# Introducción a las Redes Neuronales con Clojure

Miguel Angel Piña Avelino

23 de diciembre de 2016

Primera Edición

Texto redactado en la ciudad de México, 2016

# Prefacio

El presente documento es escrito con la intención de proveer un enfoque en la implementación de redes neuronales a través de software, esto debido a que la implementación en hardware no siempre es posible y suele estar fuera del alcance de los que buscan entrar en el mundo de las redes neuronales.

El enfoque de la implementación es a través de un lenguaje funcional para definir la mayoría de las operaciones entre redes como funciones, esto para tratar de mantener un paralelismo entre la modelación matemática de la red neuronal y la implementación en código.

El lenguaje elegido para realizar todas estas operaciones es Clojure, el cuál a mi parecer, las siguientes razones lo hacen un lenguaje adecuado para trabajar.

- **Filosofía** Clojure hace algunas muy buenas decisiones de diseño. Por ejemplo estructuras de datos inmutables, así como la tendencia a implementar funciones puras (i.e. no tienen efectos secundarios).
- **Concurrencia** Clojure es realmente bueno con problemas de concurrencia, y varias bibliotecas lo hacen mucho mejor.
- **Interoperabilidad** Clojure puede hacer uso de bibliotecas de Java existentes.
- **Desarrollo** La creación de código y el prototipado rápido permiten una mejor experiencia al programar.
- **Macros y DSL** Permite extender el lenguaje de forma muy sencilla a través de macros y con ello crear lenguajes de dominio específico (DSL - Domain Specific Languages)

Una vez elegido el lenguaje y la explicación del porqué ha sido elegido, explicaremos brevemente los tipos de redes neuronales que vamos a construir. Generalmente las redes neuronales se diferencian por el tipo de aprendizaje que usan, siendo el aprendizaje supervisado y el no supervisado los más comunes. Existen también soluciones híbridas las

cuales mencionaremos brevemente en los capítulos dedicados a las redes neuronales. En el primer capítulo se hará una exposición un poco más detallada de porqué Clojure es una buena opción.

En el segundo capítulo hablaremos de las siguientes redes:

- *Perceptrón Simple*
- *Adalina*
- *Perceptrón multicapa y backpropagation*
- *Mapas autoorganizados (Redes de Kohonen)*
- *Redes de Hopfield*

Así como su base matemática y algunas de las aplicaciones que tienen.

En el tercer capítulo haremos una breve introducción a las distintas características de Clojure que serán fundamentales conocer al momento de implementar las redes neuronales mencionadas en el capítulo 2.

En el cuarto capítulo explicaremos como realizar la implementación de las distintas redes neuronales. Continuando con el quinto capítulo en el cuál veremos una aplicación real de las redes neuronales y la comparación con otras técnicas.

Y finalizamos el documento dando conclusiones sobre los temas desarrollados en el libro. Espero que el presente documento sea de su agrado y sirva como apoyo para la comprensión de las redes neuronales artificiales.

# Índice general

# Índice de figuras

## ÍNDICE DE FIGURAS

*I think the brain is essentially a computer and consciousness is like a computer program. It will cease to run when the computer is turned off. Theoretically, it could be re-created on a neural network, but that would be very difficult, as it would require all one's memories.*

Stephen Hawking

CAPÍTULO

# 1

# Introducción

La motivación de escribir un libro sobre redes neuronales nace a partir de que es un tema que ido desarrollando como parte de mi formación como Científico de la Computación, así como ser uno de los modelos de *Aprendizaje Máquina* que más han sido investigados en los últimos años y que más fascinación me han dado.

Este libro no pretende ser más que una guía para comprender como implementar redes neuronales artificiales usando un lenguaje funcional (Clojure), tratando de reproducir resultados como el perceptrón simple [Ros58] o los mapas autoorganizados de Kohonen [Koh82, Koh01]. Esto se pretende llevar a partir del estudio matemático de las distintas redes y enfocarlos en el desarrollo de aplicaciones de software.

## 1.1 ¿Qué es una red neuronal artificial?

Las redes neuronales artificiales son un enfoque computacional basado en un conjunto de unidades de procesamiento que están interconectadas entre sí de forma similar en que las conexiones biológicas del cerebro están conectadas y se usan para resolver distintos tipos de problemas relacionados con el aprendizaje, clasificación o clusterización de información.

Generalmente cada neurona artificial (en adelante sólo *neurona*), está conectada con otras y sus conexiones pueden ser activar o inhibir otras neuronas. Cada neurona determinar estas activaciones o inhibiciones a partir de una función matemática

que está definida en término de sus entradas (conexiones), las cuáles al evaluarse en la función y llegar a cierto umbral disparan la activación o inhibición de la neurona, siendo esta activación o inhibición la salida de la función con valores discretos o continuos.

A la forma en que se encuentran conectadas las neuronas entre sí, lo vamos a denominar *Arquitectura de la Red*. Esta "Arquitectura" es el modo en que las neuronas pueden compartir información. Las formas habituales en que se distribuyen las neuronas es a través de capas de neuronas, las cuales se agrupan en en unidades estructurales. Distinguiremos tres tipos de capas, de entrada, de salida y oculta.

- **Capa de entrada**: Corresponde con la capa sensorial, encargada de recibir datos o señales del entorno.
- **Capa de salida**: Es el conjunto de neuronas que proveen los resultados del procesamiento de los datos o señales de la red.
- **Capa oculta**: Es una capa que no está conectada con el entorno, generalmente sirve para proveer representaciones internas del entorno.

Procederemos ahora a dar una definición formal de lo que es una red neuronal:

**Definición 1.1.** *Una red neuronal es un grafo dirigido con las siguientes características:*

1. *A cada vértice i se asocia a una variable de estado* $x_i$
2. *A cada conexión* $(i, j)$ *de los vértice i y j se le asocia un peso* $w_{ij} \in \mathbb{R}$
3. *A cada vértice i se le asocia un umbral* $\theta_i$
4. *Para cada vértice i se define una función* $f_i(x_j, w_{ij}, \theta_i)$ *que va a depender de las conexión, del umbral y de los estados de los j a el conectados. Esta función define el nuevo estado del vértice.*

Habitualmente, las redes neuronales se clasifican en dos tipos de operación o aprendizaje, siendo estos el aprendizaje supervisado y el aprendizaje no supervisado, los cuáles hablaremos con mayor detalle en las siguientes secciones.
**Incluir imágenes para ilustrar el contenido anterior**

## 1.2 Aprendizaje de la red neuronal

En el contexto de las redes neuronales se entiende como **aprendizaje** al proceso de adaptación de la red neuronal en el que adapta sus parámetros internos de la red a través de estímulos externos. Comúnmente el aprendizaje consiste en determinar un conjunto de pesos sinápticos que permita a la red realizar correctamente el procesamiento pretendido.

El modo convencional de aprendizaje es definiendo una regla de aprendizaje $\Delta w_{ij}$ definida en términos de la conexión $w_{ij}$ que representa la conexión del peso de la neurona presináptica de la neurona $j$ con la postsináptica $i$ en la iteración $t$, quedando de la siguiente forma:

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{1.1}$$

El proceso de aprendizaje es iterativo, actualizándose los pesos hasta que la red neuronal alcanza el rendimiento de procesamiento deseado.

Los modelos comunes de aprendizaje son el aprendizaje supervisado y el aprendizaje no supervisado. Existen otros como el aprendizaje híbrido o el aprendizaje reforzado que no van a ser tratados aquí.

### 1.2.1 Aprendizaje supervisado

De manera informal se puede hablar del aprendizaje supervisado como la presentación a la red de patrones, junto a la salida deseada u objetivo, de tal modo que la red ajuste sus pesos hasta que la salida sea tan similar como la que fue presentada previamente. De este modo la red es capaz de estimar relaciones entrada/salida sin proponer una cierta forma funcional de inicio.

### 1.2.2 Aprendizaje no supervisado

El aprendizaje no supervisado se puede describir de forma general como la estimación de la función de densidad de probabilidad $p(x)$ que va a describir la distribución de patrones x correspondientes al espacio de entrada $\mathbb{R}^n$ a partir de ejemplos.

En este tipo de aprendizaje, se presenta a la red una gran cantidad de patrones sin que nosotros le indiquemos cual es la salida deseada. La red a través de la regla de aprendizaje va intentar extraer rasgos del conjunto de entrada.

## 1.3 Breve historia de las redes neuronales

Durante mucho tiempo ha existido un encanto en tratar de modelar los distintos procesos cognitivos del cerebro a través de mecanismos de hardware. Ejemplos de estas primeras construcciones las podemos encontrar durante la Segunda Guerra Mundial con la construcción de la máquina Enigma [**Agregar referencia**] utilizada para la codificación de mensajes entre las tropas alemanas.

En 1943, el neurofisiólogo Warren McCulloch y el matemático Walter Pitts escriben un artículo sobre el posible funcionamiento de las neuronas cerebrales modelándola como una red neuronal simple usando circuitos electricos [MP43].

En 1950, Alan Turing propone una pregunta que dice: «¿Pueden las máquinas pensar?» [Tur50] influyendo en la visión de la inteligencia artificial como una imitación del comportamiento humano, pero esto no tuvo el exito esperado, si no siendo más dominante la visión de que la inteligencia artificial debe ser más como un comportamiento racional.

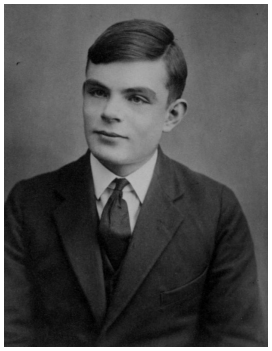

**Figura 1.1:** Alan Turing a los 16 años

En 1958, Frank Rossenblatt propone el **Perceptrón Simple** como un dispositivo electrónico construido para mantener los principios biológicos de una neurona y la habilidad para aprender. En 1962 publicó un libro llamado «Principles of neurodynamics. Perceptrons and the theory of brain mechanisms» en que extendió y desarrolló sus investigaciones sobre el perceptrón, utilizando este libro como apoyo para sus cursos [Ros58].

En 1962, Bernard Widrow y Marcian Hoff desarrollan modelos llamados **ADALINE** y **MADALINE**, donde ADALINE fue desarrollada para reconocer patrones binarios. Mientras que MADALINE fue la primer red neuronal aplicada a un problema de la vida real, utilizando un filtro adaptativo para eliminar el echo en las lineas telefónicas [WH62, WH$^+$60].

Después de estos trabajos, otros investigadores trabajaron en crear nuevas arquitecturas para los modelos existentes, llegando a resultados como las redes neuronales multicapa o con propagación hacia atrás (en adelante *backpropagation*). Con estos nuevos resultados se llegó a los modelos de redes neuronales no supervisadas.

En 1982, el interés por las redes neuronales fue renovado con el trabajo de John

Hopfield, usando un enfoque de crear redes neuronales con conexiones bidireccionales (originalmente eran grafos unidireccionales) [Hop82].

En ese mismo año, Reilly y Copper presentan una "Red híbrida" con múltiples capas, donde cada capa tenía una estrategia de resolución de problemas diferente [RCE82].

Otro modelo presentado en ese mismo año fue hecho por Teuvo Kohonen, presentando el modelo de mapas autoorganizados, inspirándose en el funcionamiento del cortex frontal del cerebro, permitiendo que estos mapas sean adaptativos y usados como herramientas para el descubrimiento de características en conjuntos de datos desconocidos[Koh82, Koh01].

Además de esto, han surgido otros modelos como las redes neuronales convolucionales [HW68, Fuk80, Beh03, LBBH98, GLM88] la cuál un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria de un cerebro biológico.

**Figura 1.2:** Teuvo Kohonen

**Agregar información sobre las redes neuronales de Google**

Con esto se pretende brindar un pequeño contexto histórico e informativo sobre las redes neuronales. En lo que resta del capítulo hablaremos brevemente acerca del lenguaje de programación Clojure.

## 1.4 Clojure

Clojure es un lenguaje dinámico, de proposito general, que combina el enfoque y el desarrollo interactivo de un lenguaje de *scripting* con una eficiente y robusta infraestructura para programación multihilo.

Además, Clojure es un lenguaje dialecto de Lisp y comparte con Lisp la filosofia del código como datos y un sistema poderoso de macros. Es predominantemente un lenguaje funcional con características tales como un conjunto rico estructuras de datos inmutables y persistentes. Cuando es necesario la mutabilidad, Clojure ofrece un sistema transaccional de memoria y un sistema de agentes reactivos que

garantiza diseños multihilo limpios y correctos.

La interfaz principal de desarrollo de Clojure es a través de REPL (Read-Eval-Print-Loop), la cuál es una simple consola que permite escribir comandos y examinar sus resultados. Ejemplos sencillos de código son:

```
(def x 6)
-> #'user/x
(def y 36)
-> #'user/y
(+ x y)
-> 42
```

Clojure tiene los siguientes tipos de datos: enteros de precisión arbitraria, cadenas, ratios, caracteres, símbolos, keywords.

```
(* 12345678 12345678)
-> 152415765279684
"string"
-> "string"
22/7
-> 22/7
3.14159
-> 3.14159
\a
-> \a
'symbol
-> symbol
:keyword
-> :keyword
;a comment
```

### 1.4.1 Programación funcional

Clojure es un lenguaje de programación funcional. Va a proveer herramientas para evitar estado mutable, provee funciones de primera clase y enfatiza en la iteración recursiva en lugar de ciclos con efectos laterales. Sin embargo, Clojure no es puro, ya que no te forza a que tu programa sea referencialmente transparente y no se esfuerza en tener programas que sean "demostrables".

La filosofía detrás de Clojure es que la mayor parte de los programas deberían de ser funcionales y los programas que son funcionales son más robustos.

Varias de estas características serán detalladas en el capítulo 3, donde se explicará a detalle todo lo anterior y con mayor rigor que el aquí expuesto.

# 1. INTRODUCCIÓN

*Machine consciousness refers to attempts by those who design and analyse informational machines to apply their methods to various ways of understanding consciousness and to examine the possible role of consciousness in informational machines.*

Igor Aleksander

CAPÍTULO

# 2

# Redes Neuronales

En este capítulo estudiaremos los conceptos principales de varias redes neuronales diviéndolas en dos tipos: Redes neuronales supervisadas y redes neuronales no supervisadas. Esta distinción está dada por la forma en que se efectúa el aprendizaje dentro de las redes neuronales.

En el capítulo 1 hablamos brevemente de estos tipos de aprendizaje, donde mencionamos que el **aprendizaje supervisado** es en el que se presenta ejemplos y salidas deseadas para entrenar a la red y el **aprendizaje no supervisado** es el que de forma autónoma, la red va descubriendo características acerca de los datos que le son presentados.

A continuación explicaremos a detalle estos tipos de aprendizaje y mostraremos las redes neuronales que están clasificadas bajo estos tipos.

## 2.1 Redes Neuronales Supervisadas

### 2.1.1 Perceptrón simple

El perceptrón simple fue introducido por Frank Rossenblatt a finales de los años 50, modelándolo como un dispositivo que mantuviera los principios biológicos de una neurona y su habilidad para aprender. El modelo implementado por el perceptrón simple es unidireccional, compuesto por dos capas de neuronas, una sensorial o

de entrada y otra de respuesta o salida. La operación básica de esta red se puede expresar a través de la función:

$$y_i(t) = f(\sum_{j=1}^{n} w_{ij}x_j - \theta_i), \forall\, i, 1 \leq i \leq n \tag{2.1}$$

Las neuronas de entrada de este modelo no realizan ningún tipo de procesamiento, solo se dedican a enviar información (consideraremos en este momento solamente señales discretas 0, 1) a las neuronas de salida. Estas últimas utilizan como función de activación la función escalón o **Función de Heaviside**. De este modo, la función 2.1 se puede escribir de la siguiente forma:

$$y_i = H(\sum_{j=1}^{n} w_{ij}x_j - \theta_i), \forall\, i, 1 \leq i \leq n \tag{2.2}$$

La forma básica de uso del perceptrón es como clasificador, como para realizar representaciones de funciones booleanas, esto último debido a que su salida es de carácter discreto.

A continuación mostraremos un ejemplo de uso del perceptrón como clasificador. En este ejemplo cada neurona del perceptrón representa una clase, al que dado un vector de entrada, cierta neurona va a responder con una salida $0, 1$, donde 0 dice si el valor no corresponde con el valor que representa y 1 si sí corresponde. Es fácil observar que el perceptrón solo va a permitir clasificar cuando las clases son linealmente separables, esto es, existe una única condición de separación lineal o hiperplano.

Sea $n_i$ una neurona de nuestro ejemplo que sólo va a tener dos entradas $x_1$ y $x_2$, con salida $y_i$ y cuya operación vamos a definir como:

$$y_i = H(w_1x_1 + w_2x_2 - \theta) \tag{2.3}$$

O desmenuzada de la siguiente forma:

$$y_i = \begin{cases} 1, si\ w_1x_1 + w_2x_2 \geq \theta \\ 0, si\ w_1x_1 + w_2x_2 \leq \theta \end{cases} \tag{2.4}$$

Si consideramos $x_1$ y $x_2$ situadas sobre el ejes de las abscisas y ordenadas en el plano, la condición

$$w_1x_1 + w_2x_2 - \theta = 0 \implies x_2 = -\frac{w_1}{w_2}x_1 + \frac{\theta}{w_2} \tag{2.5}$$

representa una recta (hiperplano si trabajamos con n entradas), la cuál va a dividir el plano en dos regiones, donde la neurona proporciona una salida 0 o 1 respectivamente[MSM02]

De esto se infiere que una neurona de tipo perceptrón representa un discriminador lineal, que al implementar una condición lineal que separa dos regiones en el espacio que representan dos tipos diferentes de patrones.

### Algoritmo de aprendizaje del perceptrón

La importancia del perceptrón radica en su capacidad como dispositivo entrenable. El algoritmo de aprendizaje propuesto por Rosenblatt, permite que automáticamente los pesos sinápticos se ajusten en proporción a la diferencia existente entre la salida actual de la red y la salida deseada, con el objetivo de minimizar el error de procesamiento de la red.

**Definición 2.1.** ***Regla de aprendizaje de Hebb***
*Se denomina* ***Aprendizaje Hebbiano*** *a aquellas formas de aprendizaje que involucran una modificación en los pesos $\Delta w_{ij}$ proporcional al producto de una entrada j por la salida i de la neurona:*

$$\Delta w_{ij} = \varepsilon y_i x_j \tag{2.6}$$

*siendo $\varepsilon$ un parámetro denominado* ***ritmo de aprendizaje****, qué suele ser un valor entre 0 y 1. De manera general podemos expresar la regla de Hebb de la siguiente forma:*

$$\Delta w_{ij}^{\mu} = t_i^{\mu} x_j^{\mu} \tag{2.7}$$

*Y así*

$$w_{ij}^{nuevo} = w_{ij}^{viejo} + \Delta w_{ij}^{\mu} \tag{2.8}$$

### Teorema de convergencia del perceptrón

**Teorema 2.1** (Teorema de convergencia del perceptrón)**.** *Supongamos que existe algún vector de parámetros $\theta^*$ tal qué $||\theta^*|| = 1$ y algún $\phi > 0$ tal que $\forall\, t = 1 \ldots n$,*

$$y_t(x_t, \theta^*) \geq \phi$$

*Además en adición, suponga que $\forall\, t = 1 \ldots n, ||x_t|| \leq R$*
*Entonces, el algoritmo del perceptrón tiene al menos*

$$\frac{R^2}{\phi^2}$$

*errores. La definición de error la vamos a entender como sigue: un error ocurre cuando tenemos $y' \neq y_t$ para algún par (j, t) en el algoritmo.*

Notemos que para cualquier vector $x$, nosotros usamos $||x||$ para referirnos a la norma Euclidiana de **x**, es decir, $||x|| = \sqrt{\sum_i x_i^2}$

*Demostración.* Denotemos como $\theta^k$ para ser el vector de parámetros cuando el algoritmo está en *k-ésimo* error. Note que tenemos

$$\theta^1 = 0$$

Ahora asumamos que el error *k-ésimo* se genera cuando vemos el ejemplar $t$, así que tenemos:

$$\begin{aligned} \theta^{k+1}\theta^* &= (\theta^k + y_t x_t)\theta^* && (2.9)\\ &= \theta^k\theta^* + y_t x_t \theta^* && (2.10)\\ &\geq \theta^k\theta^* + \phi && (2.11) \end{aligned}$$

De la ecuación 2.9 sigue la definición de las actualizaciones del perceptrón. La ecuación 2.11 sigue la suposición del teorema, donde tenemos:

$$y_t x_t \theta^* \geq \phi$$

Por inducción sobre $k$ tenemos qué:

$$\theta^{k+1}\theta^* \geq k\phi$$

Además, como $||\theta^{k+1}||x||\theta^*|| \geq \theta^{k+1}\theta^*$ y $||\theta^*|| = 1$, tenemos que:

$$||\theta^{k+1}|| \geq k\phi \qquad (2.12)$$

Ahora como segunda parte de la demostración, vamos a tomar la cota superior en $||\theta^{k+1}||$. Tenemos que:

$$\begin{aligned} ||\theta^{k+1}|| &= ||\theta^k + y_t x_t||^2 && (2.13)\\ &= ||\theta^k|| + y_t^2||x_t||^2 + 2y_t x_t \theta^k && (2.14)\\ &\leq ||\theta^k||^2 + R^2 && (2.15) \end{aligned}$$

La igualdad en 2.13 sigue la definición de actualización del perceptrón. La ecuación 2.15 parte de que tenemos: 1) $y_t^2||x_t||^2 = ||x_t||^2 \leq R^2$ por la suposición del teorema y porqué $y_t^2 = 1$; 2) $y_t x_t \theta^k \leq 0$ porqué sabemos que el vector de parámetros $\theta^k$ da error el t-ésimo ejemplo.
Siguiendo por inducción sobre $k$ (recordemos que $||theta^1||^2 = 0$) qué:

$$||\theta^{k+1}||^2 \leq kR^2 \tag{2.16}$$

Combinando las cotas de las ecuaciones 2.12 y 2.16 tehemos:

$$k^2\phi^2 \leq ||\theta^{k+1}||^2 \leq kR^2 \tag{2.17}$$

de donde tenemos

$$k \leq \frac{R^2}{\phi^2}$$

■

### 2.1.2 Adalina

### 2.1.3 Perceptrón multicapa

### 2.1.4 Back Propagation

## 2.2 Redes Neuronales no Supervisadas

### 2.2.1 Mapas autoorganizados

**Cuantificación optima de vectores**

**Modelo de neurona de Kohonen**

**Modelos de aprendizaje en mapas autoorganizados**

### 2.2.2 Redes de Hopfield

### 2.2.3 Funciones de base radial

### 2.2.4 Aprendizaje vectorial cuantificado

CAPÍTULO

# 3

# Clojure

3.1 **REPL**

3.2 **Evaluación**

3.3 **Estructuras de datos**

3.4 **Transducers***

3.5 **Multimétodos y jerarquías**

3.6 **Protocolos**

3.7 **Metadatos***

3.8 **Namespaces**

3.9 **Agents***

3.10 **Atoms***

3.11 **Interoperabilidad con Java**

3.12 **Spec**

CAPÍTULO

# 4

# Implementación

## 4.1 Primer caso de estudio: Perceptrón

### 4.1.1 Extendiendo el perceptrón simple: Perceptrón multicapa

### 4.1.2 Añadiendo una nueva regla de aprendizaje: Backpropagation

## 4.2 Segundo caso de estudio: Redes de Kohonen

### 4.2.1 implementando la cuántificación optima de vectores

### 4.2.2 Métricas y medidas de similitud

### 4.2.3 Extendiendo el modelo de aprendizaje: Batch SOM

CAPÍTULO

# 5

# Aplicaciones

## 5.1 Motivación e interés del empleo de las redes neuronales

## 5.2 Desarrollo de una aplicación: Reconocimiento de caracteres

### 5.2.1 Definición del problema

### 5.2.2 Modelación

### 5.2.3 Codificación

## 5.3 Comparación con otras técnicas

CAPÍTULO

# 6

# Conclusiones

## 6. CONCLUSIONES

# Bibliografía


[Beh03] Sven Behnke. *Hierarchical neural networks for image interpretation*, volume 2766. Springer Science & Business Media, 2003. 5

[Fuk80] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. 5

[GLM88] DANIEL Graupe, Ruey Wen Liu, and GEORGE S Moschytz. Applications of neural networks to medical signal processing. In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 343–347. IEEE, 1988. 5

[Hop82] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982. 5

[HW68] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968. 5

[Koh82] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 1(43):59–69, 1982. 1, 5

[Koh01] Teuvo Kohonen. *Self-organized maps*. Springer-Verlag, 2001. 1, 5

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 4

## BIBLIOGRAFÍA


[MSM02] BONIFACIO MARTIN and Alfredo Sanz Molina. *Redes neuronales y sistemas difusos*. 2002. 11

[RCE82] Douglas L Reilly, Leon N Cooper, and Charles Elbaum. A neural model for category learning. *Biological cybernetics*, 45(1):35–41, 1982. 5

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 1, 4

[Tur50] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. 4

[WH+60] Bernard Widrow, Marcian E Hoff, et al. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York, 1960. 4

[WH62] Bernard Widrow and Marcian E Hoff. Associative storage and retrieval of digital information in networks of adaptive "neurons". In *Biological Prototypes and Synthetic Systems*, pages 160–160. Springer, 1962. 4

# Glosario

**A | C**

**A**

**Aprendizaje Máquina**

Es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. 1

**Arquitectura de la Red**

Se entiende como arquitectura al diseño del entorno en que se conecta las neuronas entre sí dentro de la red. 2

**C**

**Clojure**

es un lenguaje de programación de propósito general dialecto de Lisp. 1, 5