

Las redes neuronales son un tipo de algoritmos de Aprendizaje Automático. Por ello, no sería prudente exponer la historia de las redes neuronales sin dar en primer lugar una nociones básicas del área del saber a la que pertenecen.
El término de Aprendizaje Automático [12] fue acuñado en 1959 por Arthur Samuel para hacer referencia a los sistemas informáticos que pueden «aprender» por sí mismos, es decir, mejorar su eficacia y rendimiento de forma autónoma a partir de los datos, sin que en esas mejoras intervenga un programador.
Fue en 1997 cuando Tom Mitchell propuso una definición más formal de aprendizaje [21], aproximada a la dada en el libro *Learning from data* [6], que exponemos en seguida.
El aprendizaje es un proceso por el cual se estima una dependencia desconocida (input-output) o la estructura de un sistema a partir de un número finito de observaciones. El aprendizaje se compone de tres elementos principales:

- Un generador o función de distribución de la cual se extraen vectores aleatorios $x \in \mathbb{R}^n$ que dependen de una función de densidad desconocida [1].
- Un sistema que produce un vector de salida y por cada entrada del vector x a partir del valor fijo $p(y|x)$, que es desconocido también.
- Una *learning machine*, que en el caso más general no es más que un conjunto de funciones abstractas cuyos elementos son de la forma $f(x, w), w \in \omega$ (relaciones entre la entradas y el espacio de salidas posibles).

Así pues, el objetivo del aprendizaje automático es encontrar una función que se aproxime a la función de densidad desconocida.
Por lo general la teoría se fundamenta en minimizar el error de estimadores, como el error cuadrático medio, ya que este se trata de un UMVUE (estimador insesgado de mínima varianza).

$$ECM = \frac{1}{n}\sum_{i=0}^{n}(f(x_i) - y_i)^2, \tag{1.1}$$

donde $f(x_i)$ representa la predicción y y_i la etiqueta de entrenamiento de x_i, es decir, su valor real.
Objetivos similares se realizan también desde otros campos de la matemática como es el de teoría de la aproximación.
No es habitual comenzar hablando del aprendizaje automático con teoría de la aproximación, pero en nuestro caso, que pretendemos centrarnos en redes neuronales. Esta teoría es fundamental para el Teorema de aproximación universal de las redes neuronales [34].

[1] De hecho, encontrar esta función resolvería el problema de aprendizaje.

2 Teoría de la aproximación. Teorema de aproximación de Stone Weierstrass.

Figura 3: Karl Weierstrass (1815-1897) [38] fue un matemático alemán considerado el padre del análisis moderno. Entre sus logros más destacados figuran la definición de la continuidad de una función, demostrando el teorema del valor medio; y el teorema de Bolzano-Weierstrass usado posteriormente para estudiar las propiedades de funciones continuas en intervalos cerrados.

El Teorema de Aproximación de Stone-Weierstrass es fundamental para el Aprendizaje Automático. Fue creado originalmente por Karl Weierstrass en 1885 y mejorado por Marshall Harvey Stone en 1937.

Teorema 2.1 (*Weierstrass, 1885*): Para cualquier $\varepsilon > 0$ y para cualquier función f continua sobre un intervalo $[a,b] \subset \mathbb{R}$ existe un polinomio con coeficientes reales, p, tal que

$$\sup_{x\in[a,b]} |f(x) - p(x)| < \varepsilon \tag{2.1}$$

Teorema 2.2 (*Stone-Weierstrass, 1937*): Sea K un subconjunto compacto de $\mathbb{R}^p$ y sea $\mathcal{A}$ una colección de funciones continuas de K a $\mathbb{R}$ con las siguientes propiedades:

1. La función constantemente uno, definida como $e(x) = 1$, para cualquier $x \in K$ pertenece a $\mathcal{A}$.
2. Cerrado para sumas y producto para escalares reales. Si f, g pertenece a $\mathcal{A}$, entonces $\alpha f + \beta g$ pertenece a $\mathcal{A}$.
3. Cerrado para producto. Para $f, g \in \mathcal{A}$, se tiene que fg pertenece a $\mathcal{A}$.
4. Separación de K, es decir si $x \neq y$ pertenecientes a K, entonces existe una función f en $\mathcal{A}$ de tal manera que $f(x) \neq f(y)$.

Se tiene que toda función continua de K a $\mathbb{R}$ puede ser aproximada en K por funciones de $\mathcal{A}$.

Figura 4: Marshall Harvey Stone (1903-1989) fue un matemático estadounidense que contribuyó al análisis, a la topología y al estudio de las álgebras booleanas. [19].Generalizó el enunciado del teorema de Weierstrass y simplificó su demostración en 1937.

Figura 5: John von Neumann (1903-1957) fue un matemático húngaro-estadounidense conocido por el desarrollo de la arquitectura de computadores que lleva su nombre y que sigue utilizándose para el desarrollo de procesadores.

Figura 6: Frank Rosenblatt (1928-1971) fue un psicólogo estadounidense notable en el campo de la inteligencia artificial. Se graduó y doctoró en letras en la Universidad de Cornell. Trabajó en el Laboratorio Cornell Aeronáutico, donde fue sucesivamente psicólogo investigador, psicólogo senior y jefe de la sección de sistemas cognitivos. Allí fue donde llevó a cabo sus primeros trabajos sobre perceptrones, que culminaron en el desarrollo y construcción del harware del perceptrón Mark I, el primer ordenador que podría aprender habilidades nuevas a base de prueba y error.

3 El perceptrón

En la historia de la Inteligencia Artifical, una de las preguntas que siempre ha existido es la posibilidad de suplantar el pensamiento en una máquina utilizando leyes lógicas. Esta pregunta tiene también otra interpretación: ¿Es posible modelar el pensamiento como un proceso mental gobernado por leyes lógicas? Esta pregunta es la que da lugar al nacimiento de las redes neuronales y el aprendizaje profundo a partir del artículo publicado en 1943 por Warren McCulloh y Walter Pitts. Titulado *A Logical Calculus of the Ideas Immanent in Nervous Activity*, formaliza la actividad nerviosa de las redes de neuronas en el cerebro humano mediante la lógica proposicional. Introduce el concepto de red neuronal artificial y algunas características como la actividad binaria de una neurona y la invariabilidad de la estructura de la red. Además, divide las neuronas en dos grupos: las que en la actualidad se conocen como las neuronas de entrada, y las neuronas de salida [36].
Este artículo serviría como inspiración no solamente para futuros trabajos en el campo de las redes neuronales, sino en la computación en general. John Von Neumann cita este artículo como una de sus inspiraciones más importantes en su trabajo [33].

3.1 El perceptrón de Rosenblatt

En 1958, Frank Rosenblatt publica un artículo de vital importancia para el desarrollo del aprendizaje automático, titulado *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*, [27].
En este trabajo el autor intenta dar respuesta a dos preguntas básicas:

1. ¿Cómo se almacenada la información?
2. ¿Cómo influye la información almacenada en la comprensión y en el comportamiento?

En la época, había dos corrientes diferenciadas que intentaban responder a estas cuestiones:

- La corriente de la “la memoria codificada” sostenía que la información sensible se almacenaba de forma codificada. Debía existir alguna regla que a cada estímulo sensible le asociase un código que almacenar. El reconocimiento posterior de un nuevo estímulo suponía su codificación y comparación sistemática con otros códigos ya almacenados en busca de una respuesta.
- La corriente “empiricista” o “conexionista” defendía que no existía tal almacenamiento de códigos, sino que la información se almacenaba en forma de conexiones entre neuronas o canales de transmisión en el sistema nervioso. Un estímulo seguirá las conexiones apropiadas para él, activando la respuesta adecuada sin necesidad de otro proceso que lo reconozca o identifique.

En su artículo, Rosenblatt se alinea con el "conexionismo". Llama "perceptrón" a un hipotético sistema nervioso, que generaliza sistemas biológicos y máquinas.
La *Teoría del Perceptrón* se basa en las siguientes suposiciones desarrolladas por otros científicos de la época, entre los que destacan Hebb, Hayek, Uttley y Ashby.

1. Las conexiones físicas del sistema nervioso relacionadas con el aprendizaje y el reconocimiento no son idénticas entre sujetos. La construcción inicial de una red neuronal es mayormente aleatoria, sujeta a un mínimo de condiciones genéticas.
2. El sistema original de células conectadas tiene una cierta plasticidad. Tras un periodo de actividad neuronal, la probabilidad de que un estímulo en un conjunto de células provoque la respuesta de otro conjunto distinto es probable que cambie, debido a cambios en las neuronas.
3. Por la exposición a grandes muestras de estímulos, estímulos "similares" tenderán a seguir caminos por las mismas células, mientras que los "distintos" seguirán otros caminos.
4. La aplicación de refuerzos positivos y negativos puede facilitar o dificultar cualquier formación de conexiones en proceso.
5. La similitud en estos sistemas se representa en algún nivel del sistema nerviosos como la tendencia de que estímulos similares activen respuestas similares. Esta similitud no tiene por qué ser un atributo formal de cierta clase de estímulos, sino que depende de la organización específica de la red, que evoluciona conforme interacciona con el entorno. La estructura de la red afecta y determina las clases de "cosas" que dividen el entorno que la red percibe.

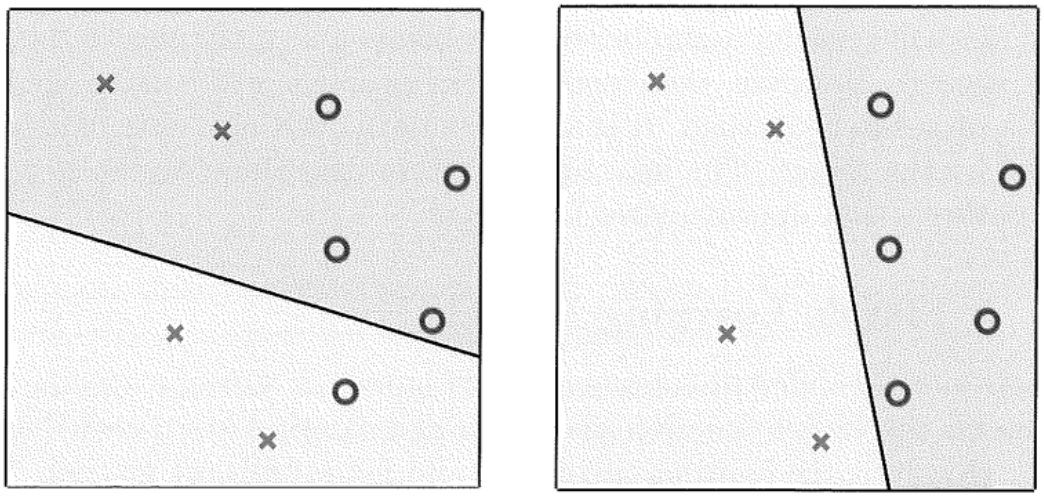
A partir de estas ideas, el autor expresa en términos matemáticos su teoría del funcionamiento de un perceptrón, que si bien no coincide exactamente con la definición formal que se usa en la actualidad, es una notable precursora.

3.2 Noción actual de perceptrón

Los modelos actuales de redes neuronales, que explicamos más adelante, se componen de unidades mínimas, llamadas neuronas. La neurona más simple posible es el perceptrón, entendido como el modelo que presentamos a continuación.
Un perceptrón [44] es una función $h : \mathbb{R}^d \longrightarrow \{-1, 1\}$ definida por:

$$h(x) = sign(\sum_{i=1}^{d} w_i x_i + b), \tag{3.1}$$

donde $x = (x_1, \ldots, x_d)$ son las entradas y $w_1, \ldots, w_d$ son los pesos que se le asignan a las entradas.
Intuitivamente, el perceptrón clasifica los posibles vectores de entrada x en dos clases, -1 y 1. Si consideramos los posibles vectores de entrada como puntos del plano, el perceptrón crea una recta que separa el plano en dos, dejando en una semiplano aquellos puntos que clasifica como -1 y en el otro semiplano los que clasifica como 1. Pero esta separación no puede hacerse de cualquier manera, ya que en principio cada punto tendrá una clase verdadera y hay que conseguir que la clase que le asigne el perceptrón sea esa misma. Para ello necesitaremos ajustar los pesos $w_1, \ldots, w_d$.



(a) Misclassified data (b) Perfectly classified data

Figura 7: Ajuste de los pesos del perceptrón

En la figura 7 vemos dos posibles elecciones de pesos, la de la izquierda no consigue clasificar correctamente los puntos, mientras que la derecha sí.
Por tanto, cabe preguntarse cuándo es posible encontrar los pesos adecuados y cómo hacerlo. La respuesta es sencilla: se puede hacer cuando las clases son linealmente separables y mediante el *Algoritmo de Aprendizaje del Perceptrón*. Sin embargo a nivel práctico puede usarse el algoritmo, estableciendo algún criterio de parada, por ejemplo el número de iteraciones guardando el mejor peso o cuando se encuentre uno cuyo error esté bajo un umbral.
El funcionamiento del susodicho algoritmo es el siguiente:

1. Partimos de unos valores iniciales ($t = 0$) de los pesos, típicamente aleatorios cercanos a cero pero no nulos: $w(t) = (w_1(t), \ldots, w_d(t))$.
2. Si existe algún dato $x = (x_1, \ldots, x_d)$ que no sea correctamente clasificado utilizando $w(t)$, actualizamos w: $w(t+1) = w(t) + h(x)y$.
3. Repetimos el paso anterior mientras haya algún dato mal clasificado, cuando no lo haya finalizamos.

Al finalizar la ejecución del algoritmo obtendremos los pesos buscados.

4 Perceptrón multicapa

4.1 Limitaciones del perceptrón. El problema XOR

El perceptrón de Rosenblatt tenía una limitación, que fue ilustrada en 1969 en el libro de Marvin Minsky y Seymour Papert, *Perceptrons: An Introduction to Computational Geometry*. En él, describían un fallo que habían cometido McCulloh y Pitts en su trabajo de 1943: no habían tenido en cuenta la equivalencia como función lógica, y expusieron lo que se conoce como el problema XOR.

La función lógica XOR es la negación de la equivalencia, y se trata de un problema no lineal. Minsky y Papert demostraron en su libro que el perceptrón era incapaz de clasificar problemas no lineales. Las consecuencias de este tratado fueron nefastas para el campo de las redes neuronales, que se quedaron casi abandonadas hasta la invención del perceptrón multicapa por David E. Rumelhart, Geoffrey E. Hinton y Ronald J. Williams en 1986.

4.2 El perceptrón multicapa de Rumelhart, Hinton y Williams

Rumelhart, Hinton y Williams eran miembros y fundadores de un grupo de investigación en la Universidad de San Diego en California, llamado *Parallel Distributed Processing (PDP)*. El objetivo inicial de este grupo era realizar un compendio de todas las investigaciones relacionadas con las redes neuronales que se habían realizado hasta la fecha. Sin embargo, el grupo evolucionó, comenzando a realizar investigaciones que culminan con la invención del perceptrón multicapa en 1986.

El nombre de “perceptrón multicapa” es desafortunado, pues esta invención no se trata de una concatenación de perceptrones, sino de un nuevo sistema con un método de aprendizaje distinto. Este algoritmo fue denominado por los autores “regla delta generalizada” [8], pieza clave en el posterior desarrollo del algoritmo de retropropagación o “backpropagation”.

El perceptrón multicapa causó un auge de interés en las redes neuronales, puesto que se trataba de una arquitectura capaz de solucionar problemas no lineales, algo que todas las estructuras propuestas anteriormente eran incapaces de resolver. La no linealidad en la estructura brinda la posibilidad también de obtener representaciones internas complejas de entidades, una característica muy importante a la hora de representar fenómenos complejos que son comunes en el pensamiento humano, como el lenguaje. Sin embargo, ninguno de los autores presentó evidencias de que el cerebro aprende mediante la regla delta generalizada. Aunque pudiera parecer un problema menor, se trata de un asunto de importancia crítica a la hora de realizar modelos de cognición que intenten ser plausibles biológicamente, que era el objetivo del PDP [5]. El modelo de aprendizaje cerebral humano sigue siendo en la actualidad un tema de debate, puesto que todavía no se ha llegado a un consenso.

4.3 Definición del perceptrón multicapa

El perceptrón multicapa clásico se define como el conjunto de los siguientes elementos:

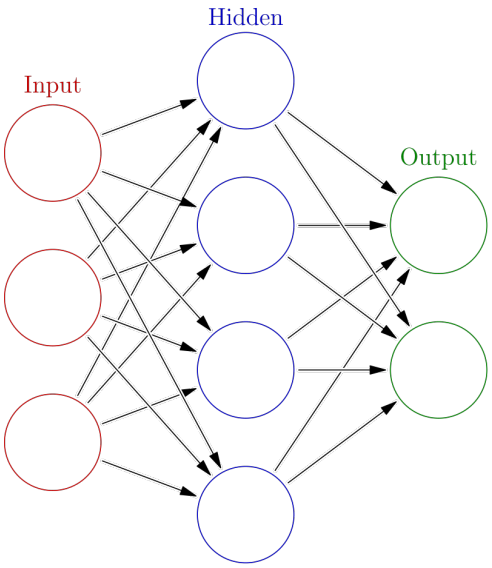


Figura 8: Perceptrón multicapa con una capa oculta

- Una función lineal que agrega las entradas y produce una salida.

$$z = f(x_n, w_n) = b + \sum_{i}^{n} x_i w_i \tag{4.1}$$

  donde

  - z es la salida.
  - b es una constante.
  - x_i son las entradas a la neurona.
  - w_i son los pesos de cada una de las entradas.

- Una función de activación sigmoidal.

$$a = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.2}$$

  El objetivo de esta función es eliminar la linealidad de la salida.

- Una función umbral en los problemas de clasificación.

$$\hat{y} = f(a) = \begin{cases} 1 & \text{si} \quad a > 0{,}5 \\ -1 & \text{en cualquier otro caso} \end{cases} \tag{4.3}$$

  O una función identidad en los casos de regresión

$$\hat{y} = f(a) = a \tag{4.4}$$

- Una función de costo que computa el error total producido por la red. Esta función se trata de la suma de los errores cuadráticos:

$$E = \frac{1}{2}\sum_{k}(\hat{y}_k - y_k)^2 \tag{4.5}$$

  donde

  - $\hat{y}_k$ es la predicción del k-ésimo ejemplo.
  - y_k es el resultado real del k-ésimo ejemplo.

- Un método de aprendizaje para realizar cambios en los pesos: regla delta generalizada.

4.4 La regla delta generalizada

Aunque se atribuya la regla delta generalizada a Rumelhart, Hinton y Williams, fue descubierta por primera vez por Paul Werbos en su tesis doctoral. No obstante, este descubrimiento pasó desapercibido. Fue descubierta por segunda vez por David Parker en 1981, que intentó obtener una patente para el descubrimiento. Acabó publicando su descubrimiento en 1985. La tercera y última vez que fue descubierta fue de forma independiente por Yann LeCun en 1985 y Rumelhart, Hinton y Williams en 1986 [33].

El proceso de esta regla es el siguiente:

1. Se realiza una prueba, es decir, se envía una entrada al perceptrón y se recoge la salida final.
2. Se compara el error realizado por la capa de salida usando la función costo definida anteriormente en (4.5).
3. Ese error es utilizado para modificar los pesos de sus conexiones con la capa anterior. La modificación es la siguiente:

$$\Delta w_i = \eta(\hat{y} - y)\sigma'(z)x_i \tag{4.6}$$

   Estos tres primeros pasos son idénticos a realizar un descenso de gradiente en un perceptrón.
4. Para cada capa oculta, en orden descendente, se computa el error, que no es más que la suma de todos los errores de las neuronas de la siguiente capa que están conectadas con ella. Esto es, si una red tiene tres capas ocultas, el error de la tercera neurona de la segunda capa es la suma de los errores de las neuronas de la tercera capa que tienen como entrada la salida de la neurona en cuestión. Cada uno de los pesos de esa neurona se modifica de la misma manera que los de la capa de salida [10].

Una de las propiedades más interesantes e importantes de esta regla es que es capaz de detectar correlaciones entre vectores linealmente independientes, mientras que las reglas anteriores eran incapaces de detectar este tipo de patrones [9].

5 Redes neuronales hacia delante. Retropropagación.

Las redes neuronales hacia delante o “feed-foward neural networks” (FFNNs) son las redes neuronales más usadas y las primeras en ser utilizadas. En ellas, las entradas siguen una única dirección (hacia delante), desde la capa de entrada hasta la capa de salida, pasando, si hay, por las capas ocultas. Es en este tipo de redes en las cuales se incluye el perceptrón y el perceptrón multicapa descritos anteriormente.

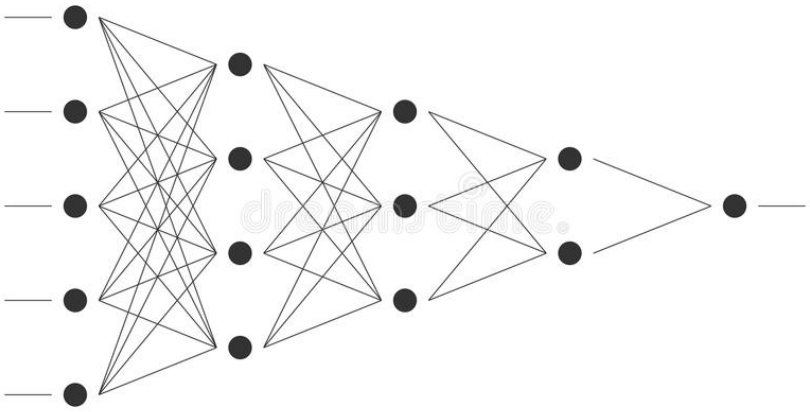


Figura 9: Ejemplo de estructura de una FFNN con cinco entradas y una salida

Cabe resaltar que el concepto de una red neuronal no cíclica es muy anterior al de perceptrón multicapa, pudiéndose remontar al primer artículo sobre neuronas artificiales de 1943 del que se ha hablado ya anteriormente, en el que en la segunda sección se describe la teoría lógica de redes neuronales “sin círculos” [36]. En el campo del Aprendizaje Profundo, la primera publicación de una red neuronal hacia delante es la publicada en 1968 por Alexey Ivakhnenko y el GMDH (*Group Method of Data Handling*), usando una estructura de perceptrones de varias capas [31].

En la actualidad, las redes neuronales hacia delante que se utilizan están basadas en el perceptrón multicapa, aunque con ciertas generalizaciones. La función de activación puede ser otra función aparte de la sigmoide. Se suelen usar las siguientes funciones de activación:

Sigmoide

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{5.1}$$

La función sigmoide o logística es la función de activación clásica. Se trata de una función diferenciable, lo que la hace cómoda a la hora de utilizar retropropagación. Se suele utilizar en modelos donde la salida es una probabilidad, puesto que la imagen está acotada entre cero y uno. Sin embargo, esta función sufre del problema de desvanecimiento de gradiente . Además, la salida de todas las neuronas tiene el mismo signo (positivo), lo que puede dificultar el

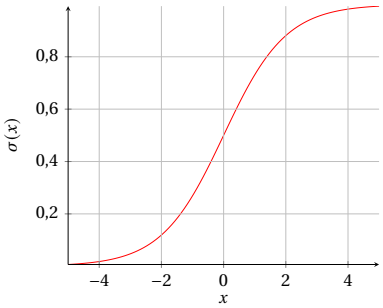


Figura 10: Gráfica de la sigmoide

aprendizaje y hacerlo más inestable [1].

Softmax

$$f(x)_i = \frac{e^{x_i}}{\sum_j^n e^{x_j}}, \quad x = (x_1, \ldots, x_n) \tag{5.2}$$

Cuando tenemos un problema de clasificación multidimensional, para unos valores de entrada, queremos saber a que clase es más probable que pertenezcan esos valores. Es en esos casos en los que usamos esta función de activación. Se trata de una generalización de la sigmoide a varias dimensiones, por lo que la imagen de cada componente x_i se puede interpretar como una probabilidad relativa. Es por ello que se suele utilizar en las neuronas de la capa de salida.

Tangente hiperbólica

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5.3}$$

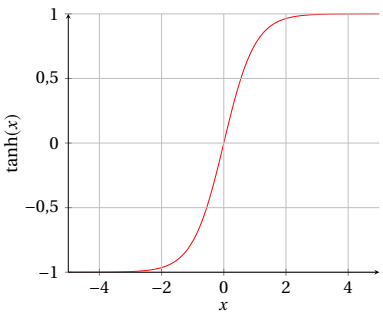


Figura 11: Gráfica de la tangente hiperbólica

Esta función es muy similar a la sigmoide, con la diferencia de que está centrada alrededor del origen y su imagen está acotada entre $[-1, 1]$. Esto hace que se puedan clasificar sus salidas como "muy positivas", "neutrales" y "muy negativas". Se suele utilizar en las neuronas de las capas ocultas puesto que la media de las imágenes es cercana a cero, ayudando a centrar los datos. Como la sigmoide, la tangente hiperbólica sufre del problema de desvanecimiento de gradiente [1].

ReLU

$$ReLU(x) = \max(0, x) \tag{5.4}$$

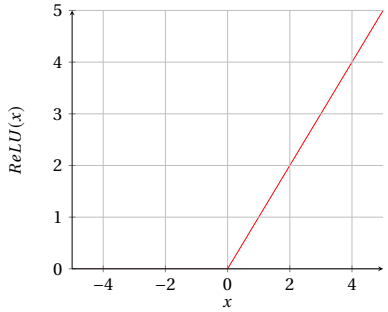


Figura 12: Gráfica de la ReLU

ReLU, también llamado Rectificador Lineal, es una función muy eficiente computacionalmente, puesto que únicamente activa las neuronas si (4.1) es positiva. Además, acelera la convergencia del gradiente debido a su propiedad lineal. Sin embargo, la ventaja de esta función es además su desventaja: para valores negativos, el gradiente es cero, por lo que en el algoritmo de retropropagación algunas neuronas no actualizan sus pesos, convirtiéndolas en "neuronas muertas" [1]. El Rectificador Lineal como función de activación, aunque de expresión muy distinta, tiene un fundamento biológico y matemático [26].

ELU

$$ELU(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}, \quad \alpha > 0 \tag{5.5}$$

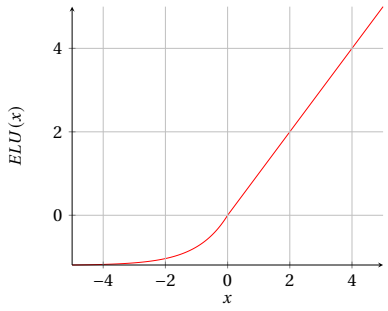


Figura 13: Gráfica de la ELU (α =1.2)

La función ELU, "Exponential Linear Unit" es una modificación del Rectificador Lineal que elimina el problema de las neuronas muertas. Sin embargo, aumenta el coste de computación al contener una exponencial y contiene un parámetro α fijo, que no "aprende" [1].

El problema de desvanecimiento de gradiente, mencionado en la función sigmoide y la tangente hiperbólica, es el siguiente: Puesto que el algoritmo de retropropagación usa la regla de la cadena, para funciones cuyo gradiente está en un intervalo cercano al cero, el valor del gradiente se va reduciendo exponencialmente conforme se va propagando, por lo que los pesos de las primeras capas se apenas se modifican, llegando incluso a permanecer intactos [37].

5.1 Retropropagación o "backpropagation"

El algoritmo de aprendizaje que se usa en la actualidad en las redes neuronales hacia delante está basado en la regla delta generalizada descrita en 1986 por Rumelhart, Hilton y Williams. Sin embargo, el concepto es muy anterior. Nace en los años 1960 dentro del campo de la teoría de control de la mano de Henry K. Kelley [16]. La versión actual fue propuesta por Yann LeCun en su tesis doctoral de 1987, *Modèles connexionnistes de l'apprentissage*. El objetivo de este algoritmo es calcular el gradiente de los pesos en cada una de las capas de la red, en orden descendente, y actualizar los pesos de manera que el error se minimice. Es decir, se trata de realizar, para cada peso w:

$$w_{\text{nuevo}} = w_{\text{antiguo}} - \eta \bigtriangledown E \tag{5.6}$$

donde η es la tasa de aprendizaje y E la función de costo.

5.2 Teorema de aproximación universal

Todos los modelos presentados pueden suscitar dudas sobre si realmente son capaces de aproximarse o incluso converger a la solución del problema que se plantea. Esto es, si un modelo es capaz de encontrar la regla o patrón subyacente en los datos con los que trabaja. En 1989, Kurt Hornik de la Universidad de Viena, y Maxwell Stinchcombe y Halbert White de la Universidad de San Diego en California demostraron que sí lo hacen.

El artículo "establece que las redes neuronales hacia delante comunes con tan sólo una única capa oculta usando una función de activación sigmoidal arbitraria son capaces de aproximar cualquier función boreliana medible que vaya de un espacio de dimensión finita a otro con el grado de precisión que se desee, siempre que haya suficientes capas ocultas. En este sentido, las redes neuronales hacia delante son aproximadores universales." [17].

Otra manera más intuitiva de enunciar lo que el artículo demuestra es la siguiente: Para cualquier función continua f en un conjunto K compacto, existe una red neuronal hacia delante con una única capa oculta que aproxima f con un error arbitrario $\epsilon > 0$ en K [20].

En 1991, Kurt Hornik demuestra que no es incluso necesario que la función de activación sea sigmoidal, sino que únicamente es necesario que la función sea acotada y no constante para que la red neuronal sea aproximadora universal. Además, da condiciones que aseguran que una red neuronal hacia delante es capaz de aproximar funciones y sus derivadas si poseen funciones de activación suficientemente suaves [14].

Una generalización más ocurre en 1993, cuando Moshe Leshno y otros demuestran que las redes neuronales hacia delante son aproximadoras universales si y sólo si tienen funciones de activación no polinómicas [22].

6 Redes neuronales profundas.

La redes neuronales profundas no son más que una red de neuronas con varias capas ocultas. En la actualidad no se ha dado una definición formal al respecto ni tampoco se tiene claro a partir de cuántas capas ya debiera considerarse «profunda».

Cabe hacerse la siguiente pregunta: Si con una capa, gracias al teorema de aproximación universal de las redes neuronales sabemos que es convergente, ¿por qué deberíamos utilizar más?

El uso de una capa fue el estándar hasta aproximadamente el año 2005, cuando de manera empírica se observaban resultados más favorables en redes con más capas [23] (de hecho este artículo que estudia sobre el la mejora que se produce introduciendo capas ocultas fue publicado en 2020). El motivo de esto es aún desconocido y es que como apuntan investigadores como Pablo Mesejo: los grandes teoremas sobre las redes neuronales aún están por descubrir.

6.1 Otros retos aún por resolver

Debido a los recientes y rápidos avances de las redes neuronales en los últimos tiempos, su uso en tanto productos comerciales e investigación está en expansión constante. Un ejemplo podría ser el uso de aprendizaje profundo para el desarrollo de sistemas de automoción autónomos. Es en estos casos donde uno de los mayores problemas sin resolver, la incertidumbre acerca de un resultado a partir de datos nuevos, es de crítica importancia.

Las redes neuronales obtienen sus resultados de manera inductiva, mientras que los algoritmos tradicionales siguen un método deductivo a partir de reglas y patrones. No se puede explicar qué hace una red neuronal en cada una de sus capas de manera general. Existen maneras de “explicar” los resultados de clasificadores, como LIME [18], pero explican el resultado específico, no la regla general.

Otro de los retos tiene que ver con los modelos en los que se usan las redes neuronales, que suelen carecer de especificaciones acerca de los requisitos y el diseño, además de una falta de robustez [11]. Esta falta de robustez es de vital problemática en los clasificadores de imágenes, en los que un pequeño cambio en las entradas, muchas veces imperceptible al ojo humano, provoca cambios radicales en el resultado [45].

7 Tipos de redes neuronales profundas

All models are wrong, but some are useful. George Box.

Ante un problema de aprendizaje automático, se debe tener en cuenta la naturaleza del mismo.:

- Tipo de aprendizaje: supervisado, no supervisado, clustering...
- Objetivo: clasificación, búsqueda de patrones...

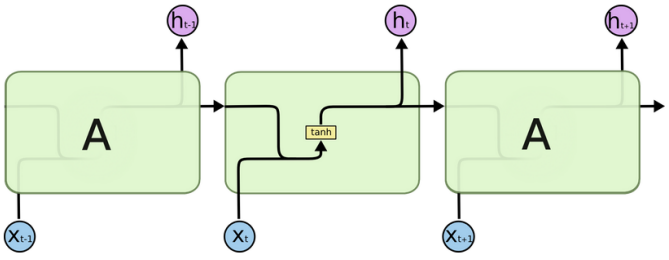
Como resultado de la diversidad de problemas que se pretenden resolver surge una gran variedad de redes neuronales profundas [35], aquí expondremos las más importantes.

Figura 14: John Hopfield (1933) [13] es un físico polaco. Trabajó en los laboratorios Bell, la universidad de Pricenton, la universidad de Berkeley y el Instituto Tecnológico de California en temas relacionados con la física, biología molecular y neurociencia.

7.1 Red recurrente, red de tensor neural o RNTN

Las redes recurrentes 16 fueron propuestas por primera vez por John Hopfield en 1982 y desarrolladas gracias a las aportaciones de David Rumelhart en 1986. Una idea intuitiva y bioinspirada sería pensar en el proceso de pensamiento humano, que realiza síntesis a partir de experiencias, pensamientos o información anterior. No toda pensamiento surge de la nada.
Estas redes utilizan grafos dirigidos o no dirigidos basados en secuencias temporales. Se utilizan para el reconocimiento de voz, el procesamiento de texto, el análisis de sentimientos, el análisis y el reconocimiento de la entidad de nombre.
Existen multitud de infraestructuras, como las CTRNN, basadas en sistemas de ecuaciones diferenciales ordinarias y que utilizaremos de ejemplo.

Figura 15: David Rumelhart (1942 - 2011) [29] fue un psicólogo estadounidense que trabajó en el campo de la modelización matemática de la psicología, la inteligencia artificial simbólica y el conexionismo.



The repeating module in a standard RNN contains a single layer.

Figura 16: Esquemas de las RNN, 18

Para cada neurona i-ésima que recibe como activación y_i su tasa de cambio viene dada por:

$$\tau_i \dot{y}_i = -y_i + \sum_{j=1}^{n} w_{i,j}\sigma(y_j - \Omega_j) + I_i(t), \tag{7.1}$$

donde:

- τ_i es una constante del tiempo postsináptico.
- y_i nodo activación postsináptica.
- $\dot{y}_i$ Ratio de activación del nodo psináptico.
- $w_{j,i}$ peso de las conexiones.
- σ sigmoide.
- y_i nodo activación presináptica.
- Ω_j sesgo de la conexión presináptica.
- $I_i(t)$ input del nodo (si es que hay).

Destacan además las redes LTSM que tratan de resolver uno de los problemas fundamentales de las redes recurrentes: que los pesos anteriores pesen demasiado en el modelo impidiendo actualizaciones útiles nuevas.

7.1.1 Redes LSTM

[3]
Las redes *Long Short Term Memory* [32] capaces de aprender de experiencias a largo plazo. Son de gran utilidad en procesado de texto, un ejemplo sería, si quisiéramos predecir la palabra a continuación de *Las nubes están en el* , necesitaremos información de las palabras anteriores a la que queremos predecir.
Puede verse un esquema de su diseño en la figura 17.
La idea central del diseño es la siguiente:
La existencia de una célula de estado C_t (en la iteración t), para calcular C_t a partir de C_{t-1} y la etiqueta son necesarios los siguientes cálculo auxiliares

$$f_t = \sigma(W_f[h_{t_1}, x_t] + b_f) \tag{7.2}$$

$$i_t = \sigma(W_i[h_{t_1}, x_t] + b_i) \tag{7.3}$$

$$\tilde{C}_t = tanh(W_c[h_{t_1}, x_t] + b_c) \tag{7.4}$$

Hasta que finalmente obtenemos

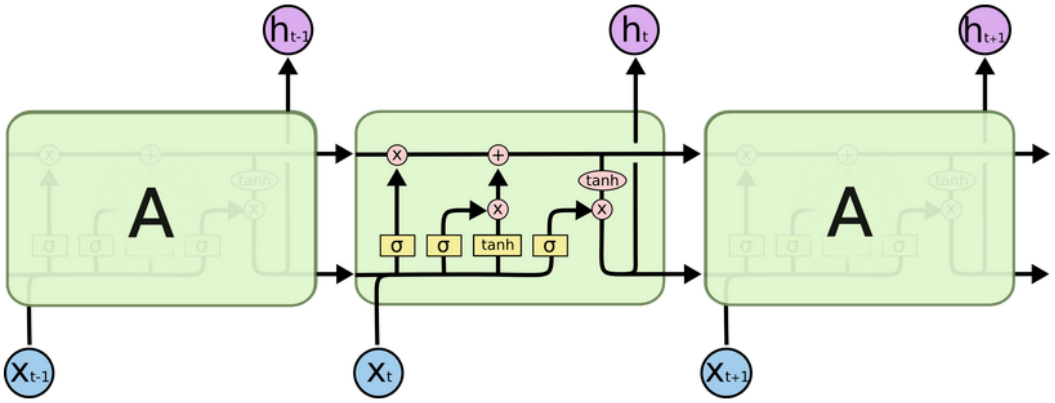
$$\tilde{C}_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{7.5}$$

Para la salida del modelo h_t se calcula como

$$\tilde{h}_t = o_t * tanh(C_t) \tag{7.6}$$

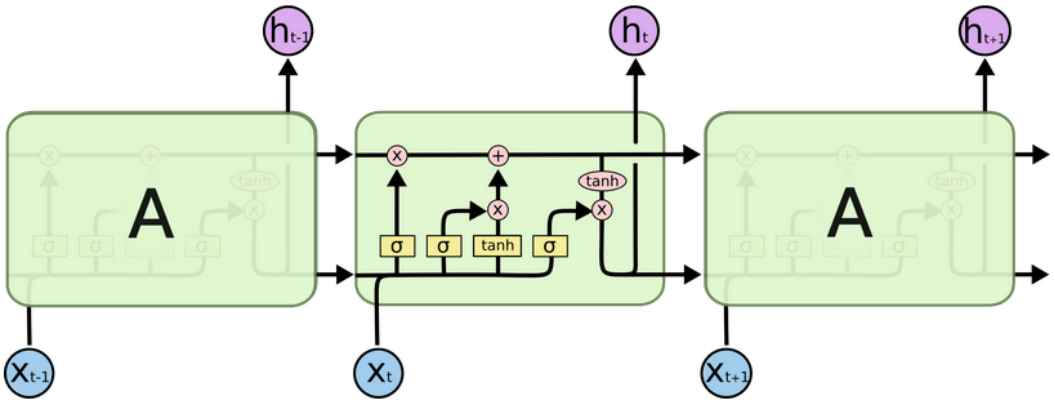
Donde

$$\tilde{h}_t = \sigma((W_o[h_{t_1}, x_t] + b_0)) \tag{7.7}$$



The repeating module in an LSTM contains four interacting layers.

Figura 17: Esquemas de las LSTM 18



The repeating module in an LSTM contains four interacting layers.

Figura 18: Esquemas de las LSTM

7.1.2 GRU

Otra variable creada con la misma idea que LSTM son las GRU, *Gated recurrent unit* que fueron introducidas en 2014 por Kyunghyun Cho.
Un esquema de GRU es 19 [24] que se diferencia de las LSTM por tener menor número de parámetros y carecer de *gated output*.

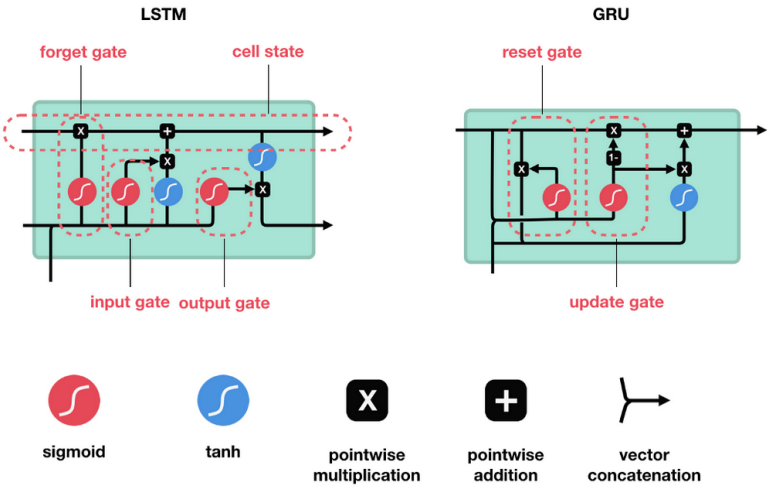


Figura 19: Esquema GRU

Se utilizan para ciertas tareas como modelos de música polifónicos, reconocimiento de voz y procesamiento de lenguaje natural. [25].
(La formalización matemática se puede hacer siguiente el mismo esquema que la anterior o consultando directamente [25]).

Figura 20: Yan LeCun nació en Francia en 1961. Ha trabajado en el aprendizaje automático, la visión por computador, los robot móviles y la neurociencia computacional. En 2018 ganó el premio Turing y es considerado junto con Geoffrey Hinton y Yoshua Begio el padrino del aprendizaje profundo. Actualmente trabaja como investigador en inteligencia artificial en Meta Platforms (antiguo Facebook).

7.2 Red convolucional o CNN

Las redes neuronales convolucionales (CNN), también conocidas como ConvNets, son redes neuronales profundas para el procesamiento y detección de objetos. Fueron desarrolladas pro Yann LeCun en 1988, que él llamó LeNet y las utilizó para el reconocimiento de características como el código ZIP y dígitos.

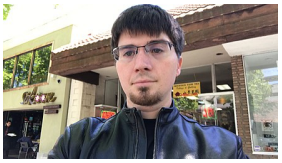


Figura 21: Ian Goodfellow (1986) es un ingeniero informático estadounidense. Trabajó en Google como parte del equipo de Google Brain, después pasó a formar parte de OpenAI y actualmente trabaja para Apple.

7.3 Red generativa antagónica o GAN

Este tipo de redes se usan para la generación de contenido multimedia. Por ejemplo, la reconstrucción de imágenes y la generación de rostros que parezcan reales [15]. Fueron creadas por Ian Goodfellow y sus compañeros en junio de 2014.
Su diseño es el siguiente: Se tienen dos redes neuronales profundas que son «adversarias» en un juego de suma cero. Una de ellas realiza la tarea de *Discriminador* y la otra la tarea de *generador* .

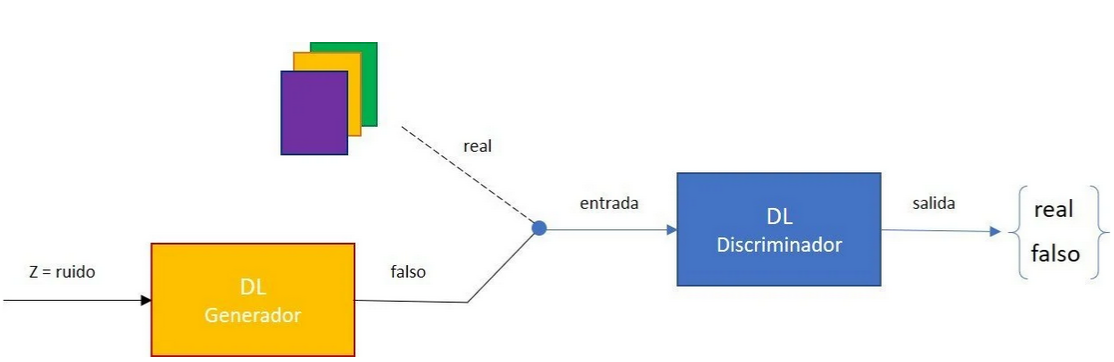


Figura 22: Imagen explicativa del GAN

Para entrenar esta arquitectura se necesita un conjunto de datos reales, de esta forma cuando le pasemos una foto real al discriminador esta deberá de ser tomada como correcta. El generador deberá actualizar sus pesos hasta que el discriminador tome sus fotos generadas como buenas.

8 Optimización de Redes Neuronales

A la hora de calcular los pesos del perceptrón, podemos seguir el método de regresión lineal esto es, minimizar el error cuadrático medio mediante derivación. Sin embargo, este método tiene la debilidad de no ser eficiente para dimensiones elevadas, especialmente en casos en los que la función de coste (error cuadrático medio) tiene más de un mínimo local. De aquí surge la necesidad de buscar otros métodos, como el método del descenso de gradiente.

Figura 23: Augustin Louis Cauchy (1789-1857) [39] fue un matemático francés, uno de los más prolíficos de la historia y contribuyó en todas las áreas de la matemática de su época.

Figura 24: Jacques Salomon Hadamard (1865 - 1963) [43] fue un matemático francés. Trató temas de física matemática, colaboró en el establecimiento de las bases del análisis infinitesimal y desarrolló el teorema sobre el valor absoluto de un determinante.

Figura 25: Haskell Brooks Curry (1900-1982) [42] fue un matemático y lógico estadounidense, especializado en teoría de sistemas y lógica combinatoria, fundamento de los lenguajes funcionales.

8.1 Método de regresión lineal

Este método se basa en la minimización de el error cuadrático medio entre la matriz asociada a la proyección $h(x) = w^T x$ e y, la etiqueta ideal, donde $x \in \mathbb{R}^{N\times(d+1)}$ es la matriz característica y $N \in \mathbb{N}$ el tamaño del conjunto de entrenamiento.

$$E_{in}(w) = \frac{1}{N}\sum_{n=1} N(w^T x_n - y)^2 = \frac{1}{N}\|Xw - y\|^2 \tag{8.1}$$

donde $\|.\|$ es la norma euclídea de un vector.

Como $E_{in}(w)$ es diferenciable podemos calcular los w que minimizan E_{in}.

$$\nabla E_{in}(w) = \frac{2}{N}(X^T Xw - X^T y) = 0 \tag{8.2}$$

Finalmente, para conseguir $\nabla E_{in}(w) = 0$, buscamos w cumpliendo que

$$X^T Xw = Xy \tag{8.3}$$

Si $X^T X$ es invertible, $w = X^\dagger y$, donde $X^\dagger = (X^T X)^{-1}$ es la *pseudo inversa* de X. El w resultante es el único óptimo que minimiza E_{in}. De lo contrario, la pseudo inversa sí podría ser definida, pero la solución no sería única.

En la práctica, $X^T X$ suele ser invertible porque N es mucho mayo que $d+1$, así que habrá $d+1$ vectores linealmente independientes x_n.

8.2 El descenso de gradiente

El descenso de gradiente [28] [7] es uno de los algoritmos más populares para optimizar las redes neuronales. Su autoría se atribuye a Cauchy, quien fue el primero en sugerirlo en 1847. Hadamard propuso de forma independiente un método similar en 1907. La convergencia de este método para optimización de funciones no lineales fue estudiado por Haskell Curry en 1944. El método fue ampliamente estudiado y utilizado en las décadas posteriores [41].

Este método consiste en minimizar una función objetivo $J(\theta)$ parametrizada por los parámetros $\theta \in \mathbb{R}^d$ de un modelo mediante la actualización de los parámetros en la dirección opuesta a la del gradiente de la función objetivo $\nabla_\theta J(\theta)$. La tasa de aprendizaje η determina el número de pasos necesarios hasta alcanzar un mínimo local.

Intuitivamente, seguimos la dirección de la pendiente de la superficie creada por la función objetivo (normalmente, el error cuadrático medio) hasta que alcanzamos un valle, dando pasos de una longitud determinada por η.

8.2.1 Variantes del Descenso de Gradiente

Descenso de Gradiente Batch

Calcula el gradiente de la función de coste con respeto a los parámetros θ para el conjunto de entrenamiento completo:

$$\theta = \theta - \eta \nabla_\theta J(\theta). \tag{8.4}$$

Tiene un alto coste computacional y puede llegar a ser muy lento.

Descenso de Gradiente Estocástico

Realiza la actualización de un parámetro para cada ejemplo de entrenamiento $x^{(i)}$ y etiqueta $y^{(i)}$.

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{8.5}$$

Descenso de Gradiente Mini-Batch

Realiza una actualización para cada subconjunto de n ejemplos de entrenamiento:

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{8.6}$$

Con frecuencia se utiliza una combinación del método estocástico y mini Batch

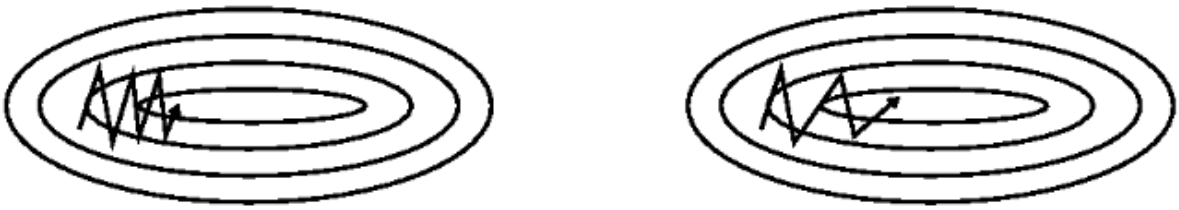
8.3 Algoritmos de Descenso de Gradiente

Las variantes del descenso de gradiente presentadas se pueden refinar en algoritmos más eficientes, siguiendo su filosofía.

Además, el valor que le demos a la tasa de aprendizaje η es relevante. Cada algoritmo concreto le dará un valor, que puede ser constante o variable a conveniencia.

8.3.1 Momentum

El algoritmo Momentum está basado en el Descenso de Gradiente Estocástico. Intuitivamente, se basa en un pelota rodando cuesta abajo. Cuanto más rueda hacia una zona de menor altura, más rápido avanza. Esto lo aplicamos a la actualización de los parámetros: cuántas más actualizaciones de los parámetros que nos acercan a un mínimo local hacemos, más abruptas son.



(a) SGD without momentum (b) SGD with momentum

Figura 26: Descenso de Gradiente Estocástico básico (izq) y algoritmo Momentum (dcha).

9 Otras grandes figuras

9.1 Geoffrey Hinton

Figura 27: Geoffey Hinton.

Geoffrey Hinton (1947) es un psicólogo cognitivo e informático británico y canadiense [40].

En 1986 publicó junto a David Rumelhart y Ronald J. Williamsel el artículo que popularizaría el algoritmo de *backpropagation* para redes neuronales multicapa [8].

En 2018 se le concedió el premio Turing.

Actualmente trabaja a tiempo parcial en la universidad Toronto (Canadá) y en el equipo de investigación *Google Brain*, un grupo de trabajo de Google dedicado al aprendizaje profundo y la creación y mejora de herramientas como TensorFlow y Google Translate, así como la mejora de imágenes y el aprendizaje de sistemas de encriptación simétrica usando redes generativas adversativas [4].

9.2 Yoshua Bengio

Figura 28: Yoshua Bengio.

Yoshua Bengio [2] es un informático canadiense que investiga sobre deep learning, traducción automática neural, redes generativas, auto encoders, modelos neurales del lenguaje y el meta aprendizaje.

Desde 2007 trabaja para Facebook aunque también es investigador en la universidad de Montréal y director del *Montreal Institute for Learning Algorithms*.

En 2018 se le concedió el premio Turing.

10 Conclusiones

El aprendizaje automático es un área de la informática joven, con una breve pero intensa historia. A pesar de basarse en conocimientos matemáticos sobradamente conocidos desde hace siglos, no ha sido hasta hace pocas décadas cuando se han puesto intensamente en práctica.

Al igual que el resto de disciplinas de la informática, su rápido desarrollo en un corto intervalo de tiempo ha supuesto una reinvención de la vida y relaciones humanas con respecto a tal y como las conocíamos.

Pocas ciencias han progresado tan velozmente como esta, tanto es así que la teoría matemática disponible empieza a no ser suficiente, lo que abre nuevas líneas de investigación.

Además, el gran impacto del aprendizaje automático en la sociedad ha repercutido también en la economía global. Ahora son las empresas tecnológicas las que lideran la economía y sus empleados, la nueva vanguardia científica.

Con la realización de este trabajo nos hemos acercado a los entresijos de este campo que desconocíamos y nos ha aportado una visión histórica más profunda del desarrollo de la ciencia.

Referencias

[1] Pragati Baheti. *12 Types of Neural Network Activation Functions: How to Choose?* 2021. URL: https://www.v7labs.com/blog/neural-networks-activation-functions (visitado 02-12-2021).

[2] Wikipedia. Yoshua Bengio. URL: https://en.wikipedia.org/wiki/Yoshua_Bengio (visitado 13-12-2021).

[3] Colah's blog. *Undertanding LSTM*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visitado 27-08-2015).

[4] Google Brain. *Google brain*. 2021. URL: https://en.wikipedia.org/wiki/Google_Brain#Projects (visitado 29-11-2021).

[5] Pablo Caceres. *Introduction to Neural Network Models of Cognition*. 2021. URL: https://com-cog-book.github.io/com-cog-book/intro.html (visitado 30-11-2021).

[6] Vladimir Cherkassky y Filip Mulier. *Learning From Data: Concepts, Theory, and Methods*. 2.ª ed. John. Wiley & Sons, 2007. Cap. 1,2.

[7] Dot Csv. *¿Qué es el Descenso del Gradiente? Algoritmo de Inteligencia Artificial | DotCSV*. URL: https://www.youtube.com/watch?v=A6FiCDoz8_4 (visitado 13-12-2021).

[8] Ronald J. Williams David E. Rumelhart Geoffrey E. Hinton. *Parallel distributed processing: Explorations in the microstructure of cognition*. MIT Press, 1986.

[9] Michael R.W. Dawson. "Minds And Machines: Connectionism And Psychological Modeling." En: Blackwell Publishing, 2004, pág. 166.

[10] Dr. Michael R.W. Dawson y Dr. David A. Medler. *Dictionary of Cognitive Science*. 2011. URL: http://www.bcp.psych.ualberta.ca/~mike/Pearl_Street/Dictionary/contents/G/generalizeddr.html (visitado 30-11-2021).

[11] Hirotoshi Yasuoka y Toshihiro Nakae Hiroshi Kuwajima. *Open Problems in Engineering Machine Learning Systems and the Quality Model*. URL: https://arxiv.org/abs/1904.00001v1.

[12] Hisour. *Nocción del aprendizaje automático*. 2021. URL: https://www.hisour.com/es/machine-learning-42773/ (visitado 27-11-2021).

[13] Hopfield. *Carrera de Hopfield*. 2021. URL: http://www.scholarpedia.org/article/User:Hopfield (visitado 28-11-2021).

[14] Kurt Hornik. "Approximation Capabilities of Muitilayer Feedforward Networks". En: *Neural Networks* 4 (1991), págs. 251-257.

[15] IA artificial inteligent. *Explicación de las redes Neuronales Generativas Adversativas*. 2021. URL: https://www.iartificial.net/redes-neuronales-generativas-adversarias-gans/ (visitado 29-11-2021).

[16] Henry J. Kelley. "Gradient Theory of Optimal Flight Paths." En: *ARS Journal* 30.10 (1960), págs. 947-954.

[17] Maxwell Stinchcombe y Halbert White Kurt Hornik. "Multilayer Feedforward Networks are Universal Approximators." En: *Neural Networks* 2 (1989), págs. 359-366.

[18] S. Singh y C. Guestrin M. T. Ribeiro. "Why Should I Trust You?" Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16." En: (2016).

[19] MacTutor. *Marshall Stone*. 2021. URL: https://mathshistory.st-andrews.ac.uk/Biographies/Stone/ (visitado 28-11-2021).

[20] Daniel McNeela. *The Universal Approximation Theorem for Neural Networks*. 2017. URL: https://mcneela.github.io/machine_learning/2017/03/21/Universal-Approximation-Theorem.html (visitado 30-11-2021).

[21] Tom Mitchell. *Machine Learing*. McGraw, 1997. URL: https://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/mlbook.html.

[22] Allan Pinkus y Shimon Schocken Moshe Leshno Vladimir Ya. Lin. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". En: *Neural Networks* 6 (1993), págs. 861-867.

[23] Noreen Jamil Muhammad Uzair. "Effects of Hidden Layers on the Efficiency of Neural networks". En: (2020).

[24] Michael Phi. *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (visitado 23-07-2018).

[25] Michael Phi. *Wikipedia gru explanation*. URL: https://en.wikipedia.org/wiki/Gated_recurrent_unit (visitado 13-12-2021).

[26] H. Sebastian Seung Richard H.R. Hahnloser. *Permitted and forbidden sets in symmetric threshold-linear networks*. Neural information processing systems foundation, 2001.

[27] Frank Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storageand Organization in the Brain." En: *Psychological Review* 65.6 (Abril de 1958), págs. 386-408.

[28] Sebastian Ruder. "An overview of gradient descent optimization algorithms". En: (2017).

[29] Rumelhart. *Biografía de Rumelhart*. 2021. URL: https://web.archive.org/web/20131030220027/http://rumelhartprize.org/?page_id=10 (visitado 29-11-2021).

[30] Wikipedia. Arthur Samuel. URL: https://en.wikipedia.org/wiki/Arthur_Samuel (visitado 30-11-2021).

[31] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". En: *Neural Networks* 61 (2015), págs. 85-117.

[32] Jürgen Schmidhuber Sepp Hochreiter. "Long short -term Memory". En: (1997).

[33] Sandro Skansi. "Introduction to Deep Learning". En: Springer International Publishing, 2018.

[34] Karl-Georg Steffens y George Anastassiou. "The history of approximation theory. From Euler to Bernstein". En: *The History of Approximation Theory: From Euler to Bernstein* (ene. de 2006), págs. 1-219. DOI: 10.1007/0-8176-4475-X.

[35] Código fuente. Vlog de divulgación informática. *Redes neuronales profundas. Tipo y Caractarerísticas*. 2021. URL: https://www.codigofuente.org/redes-neuronales-profundas-tipos-caracteristicas/ (visitado 28-11-2021).

[36] Warren S. McCulloh y Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". En: *Bulletin of Mathematical Biophysics* 5 (1943), págs. 115-133.

[37] Chi-Feng Wang. *The Vanishing Gradient Problem*. 2019. URL: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484 (visitado 02-12-2021).

[38] Wikipedia. Karl Weierstrass. URL: https://es.wikipedia.org/wiki/Karl_Weierstrass (visitado 30-11-2021).

[39] Wikipedia. *Agoustin Louis Cauchy*. URL: https://es.wikipedia.org/wiki/Augustin_Louis_Cauchy (visitado 13-12-2021).

[40] Wikipedia. *geoffrey-hinton*. 2021. URL: https://en.wikipedia.org/wiki/Geoffrey_Hinton#cite_note-schmidhuber-15 (visitado 29-11-2021).

[41] Wikipedia. *Gradient Descent*. URL: https://en.wikipedia.org/wiki/Gradient_descent (visitado 13-12-2021).

[42] Wikipedia. *Haskell Curry*. URL: https://es.wikipedia.org/wiki/Haskell_Curry (visitado 13-12-2021).

[43] Wikipedia. *Jacques Hadamard*. URL: https://es.wikipedia.org/wiki/Jacques_Hadamard (visitado 13-12-2021).

[44] Hsuan-Tien Lin Yaser S. Abu-Mostafa Malik Magdon-Ismail. *Learning From Data. A short Course*. AMLbook.com, 2012.

[45] Fuyuki Ishikawa y Yutaka Matsuno. "Continuous Argument Engineering: Tackling Uncertainty in Machine Learning Based Systems". En: *Lecture Notes in Computer Science* 11094 (2018), págs. 14-21.