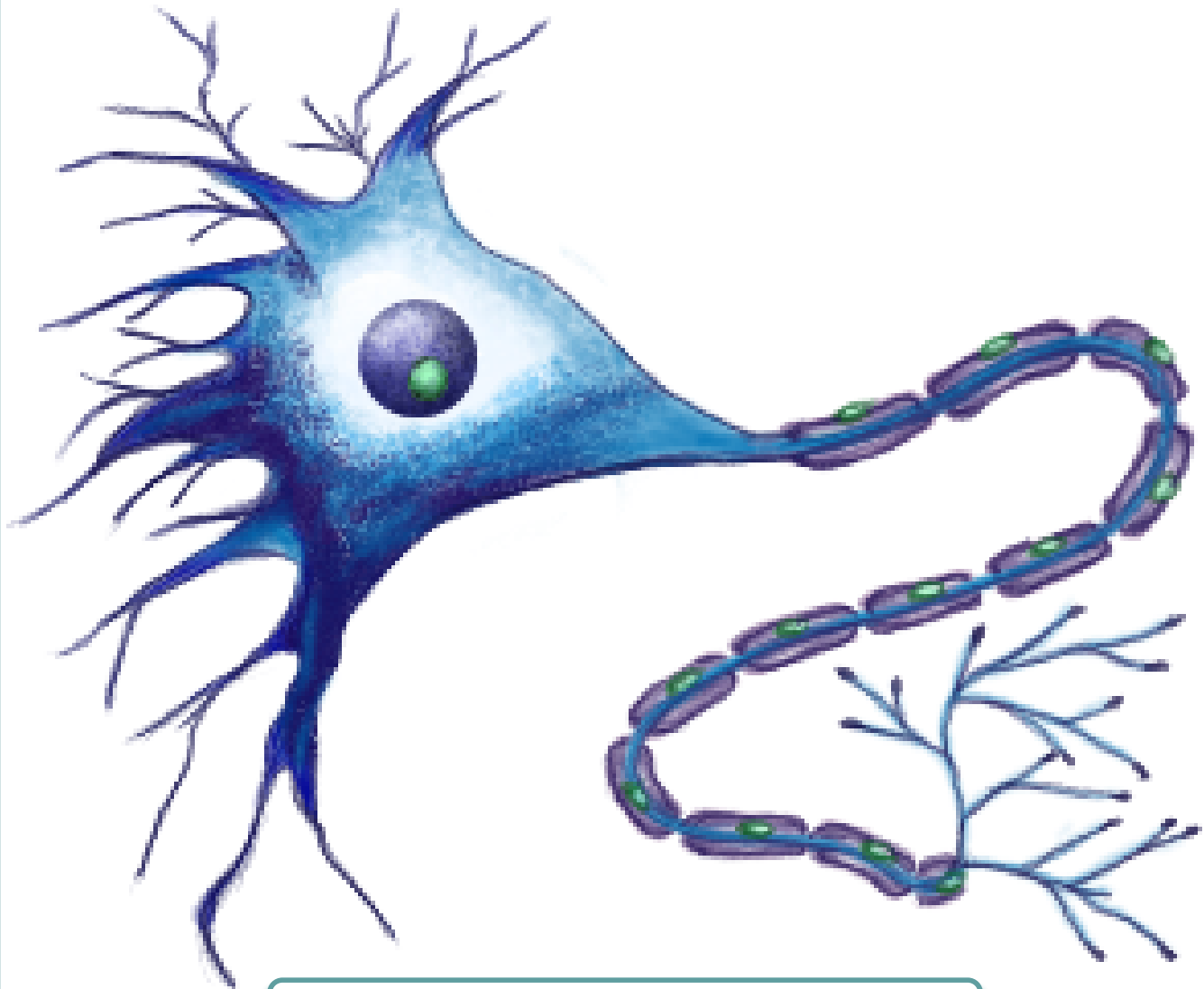


Redes Neuronales

Notas de clase

Karla Fernanda Jiménez Gutiérrez
Verónica Esther Arriola Ríos



FACULTAD DE CIENCIAS, UNAM

Índice general

Índice general	I
I Antecedentes	2
1 Neurona biológica	3
1.1 Neurociencias computacionales	3
1.2 Sistema Nervioso	4
1.2.1 Cerebro	5
1.2.2 Zonas funcionales	6
1.3 Neurona biológica	9
1.3.1 La neurona	9
1.3.2 Elementos de las neuronas en la transmisión de señales	9
1.3.3 Sinapsis	11
1.3.4 Señal eléctrica	12
2 Modelo de Hodgkin-Huxley	15
2.1 Introducción	15
3 Aprendizaje de máquina	16
3.1 Introducción	16
3.2 Espacio de Hipotesis	18
3.3 Clasificación de los conjuntos de datos	19
3.3.1 Tipos de aprendizaje	20
II Redes dirigidas acíclicas	22
4 Perceptrón simple	23
4.1 Perceptrón	23
4.2 Compuertas lógicas con neuronas	26
4.3 Funciones de activación	28
4.4 Funciones de error	28
4.5 Medidas de rendimiento	31

5	Perceptrón multicapa	37
5.1	Intro	37
5.2	XOR	37
5.3	Propagación hacia adelante manual	37
5.4	Propagación hacia adelante vectorizada (con matrices)	37
5.5	Interpretación matemática del mapeo no lineal	37
5.6	Propagación hacia adelante para el perceptrón multicapa	37
6	Entrenamiento por retropropagación	38
6.1	Esquema general entrenamiento	38
6.2	Función de error: Entropía cruzada	38
6.3	Derivada de la función logística	38
6.4	Entrenamiento en la última capa	38
6.5	Parcial con respecto a los pesos en la última capa	38
6.6	Vectorización	38
7	Optimización del entrenamiento	39
7.1	Problemas en redes profundas	39
7.2	Gradiente desvaneciente (o que explota)	39
7.3	Estilos de entrenamiento	39
7.4	Normalización y normalización por lotes	39
7.5	Regularización	39
8	Caso de análisis e interpretación	40
8.1	Red Hinton árbol familiar con numpy (entrenamiento)	40
8.2	Red Hinton árbol familiar con pytorch	40
9	Entrenamiento con genéticos	41
9.1	Algoritmos genéticos	41
9.2	Neuroevolución	41
9.2.1	Antecedentes: Aprendizaje por refuerzo en videojuegos	41
9.2.2	Arquitectura para estimar la función de recompensa	41
9.2.3	Entrenamiento	41
10	Mapeos autoorganizados	42
10.1	Introducción	42
10.2	Aprendizaje no supervisado	42
10.3	Mapeos autoo-organizados	42
10.4	Kohonen	42
11	Redes Neuronales Convolucionales	43
11.1	Convolución	43
11.2	Redes Convolucionales	43
11.3	Softmax	43
11.4	MNIST	43

III	Redes con ciclos	44
12	Redes Neuronales Recurrentes	45
12.1	Derivadas ordenadas	45
12.2	Retropropagación en el tiempo	45
12.3	Sistemas dinámicos y despliegue del grafo	45
12.4	Arquitectura recurrente universal	45
12.5	Función de error	45
12.6	Forzamiento del profesor	45
13	Atención	46
14	LSTM	47
15	GRU	48
16	Casos de análisis: etiquetado de palabras y conjugación de verbos	49
IV	Redes no dirigidas	50
17	Redes de hopfield	51
17.1	Entrenamiento	51
18	Máquinas de Boltzman	52
18.1	Entrenamiento	52
18.1.1	Partículas y partículas de fantasía	52
18.1.2	Máquinas de Boltzman Restringidas	52
19	Redes adversarias	53
19.1	GANs	53
A	Ecuaciones diferenciales	54
	Bibliografía	55

Etc

A lo largo del texto se utilizará la siguiente notación para diversos elementos:

Conjuntos	C
Vectores	x
Matrices	M
Unidades	cm

PARTE I

Antecedentes

1 | Neuronas biológicas

Neurociencias computacionales

El campo conocido como **neurociencias computacionales** (Trappenberg 2010) es el que se dedica explícitamente al estudio/modelado de los sistemas biológicos con ayuda de varios campos de estudio. Se interesa notablemente en descripciones y modelos funcionales biológicamente realistas de neuronas y sistemas neuronales.

Los campos de estudio de los cuales nos ayudamos son:

- **Ciencias cognitivas** dedicadas a tratar directamente con los humanos, de estas tomamos los conceptos de aprendizaje.
- **Biofísica** por el estudio de las propiedades físicas en de los sistemas biológicos.
- **Neurociencias tradicionales** con modelos matemáticos.
- **Ciencias de la computación** modelar e implementar los modelos dados, por las áreas aquí listadas. Simulaciones computacionales.
- **Ingeniería eléctrica** diseño de hardware específico y eficiente para tomar los pulsos y medidas exactas.

Las neurociencias computacionales, estudian modelos del sistema nervioso y clasifican estos modelos en tres tipos, recordemos el método científico:

1. **Modelos descriptivos**, Describen. Por ejemplo describe el comportamiento de los ratones a ciertas sustancias. La **comparación**, lo que estaba pasando antes de la sustancia, con la sustancia y después.
2. **Modelos mecanistas**, Mecánicamente. El **cómo** paso el evento o la acción. Siguiendo con el ejemplo. El ratón se cayó y se levantó, ¿Cómo? Doblo sus articulaciones, tomo impulso con sus músculos, roto su cuerpo, hasta reincorporarse totalmente.

1. Neuronas biológicas

3. **Modelos interpretativos**, Se interpreta. ¿**Por qué** hizo los movimientos mecánicos?. Siguiendo con el ejemplo. ¿Por qué el ratón se reincorpora? Porque el ratón quiere regresar a su estado anterior antes de la perturbación por la sustancia. ¿Será verdad? Es lo que se interpreta, no la verdad absoluta. Se tiene que buscar intencionalidad, razonamiento de más alto nivel.

Los **objetivos del modelado** en neurociencias de nuestro interés en el curso son; Las corrientes, las proteínas, las oscilaciones de las redes completas, la arquitectura topográfica y de columnas, el aprendizaje y la memoria. Por lo menos para el curso de Redes Neuronales en la Universidad Nacional Autónoma de México con la doctora Verónica Esther Arriola Ríos.

Para más información en Neurociencias (Kandel [2012](#)), siempre está el interés propio.

Las redes neuronales artificiales (RNA), están inspiradas en las redes neuronales biológicas, tales como las de los animales (Francisco López-Muñoz [2006](#)), en un primer momento. Hasta llegar al ser humano con el sistema nervioso y nuestras neuronas. (Waldeyer-Hartz [1891](#)).

Los modelos de redes neuronales que tomaremos, son simples. Distan mucho de los sistemas naturales y aun así dan solución. Lo que nos interesa es que resuelvan los problemas inmediatos y de corto plazo. Si el cerebro humano funciona, la imitación debe darnos algunos resultados, idea que ha funcionado para procesar información y dar solución con modelos diseñados.

Los problemas más notorios a resolver son:

- Problemas de visión por computadora.
- Procesamiento del lenguaje natural.

Sistema Nervioso

El sistema nervioso se estudia principalmente en dos partes, sistema nervioso periférico (SNP) y sistema nervioso central (SNC). El SNC es donde nos enfocaremos, este se compone principalmente por la médula espinal y el encéfalo que son los centros principales donde ocurre la correlación e integración de la información nerviosa. Es en el encéfalo donde se encuentra principalmente el cerebro, cerebelo y el tallo encefálico. (Snell [2001](#))

Un nervio es una estructura conductora de impulsos nerviosos. Está compuesto por una colección de axones los cuales son prolongaciones largas y delgadas de las células nerviosas agrupados en fibras y rodeados de vasos sanguíneos. Se origina desde la médula espinal o el encéfalo.

Los nervios en conjunto son la estructura organizada por tejido conectivo en el sistema nervioso, encargada de transmitir información entre diferentes partes del cuerpo del SNP y del SNC, permitiendo así el control y coordinación eficientes de las actividades corporales. Se extienden desde el cráneo y la médula espinal para abarcar todo el cuerpo humano. (Kandel 2012)

Los nervios del SNP se pueden clasificar en tres categorías:

- **Nervios motores:** están asociados con las salidas y la ejecución de acciones, conectándose y ejerciendo su influencia sobre los músculos.
- **Nervios sensitivos:** estos reciben señales de entrada, como en ojos, oídos y piel. Son responsables de transmitir información sensorial hacia el sistema nervioso.
- **Nervios mixtos:** estos tienen tanto fibras sensitivas como motoras. Es decir, cumplen funciones tanto de entrada como de salida de información.

El SNC está compuesto por gran cantidad de células nerviosas excitables y sus prolongaciones, denominadas neuronas. (Snell 2001)

Cerebro

Desde la parte tangible del cerebro hasta la intangible del pensamiento, está directamente relacionada con la forma que funciona nuestro pensamiento y reacciones motoras.

El cerebro, por su gran tamaño, da lugar a un mejor registro de su actividad. Los muchos registros realizados muestran que se incrementa la circulación de la sangre en el área del cerebro con mayor actividad neuronal. Se tienen identificadas regiones que se activan ante cierto estímulo. (John G. Nicholls 2011)

Se detecta cuánta sangre se está bombeando en diferentes regiones del cerebro dependiendo de los estímulos que se le presentan a una persona; o si alguna persona tiene un padecimiento, se toman escaneos para ver qué regiones del cerebro están funcionando y cuáles presentan lesiones. A partir de las lesiones y de la identificación de la actividad que ya no se puede realizar de forma normal, se infiere qué región era responsable de esa actividad que ahora está dañada. (Doidge 2007)

En la figura Figura 1.1 se puede observar las regiones del cerebro que han sido identificadas y sus respectivas funciones asociadas.

Para obtener más detalle de los estudios de la actividad del cerebro, ver el documental "The Brain with David Eagleman".

1. Neurona biológica

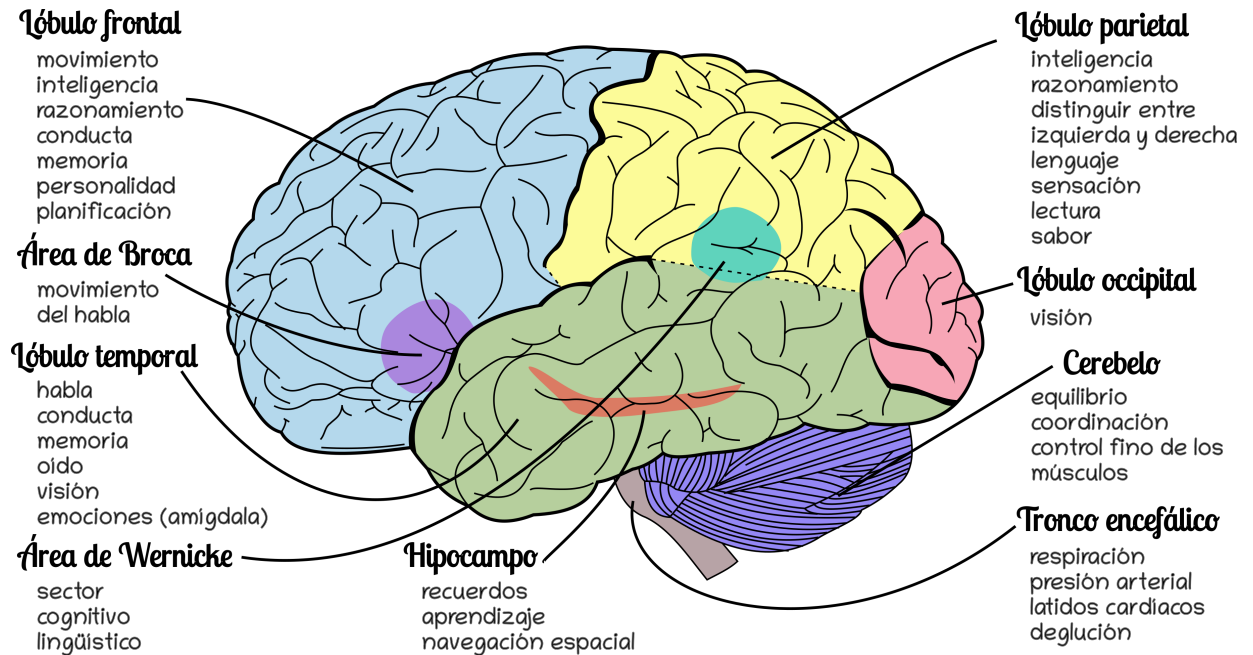


Figura 1.1 Representación de las regiones del cerebro. Adaptado de https://commons.wikimedia.org/wiki/File:Brain_diagram_pl.svg. Redibujado por Verónica E. Arriola Ríos, CC-BY-SA 3.0

Zonas funcionales

En el cerebro, las diversas zonas identificadas con funciones específicas, están conectadas, por una ruta que se conoce como "la ruta desde la sensación hasta la cognición". Esta ruta está representada por el diagrama de la figura Figura 1.2. (M Marsel Mesulam 1998a)

La ruta es una abstracción que registra la manera en que una señal recorre las zonas funcionales del cerebro. La Figura 1.2, describe un gradiente sensorial que transmite información desde las áreas sensoriales primarias hasta las áreas límbicas del cerebro.

El punto de partida de la ruta es la capa sensorial, que vamos a considerar como **la entrada**. El cerebro está recibiendo información en el momento que existe una señal que viene del mundo exterior.

Su primera conexión es hacia la capa unimodal superior, que vamos a considerar como **la primera capa**, donde se procesa la información más básica de la señal (del sentido), tales como el color, las formas o los tonos de sonido.

Luego están las capas intermedias unimodal inferior, heteromodal, paralímbico, límbico, donde se dará un procesamiento intermedio; se juntarán varias características interpretadas por la primera capa. Conforme avance la señal a las siguientes capas, las neuronas notarán características más complejas.

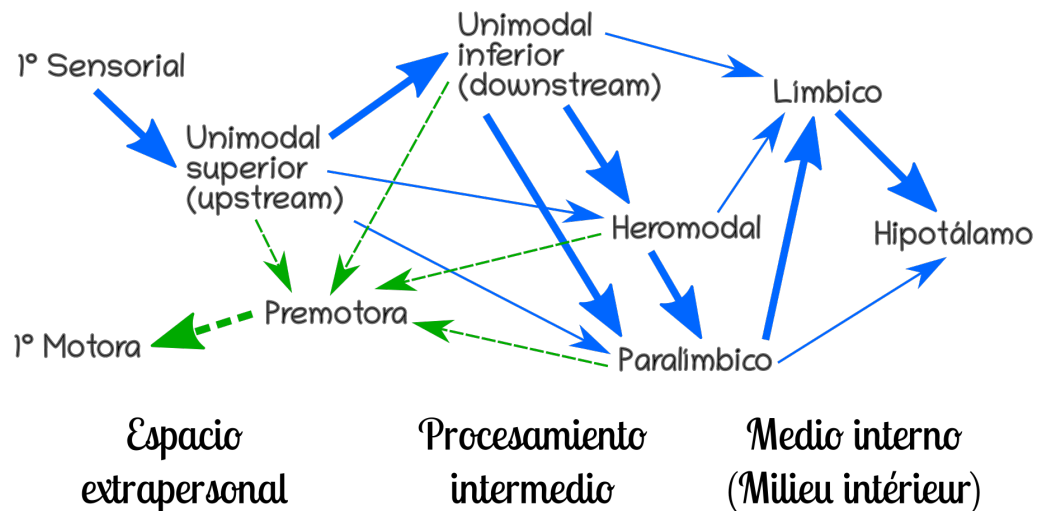


Figura 1.2 Diagrama de la arquitectura del cerebro a nivel funcional. Enfocado en sentido de la visión o la audición. (M. Marsel Mesulam 1998b). Redibujado por Verónica E. Arriola Ríos.

Una vez que llegue la señal con la codificación de todas las capas por las que paso al hipotálamo, este se encarga de coordinar las respuestas de regreso. Ya sea hacia la capa premotora o paralímbica. En el caso del diagrama las flechas en color verde indican que la señal tuvo respuesta antes de llegar al hipotálamo y hubo una respuesta hacia la capa premotora, que tendrá una salida hacia los nervios motores, es decir una reacción motora inmediata de la señal. Las flechas de color azul indican que la señal llegó hasta el hipotálamo que pudo coordinar una o varias respuestas.

En este contexto, las áreas unimodales y heteromodales están más involucradas en la elaboración perceptual y la planificación motora, mientras que las áreas paralímbicas desempeñan un papel crítico en canalizar la emoción y la motivación hacia objetivos intrapsíquicos y extrapersonales relevantes para el comportamiento. Este mapa funcional del córtex cerebral proporciona la plantilla básica para vincular la sensación con la cognición en el cerebro de primates.

Cabe destacar que, si bien no es imprescindible contar con un conocimiento profundo del cerebro para trabajar con Redes Neuronales Artificiales (RNA), poseer una comprensión básica del sistema nervioso y las zonas funcionales del cerebro es esencial para el desarrollo de una RNA por varias razones:

- Inspiración biológica:
 - ★ Nuestro sistema nervioso ha evolucionado para procesar información de manera eficiente, comprender sus componentes proporciona ideas valiosas para diseñar algoritmos y arquitectura de redes neuronales.
- Arquitectura y función:

1. Neurona biológica

- ★ Comprender someramente la arquitectura y funciones del sistema nervioso en nuestro cuerpo, nos ayuda en la creación de modelos de redes artificiales que pueden realizar tareas específicas, como reconocimiento de patrones, procesamiento de lenguaje natural, y más.
- Aprendizaje y Adaptación:
 - ★ El sistema nervioso es capaz de aprender y adaptarse a través de experiencias. Conocer los mecanismos de aprendizaje en el cerebro puede inspirar algoritmos de aprendizaje profundo que permitan a las RNA a mejorar su rendimiento con el tiempo.
- Optimización de Rendimiento:
 - ★ El sistema nervioso está altamente interconectado, permite la comunicación rápida y eficaz entre las células nerviosas. Esto nos guía en el diseño de conexiones eficientes en las RNA para optimizar el rendimiento del paso de información.
- Robustez y tolerancia a fallos:
 - ★ El sistema nervioso es robusto y capaz de funcionar en condiciones adversas. La observación cuidadosa de esta capacidad, nos sugiere estrategias para mejorar la robustez y la tolerancia a fallos en las RNA.
- Ética y Seguridad:
 - ★ Al notar la gran complejidad de nuestro sistema nervioso, notamos también nuestros límites de imitación y semejanza que realmente se puede lograr. La responsabilidad de crear RNA conlleva sus posibles riesgos y dilemas/desafíos éticos.

Esta atención especial a las zonas funcionales del cerebro se hace por la necesidad de entender los conceptos fundamentales de entradas, capas, salidas y conexiones, los cuales son la base para comprender conceptos más avanzados como memoria y reacciones, tanto inmediatas como a largo plazo. Estos últimos conceptos son especialmente relevantes en modelos específicos, como las RNA de memoria a corto y largo plazo (LSTM, por sus siglas en inglés Long Short Time Memory) y las RNA con Unidad Recurrente con Puertas (GRU, por sus siglas en inglés Gate Recurrent Unit) que se estudiarán detenidamente en etapas posteriores del curso.

Neurona biológica

La neurona

La neurona es un tipo de célula perteneciente al sistema nervioso central, que se comunica tanto por señales eléctricas como por señales químicas. Cada neurona tiene (Snell 2009):

- Un cuerpo celular llamado **soma** que contiene un núcleo y otros componentes celulares.
- Una zona de recepción con protuberancias elongadas llamadas **dendritas**.
- Una zona de emisión llamado **axón**, el cual está compuesto de:
 - ★ Cono axónico.
 - ★ Membrana plasmática axónica y citoplasma.
 - ★ Recubrimientos de mielina, interrumpido a intervalos regulares por nodos de Ranvier.
 - ★ Terminales del axón donde se encuentran los botones sinápticos.

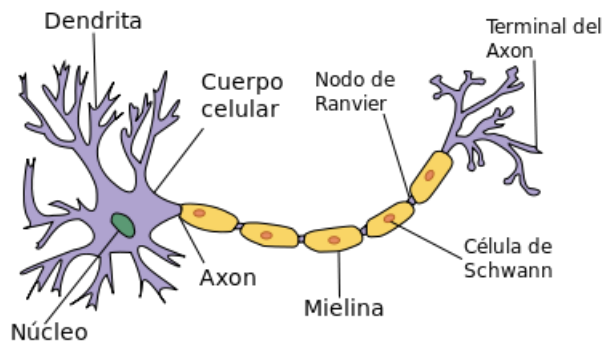


Figura 1.3 Neurona, Acracia, 14 January 2007, Wikimedia Commons, <https://commons.wikimedia.org/wiki/File:Neurona.svg>, Creative Commons Attribution-ShareAlike 2.5 Generic

Elementos de las neuronas en la transmisión de señales

Los principales elementos que participan durante la transmisión de señales entre neuronas son:

1. Neurona biológica

- **Impulsos eléctricos:** potenciales de acción, cambios de voltaje que ocurren a lo largo del axón. Pueden generar dos efectos en la membrana de la neurona:
 - ★ **El efecto excitatorio**, despolariza la membrana postsináptica. La neurona es más propensa a mandar un pulso eléctrico.
 - ★ **El efecto inhibitorio**, hiperpolariza la membrana postsináptica. La neurona no manda pulso eléctrico.
- **Neurotransmisor/es:** donde se encuentra la sustancia química, son los mensajeros químicos que se comunican entre neuronas adyacentes. La liberación de neurotransmisores de una neurona ayudará a despolarizar o hiperpolarizar (aumentar la magnitud de la carga) de la neurona adyacente, lo que hará que sea más o menos probable que ocurra un potencial de acción en la siguiente neurona.
- **Plasticidad:** La modificación a largo plazo de las conexiones entre neuronas.

En el contexto de la transmisión neuronal de señales, es necesario realizar ciertas distinciones para denotar claramente si una neurona particular está recibiendo o transmitiendo una señal. Las distinciones son las siguientes:

- **Neurona presináptica:** Transmite una señal a otras neuronas o células a lo largo del sistema nervioso.
- **Neurona postsináptica:** Recibe una señal de otras neuronas o células, procesando y respondiendo a dicha señal.

La transmisión de señales y almacenamiento de información en las neuronas se da de la siguiente forma:(John G. Nicholls [2011](#))

1. La neurona desde sus dendritas recibe señales de otras neuronas vecinas.
2. Cada señal se va acumulando en su cuerpo hasta el cono axónico, donde se van a estar sumando la contribución de todos los efectos de cambios de potencial.
3. En el momento que se rebase un cierto valor umbral, la diferencia de potencial se propaga hasta los botones terminales.
4. La neurona entra en un período refractario, donde empieza a cambiar el potencial entre el cono axónico y el axón de la neurona.
5. Se transmite un disparo eléctrico en seguida.
6. La neurona se va a quedar totalmente quieta, durante un breve momento para que la señal pueda viajar hacia el axón.

7. Se va a notar un cambio muy violento en el voltaje, que se va recorriendo a lo largo de todo el axón.

A lo anterior también se le conoce como neurotransmisión que esta descrita con mayor precisión en la figura Figura 1.4.

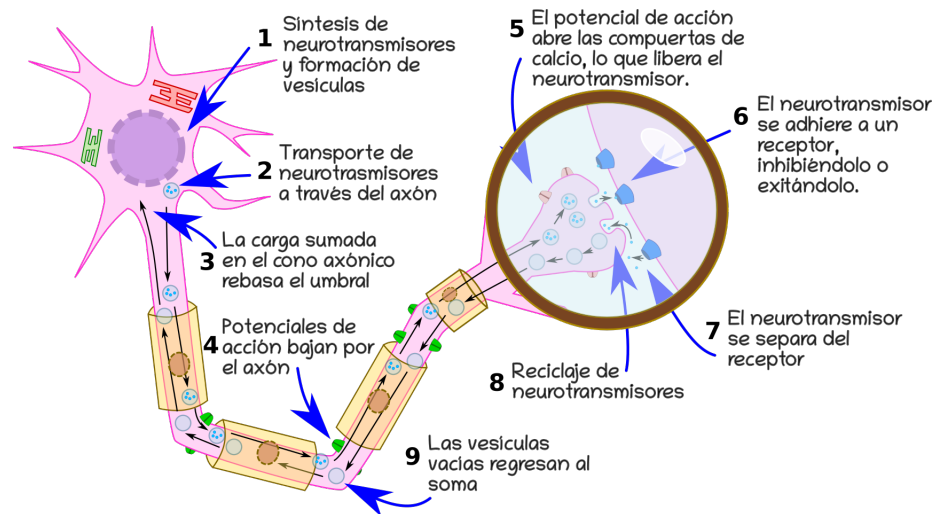


Figura 1.4 Esquema detallado de una neurotransmisión. Autor: Verónica Arriola Rios.

La neurona (típica) a lo largo de su axón, está cubierta de nodos (y de células de mielina). Estos nodos están para evitar que se distorsionen o se pierda la señal recibida en la dendritas, en ellos se recarga la señal para que llegue hasta el soma de la neurona. Donde se mandará o no una respuesta, un pulso eléctrico.

Sinapsis

Donde dos neuronas en estrecha proximidad, se transmiten información se llama **sinapsis**. La sinapsis (típica) se establece entre el axón de una neurona y la dendrita de otra neurona. (Snell 2009)

Clasificación de sinapsis

Los dos tipos de sinapsis (relevantes para el curso) son:

Sinapsis eléctrica: las membranas de las células pre y post sinápticas se unen en la brecha sináptica, que son pequeños canales que permiten el paso de iones (Figura 1.6).

Sinapsis química: la neurona libera moléculas neurotransmisoras a otra neurona adyacente en un pequeño espacio, la brecha sináptica (Figura 1.7).

1. Neurona biológica

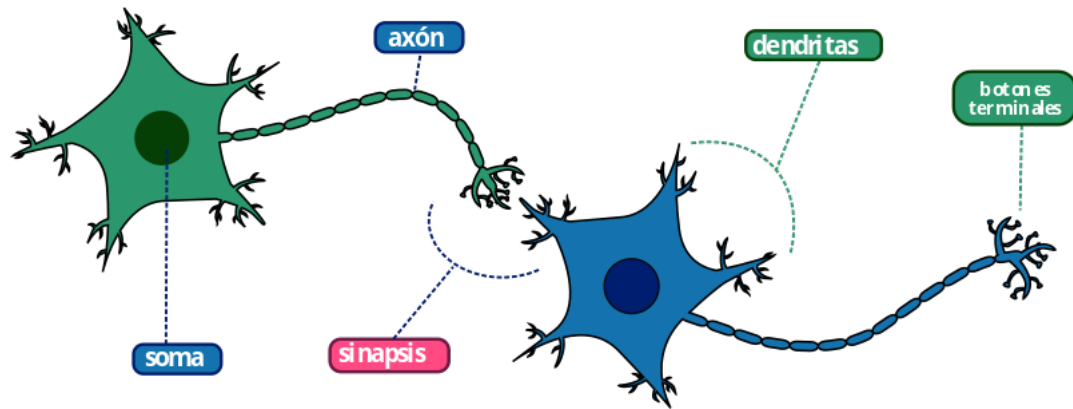


Figura 1.5 Part of neurons in Spanish, Dana Scarinci Zabaleta, 24 February 2019, Wikimedia Commons, https://commons.wikimedia.org/wiki/File:Part_of_neurons_in_Spanish.svg, OpenStax, CC0

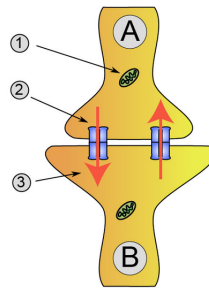


Figura 1.6 Synaptical transmission (electrical). Neurona A transmisora, Neurona B receptora, 1. Mitocondria, 2. Uniones gap formadas por conexiones, 3. Señal eléctrica, Nrets commonswiki, 23 September 2005, Wikimedia Commons, https://commons.wikimedia.org/wiki/File:Synapse_diag2.png, Inkscape 0.42, CC-BY-SA 3.0

Señal eléctrica

El paso de las señales eléctricas entre neuronas, se dan gracias a los canales de iones, los que destacan son los siguientes:

- **Canal por fuga:** Estos se abren y cierran aleatoriamente, todo el tiempo están activos en la neurona. Por ejemplo: sodio y potasio.
- **Canal regulado por ligado:** En este un neurotransmisor que es el que provocar o impide que se abran.
- **Canal por estímulo mecánico:** Permiten que pasen más iones o menos iones dependiendo, si se ejerció una presión, por ejemplo, con las neuronas cerca de la piel.

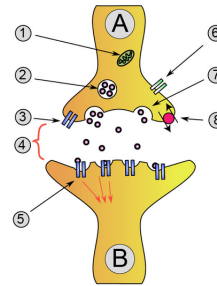


Figura 1.7 Synaptical transmission (chemical). Neurona A transmisora, Neurona B receptora, 1. Mitocondria, 2. Vesícula sináptica llena de neurotransmisor, 3. Autorreceptor, 4. Brecha sináptica, 5. Receptor de neurotransmisores, 6. Canal de calcio, 7. Neurotransmisor liberador de vesículas fusionadas, 8. Bomba de recaptación de neurotransmisores, Utilisateur:Dake, 23 September 2005, Wikimedia Commons, https://commons.wikimedia.org/wiki/File:Synapse_diag1.png, Inkscape 0.42, CC-BY-SA 3.0

- **Canales regulados por el voltaje:** Un voltaje, es el que abre o cierra el canal. Permite o no el paso del pulso eléctrico.

Existen realmente una buena cantidad de iones presentes en el cerebro, pero los más protagonistas son **potasio**, el **sodio** y el **cloro**. Los que vamos a utilizar para un modelo matemático de las neuronas.

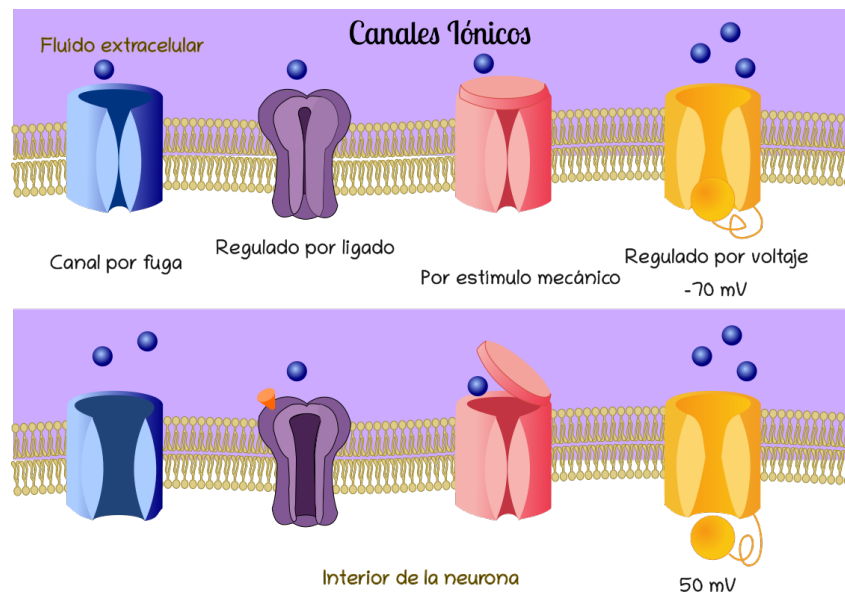


Figura 1.8 Representación de la clasificación de los canales iónicos.

Neuroplasticidad

Una característica importante de la conexión entre neuronas es la neuroplasticidad, que es lo que nos permite el aprendizaje a largo plazo en el cerebro, es un mecanismo de aprendizaje del cerebro en el cual cuando las neuronas se activan simultáneamente con frecuencia la conexión entre ellas se fortalece.

Este mecanismo constituye la principal inspiración para el diseño de las redes neuronales artificiales, concretamente en esto se inspiran los algoritmos de entrenamiento. Calculamos, qué conexiones debemos reforzar y cuáles debemos de debilitar para que nuestras redes neuronales calculen las funciones que a nosotros nos interesan.

Comentario Importante a preguntar: En esta sección estaba también explicado el paso de iones por la membrana, pero no sé si sea relevante en esta sección o mejor directamente en el modelo de HH, que es lo importante para sus prácticas de lab.

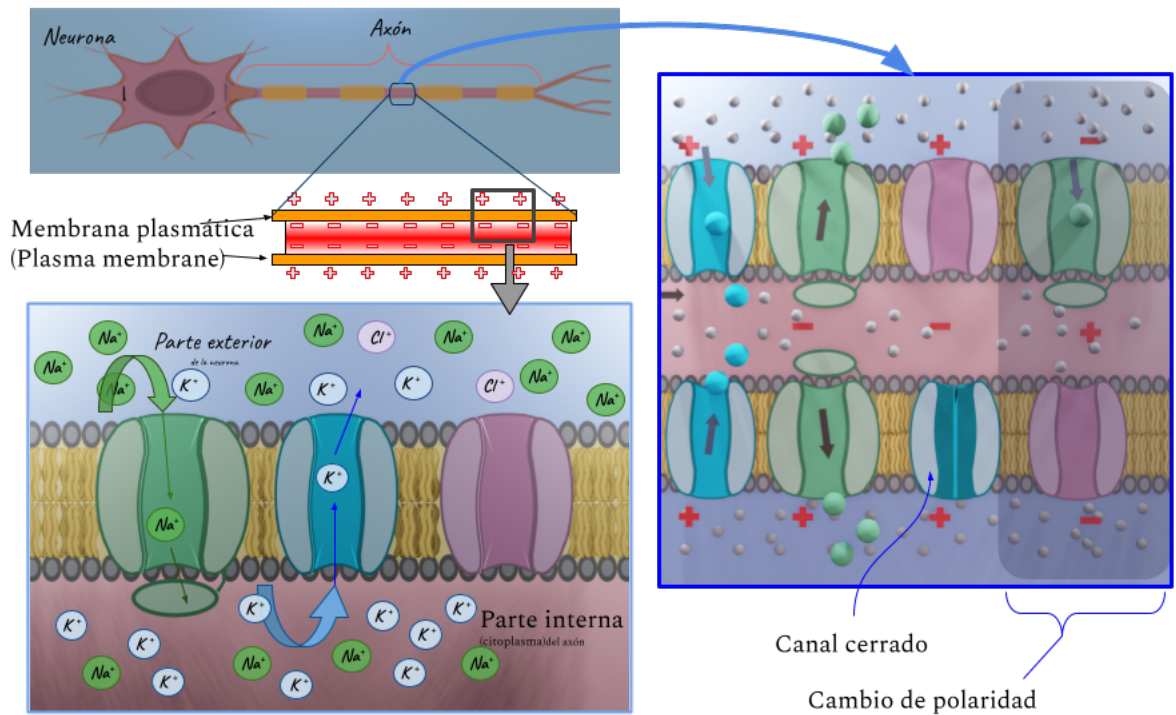


Figura 1.9 Representación de la membrana axónica en potencial de reposo en la parte inferior izquierda, y en la parte derecha con un estímulo que genera el cambio de polaridad en la misma, así como el cierre de canales y paso de iones.

2 | Modelo de Hodgkin-Huxley

Introducción

El modelo de HH.

3 | Aprendizaje de máquina

Introducción

En este capítulo se desarrolla el procesos que pasa una maquina para que *aprenda*, para esto notemos el concepto de aprender. En lo seres humanos se denota como el hecho de adquirir el conocimiento de algo mediante el estudio o la experiencia a partir de ejemplos específicos en nuestro medio, entonces aquellos problemas que inicialmente no pueden resolverse, puedan ser resueltos después de obtener más información acerca del problema. Desde pequeños empezamos por aprender por palabras (conceptos) que asociamos con algo específico, para después relacionar que varios objetos pertenecen a un tipo de conjunto y otros no. Como que un muñeco, una pelota y unos bloques de construcción de plástico, pertenecen a un conjunto denotado como "juguetes", y que plato, taza, y tazón no pertenecen a este conjunto sino al conjunto "vajilla". Entonces para organizar los conceptos que vamos aprendiendo hacemos uso de una función booleana, donde la entrada es el concepto, la pregunta es, ¿Este objeto pertenece a un cierto conjunto de objetos con características similares? y la salida es falso o verdadero. A este proceso, se le conoce como *aprendizaje de conceptos* y en este curso lo simplificamos como *función booleana de aproximación mediante ejemplos*.

Ahora lo que denotamos como el hecho que una maquina aprenda, lo vemos como cualquier programa que mejore su desempeño en alguna tarea mediante la experiencia. Con más formalidad se denota como:

Definición 3.1

Aprendizaje maquina: Se dice que un programa de computadora aprende, si su desempeño en T , medido por P , mejora con la experiencia E . Tal que:

- T es un tipo de tarea.
- P es una medida de desempeño.

- E la experiencia con ejemplos (entrenamiento).

, ^a.

^aMachine Learning, Mitchell 1997, pag. 14.

Algunos ejemplos para definición anterior pueden ser los siguientes:

- T , como:
 - ★ Jugar un juego de mesa.
 - ★ Clasificar varios tipo de hojas.
 - ★ Reconocer una voz en particular.
- P , como:
 - ★ Porcentaje de juegos ganados en las partidas.
 - ★ Porcentaje de hojas correctamente clasificadas.
 - ★ Porcentaje de reconocimiento de timbre de la voz.
- E , como:
 - ★ Jugar juegos de práctica.
 - ★ Una secuencia de imagenes etiquetadas.
 - ★ Una secuencia de audios etiquetados.

A partir de ahora nos dedicaremos a definir correctamente tareas que nos interese que un programa aprenda, para entender la forma más abstracta del problema y así proponer los algoritmos que nos ayuden a resolverlo.

Consideremos también que los sistemas de redes neuronales artificiales son un tipo de algoritmo para la representación del proceso de aprendizaje. Un problema de aprendizaje bien definido requiere una tarea bien especificada, medidas de desempeño y datos para obtener experiencia.

El aprendizaje maquina se apoya de disciplinas, como la inteligencia artificial, probabilidad, estadística, complejidad computacional, psicología, neurobiología y filosofía.

Para proponer un algoritmos de aprendizaje automático necesitamos, elegir el tipo de experiencia de entrenamiento, definir la función objetivo a aprender y un algoritmo para aprender la función objeto a partir de ejemplos de entrenamiento.

Los algoritmos de aprendizaje maquina han sido utilizados ampliamente por la industria bancaria, por gobiernos y por su puesto por el area de la salud. En la industria bancaria por

ejemplo, donde es necesario aprender las reglas generales para determinar la solvencia crediticia, a partir de las bases de datos. Por los gobiernos para el reconocimiento de rostros humanos a partir de imágenes. En el área de salud para a partir de bases de datos de pacientes descubrir automáticamente regularidades implícitas en los resultados de tratamientos.

Espacio de Hipotesis

El aprendizaje automático, es utilizar datos disponibles para, aprender una tarea mediante una función que mejor mapee entradas a ciertas salidas. A esto se le llama aproximación de función, en el que aproximamos una función de destino desconocida (que suponemos que existe) que puede asignar mejor las entradas a las salidas en todas las observaciones posibles del dominio del problema.

Una función de un modelo que se aproxima a la función objetivo y realiza asignaciones de entradas a salidas se denomina hipótesis.

Ahora estas funciones pueden tener formas muy generales en el aprendizaje de máquina pueden tener forma, por ejemplo, de estructuras de datos, como los árboles de decisión, donde cada nodo pregunta si o no, pertenece una clasificación. pueden ser también funciones matemáticas como el caso de las redes neuronales, entonces la forma que tomen estas hipótesis en general puede abarcar muchos métodos y estructuras.

Entonces el aprendizaje consiste en, explorar un espacio de posibles hipótesis para encontrar la hipótesis (una función) que mejor encaje, de acuerdo a lo se obtuvo en los ejemplos de entrenamiento, y predecir alguna característica de salida deseada. Usualmente se denotan como sigue:

- h (hipótesis): una sola hipótesis, por ejemplo una instancia o modelo candidato específico que asigna entradas a salidas, se puede evaluar y se usa para hacer predicciones.
- H (conjunto de hipótesis): Un espacio de posibles hipótesis para mapear entradas.

Una breve ejemplo para denotar un espacio de hipótesis sería el problema es saber los días que nos conviene ir al cine, donde nuestra tarea T es aprender a predecir el conjunto de días que nos conviene ir al cine, basado en los atributos de los días, donde cada hipótesis la representaremos a partir de un conjunto de atributos de las instancias (días), entonces cada hipótesis es un vector con tres atributos, *tiene2x1*, *esEstrenoDePelicula*, *actoresConocidos*. Para cada atributo de la hipótesis tendría uno de los siguientes valores; Si, No, ?. Donde ? indica que cualquier valor es válido para ese atributo.

Cuando alguna instancia x cumpla con todos los atributos de una h , entonces $h(x) = 1$ y x es un ejemplo positivo. Entonces para representar la hipótesis, que nos conviene ir

solo los días con 2x1, y que hay películas donde los actores son conocidos, la escribimos como $h(<S_i, ?, S_i>) = 1$, la hipótesis que cualquier día nos conviene ir al cine la denotamos como $h(<?, ?, ?>) = 1$, nuestra función objetivo la denotamos como una función booleana $c : X \rightarrow \{0, 1\}$, el conjunto de los 365 días del año, entonces $c(x) = 1$ cuando en los datos nos dicen que con la instancia x conviene ir al cine, $c(x) = 0$ en caso que no. Por tanto para aprender la tarea T , necesitamos *una hipótesis h en H tal que $h(x) = c(x)$ para todas las x en X* . La tarea de aprendizaje del concepto c requiere aprender el conjunto de instancias que lo satisface, describiendo este conjunto mediante una conjunción de restricciones sobre los atributos de la instancia.

Estas hipótesis (funciones) pueden llegar a ser sumamente complejas y tener que mapear datos de entrada con muchas formas ej. imágenes, trayectorias, etc. En el caso de las redes neuronales, el espacio de hipótesis está determinado por la arquitectura de la red. Vamos a definir el espacio de hipótesis, cuando decidamos qué neuronas vamos a poner en nuestro sistema, como las conectamos entre sí y cómo van a transferirse información de una a la otra y cuántas neuronas van a ser. Lo que veremos a lo largo del curso son diferentes arquitecturas y el impacto que tiene hacer diferentes modificaciones así como las matemáticas que existen detrás de estas.

Clasificación de los conjuntos de datos

La experiencia E para aprender la vamos a obtener mediante un conjunto de datos, llamados datos de entrenamiento, estos se separan en tres bloques:

- **Entrenamiento:** Datos con los cuales se ajustan los parámetros de la hipótesis (del 50 % al 80 % de los datos). En este bloque se elige que función del espacio fue mejor para el aprendizaje.
- **Validación:** Datos utilizados para ajustar los parámetros (hiperparámetros) del algoritmo de entrenamiento, que puedan afectar qué hipótesis es seleccionada (del 25 % al 10 % de los datos y no deben ser usados durante el entrenamiento). Un ejemplo de un hiperparámetro para redes neuronales son el número de nodos ocultos en cada capa.
- **Prueba:** Datos utilizados para evaluar la posibilidad de que la hipótesis aprendida generalice ¹ a datos no vistos anteriormente. Esta porción que se mantiene aparte. Con estos se evalúa el modelo, se reporta la eficacia del modelo según los resultados en este conjunto (del 25 % al 10 % de los datos).

¹Se desea que nuestro modelo de aprendizaje, una vez entrenado con datos que ya hemos visto, se pueda usar con datos nuevos. Para ello debemos asegurarnos que el modelo no ha simplemente memorizado las muestras de entrenamiento, sino que ha aprendido propiedades del conjunto.

El conjunto de datos de entrenamiento se usa para aprender una hipótesis y el conjunto de datos de prueba para evaluarla.

Tipos de aprendizaje

Aprendizaje Supervisado , el modelo usa datos etiquetados a una respuesta específica (labeled data), durante el entrenamiento se intenta encontrar una función que aprenda a asignar los datos de entrada (input data) con los datos en el etiquetado. Para después predecir una relación, dado un dato totalmente nuevo para el modelo. Los modelos pueden ser:

- Regresión: Un modelo de regresión busca predecir valores de salida continuos. Por ejemplo, en predicciones meteorológicas, de expectativa de vida, de crecimiento de población.
- Clasificación: En un problema de clasificación se desea predecir una salida discreta. Por ejemplo, identificación de dígitos, diagnósticos.

Aprendizaje no supervisado , es usado cuando no se tienen datos “etiquetados” para el entrenamiento. Solo sabemos los datos de entrada. Por tanto, únicamente podemos describir la estructura de los datos, para intentar encontrar algún tipo de organización que simplifique un análisis. Por ello, no se tienen valores correctos o incorrectos (es utilizado para aprender de una manera autoorganizada).

Aprendizaje por refuerzo , inspirado en la psicología conductista; donde el modelo aprende por sí solo el comportamiento a seguir basándose en *recompensas y penalizaciones*. Este tipo de aprendizaje se basa en mejorar la respuesta del modelo usando un proceso de retroalimentación (*feedback*). Su información de entrada es el feedback que obtiene del mundo exterior como respuesta a sus acciones. Aprende a base de ensayo-error.

Mientras que el aprendizaje supervisado y el no supervisado aprenden a partir de datos obtenidos en el pasado, el aprendizaje por refuerzo aprende desde cero, es decir, con un estado inicial y son su ambiente, va aprendiendo a futuro, mediante posibles penalizaciones o recompensas. El *aprendizaje por refuerzo* es usado en videojuegos porque cada vez que se realizan las acciones correctas se ganan puntos y entonces se entrena a la gente para que pueda conseguir la mayor cantidad de puntos. En este siempre hay: un agente, un ambiente definido por estados, acciones que el agente lleva a cabo (que le llevan de un estado a otro), y recompensas o penalizaciones que el agente obtiene.

En cada acción, el agente solo conoce el estado en el cual se encuentra y las acciones posibles que puede elegir a partir de ese estado. No sabe si llegando al siguiente estado, obtendrá mejores o peores recompensas, irá aprendiendo en cada estado qué acciones lo llevarán a obtener una mayor recompensa a largo plazo, y que el valor de las acciones en ese estado puedan subir. *Se enfoca en que el agente aprenda una política óptima*

para alcanzar el objetivo. El agente siempre está en fases de *exploración* y *explotación*, en la fase de exploración el agente toma una acciones de manera aleatoria, y en la de explotación va a tomar acciones basándose en cuán valiosa es realizar una acción a partir de un estado dado.

En plataforma de ventas en línea es donde podemos encontrar este tipo de modelo que están entrenados con este tipo de aprendizaje, donde al iniciar la sesión no conoce nada del usuario, solamente tiene un ambiente dado por los productos de la plataforma y su estado inicial es cero, para hacer individual la experiencia del usuario y que compre más. El algoritmo realiza la acción de mostrar ciertos productos (algún estado) si el usuario da clic a estos productos, el agente recibirá un punto de recompensa, por lo cual pasará a otro estado donde ofrecerá productos del mismo estilo donde pueda maximizar una venta, así se ira adaptando a cada usuario.

PARTE II

Redes dirigidas acíclicas

4 | Perceptrón simple

Perceptrón

El perceptrón fue la primera red neuronal artificial (o ANS, Artificial Neural Systems) descrita algorítmicamente. En las décadas de los 60's y 70's, los popularizó el psicólogo Franck Rosenblatt, en su libro llamado Principios de neurodinámica, donde presentó varios modelos de perceptrones, en el Laboratorio Aeronáutico de Cornell en Estados Unidos, originalmente estaba diseñado para ser una máquina, en vez de un algoritmo. Estaba diseñado específicamente para el reconocimiento de imágenes donde, cada peso era un cable físico por pixel de entrada, este era una matriz de 200×200 , conectados aleatoriamente a las "neuronas", las actualizaciones de los pesos se realizaron mediante motores eléctricos.

El perceptrón es en sí, es la representación de una sola neurona, este se ocupa para la clasificación de patrones en un conjunto de datos multivariados, con ese se obtienen fronteras lineales en el plano, mediante un algoritmo de aprendizaje que veremos más adelante.

Recordando, una neurona es una célula elemental que a partir de un vector de entrada procedente del exterior o de otras neuronas (estímulo), proporciona una única respuesta (si activo el potencial de acción o no), ver figura 4.1. Los elementos que actúan en una neurona los podemos listar como:

- **Entradas:** $x_j(t)$. Las variables de entrada y salida pueden ser binarias (digitales) o continuas (analógicas) dependiendo del modelo de aplicación.
- **Pesos sinápticos:** w_{ij} . Representan la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i .
- **Regla de propagación:** $h_i(t) = \sigma(w_{ij}, x_j(t))$. Proporciona el valor del potencial postsináptico, de la neurona i en función de sus pesos y entradas.

★ $h_i(t) = \sum_{j=0}^n w_{ij} x_j$, Es una suma ponderada de las entradas con los pesos sinápticos. Así, si la entrada es positiva, dependiendo de los pesos podemos

saber si fue una sinapsis excitadora (pesos positivos) o inhibidora (pesos negativos).

- **Función de activación o de transferencia:** $\alpha_i(t)$ Proporciona el estado de activación actual, de la neurona i en función de su estado anterior, $\alpha_i(t-1)$ y de su potencial postsináptico actual.

★ $\alpha_i(t) = f_i(\alpha_i(t-1), h_i(t))$, es la que usualmente se usa.

★ $\alpha_i(t) = f_i(h_i(t))$, en algunos modelos solo se considera que el estado actual no depende del tiempo anterior.

- **Función de salida:** $F_i(\alpha_i(t))$ Da la salida actual, $y_i(t)$, de la neurona i en función de su estado de activación actual. El estado de activación de la neurona se considera como la propia salida.

★ $y_i(t) = F_i(\alpha_i(t))$

★ $y_i(t) = F_i(f_i(\alpha_i(t-1), \sigma(w_{ij}, x_j(t))))$

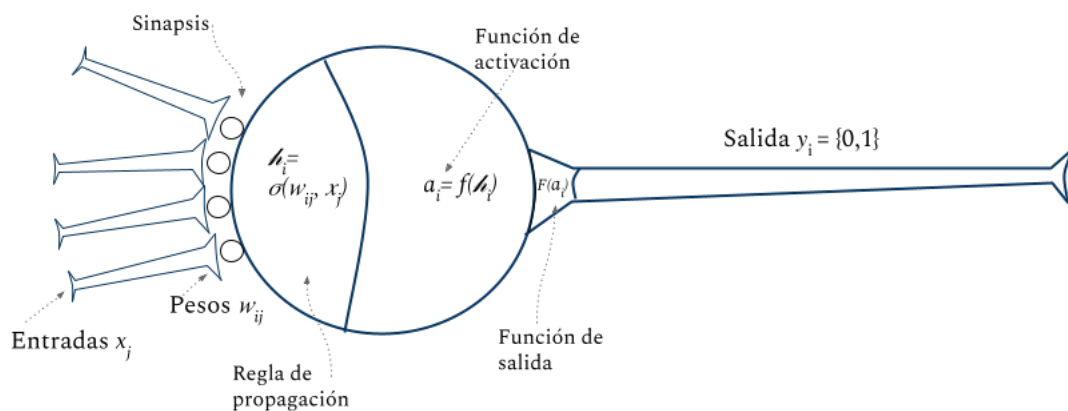


Figura 4.1 Neurona vista como un modelo artificial (perceptrón).

Un perceptrón toma un vector de entradas de números reales, calcula una combinación lineal de estas entradas, luego emite un 1 si el resultado es mayor que algún umbral y -1 de lo contrario. Es decir, dadas las entradas $x_1 \dots x_n$, la salida $o(x_1, \dots, x_n)$ calculada por el perceptrón es 1 si $w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n > 0$ y -1 de lo contrario, donde cada w es una constante \mathbb{R} , un peso, que determina la contribución de la entrada x a la salida del perceptrón. La constante w_0 es un *umbral* (bias) que la suma de las entradas con los pesos debe superar para que el perceptrón emita un 1. En otras palabras es un peso que va a actuar junto con una entrada de valor 1, que vamos a poder ajustar para que nuestra función de activación, se mueva de derecha a izquierda en el plano para ayudarnos a ajustar nuestros resultados, provocando un gran impacto en el aprendizaje.

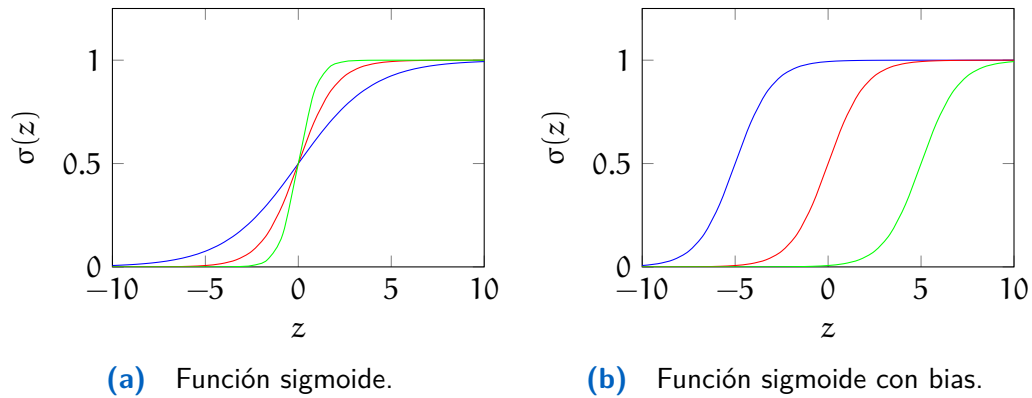


Figura 4.2 Comportamiento de la función de activación (sigmoide) de un perceptrón con una sola entrada, (b) el perceptrón sin el uso de bias, (b) con el uso del bias, donde apesar de estar representandos con la misma entrada , el uso del bias afecta en los resultados de salida. Así si quisieramos que este perceptrón nos diera $y = 0$ con una entrada $x = 2$ sin el uso, ni ajuste del bias sería imposible, pues en (a) apesar que la gráfica azul está la entrada está ajustada con el peso $w = 0.5$, la roja con el $w = 1$, y la verde con el $w = 2$ solo lo gramos alargarla un poco, haciendo que entradas que antes eran correctas ahora caigan 0 también. Entonces lo que necesitamos es más bien "mover"la gráfica, esto lo logramos con la gráfica (b) donde la entrada (única) está sumada con un bias (umbral) $x_0 = 1$, ajustado en azul con peso $w_0 = 5$, en rojo con $w_0 = 0$, en verde con $w_0 = -5$ y el peso $w_1 = 1$. Donde con $w_0 = -5$ logramos nuestro objetivo de tener una salida $y = 0$ con $x = 2$. El bias nos permite mover la función fuera del origen.

Esto se muestra en la siguientes graficas 4.2b. Más adelante hablaremos de su regla de entrenamiento (Training rule).

El hecho de que un perceptrón aprenda implica elegir valores para los pesos denotados también por θ . Ahora, el espacio H de las hipótesis candidatas consideradas en el aprendizaje del perceptrón, es el conjunto de todos los posibles vectores de pesos.

$$H = \{\mathbf{w} | \mathbf{w} \in \mathbb{R}^{n+1}\}$$

Si bien en el momento que se publicó los logros con el modelo del perceptrón las expectativas eran bastante altas, a medida de los años los científicos Marvin Minsky and Seymour Papert desestiman en gran medida los alcances que realmente se puede tener con el perceptrón, al mostrar ¹ que no puede predecir operaciones lógicas que no sean linealmente separables, tal es el caso de la función XOR, que no es separable linealmente, siendo imposible que pueda aprender esta función. Esto y el gran costo que representaba procesar todos los elementos que implicaba el entrenamiento, causa que por un buen

¹En 1969 publican Marvin Minsky y Seymour Papert que perceptrones de una sola capa (simples) solo son capaces de aprender a distinguir patrones linealmente separables, en el libro "Perceptrons".

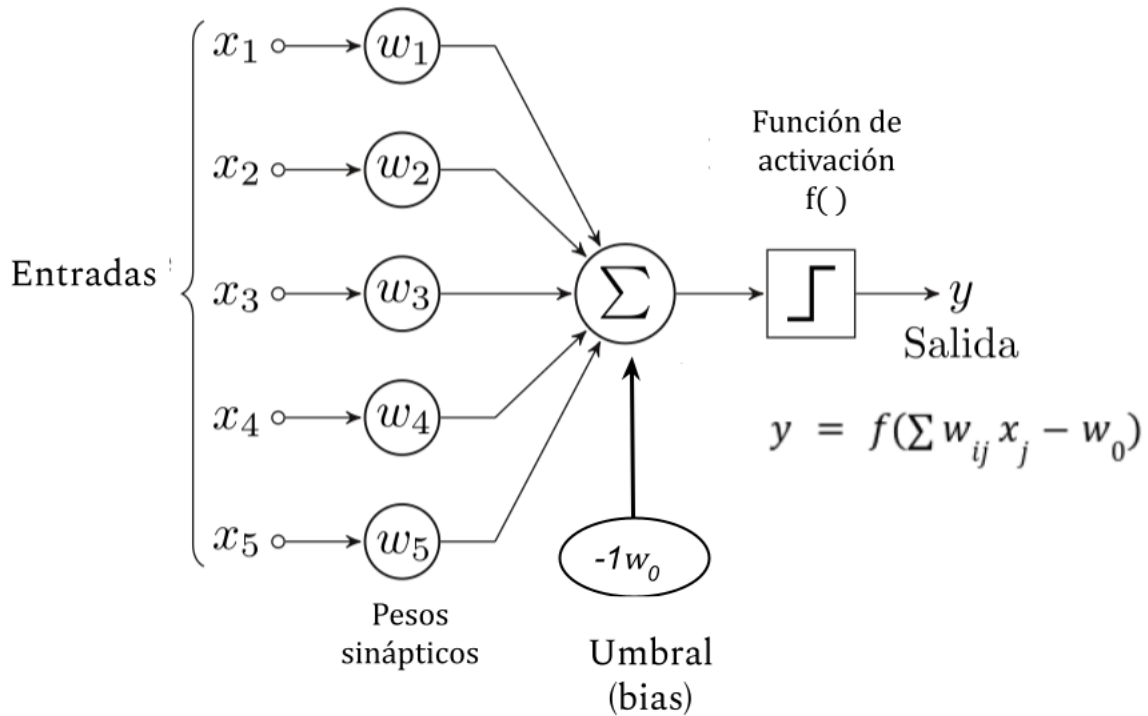


Figura 4.3 Modelo estándar de un perceptrón.

tiempo se desestime el uso del perceptrón. Es hasta después de unas décadas que vuelve a tener relevancia con la propuesta de un perceptrón multicapa usando retropropagación (Feedforward), siendo estos capaces de resolver la función XOR

Compuertas lógicas con neuronas

Aquí se muestra como se puede utilizar un perceptrón para simular compuertas lógicas tales como el or, not, and.

Para simular la compuerta *not*, como está es una función boolean de $B \rightarrow B$, talque $\text{not}(x) = -x$ entonces en un plano de dos dimensiones, la podemos representar con dos puntos, el $p_1 = (0,0)$ y $p_2 = (1,0)$ donde p_1 representa cuando $\text{not}(0) = 1$, p_1 representa cuando $\text{not}(0) = 1$. Teniendo el espacio de la función definido lo que nos toca es, separar el plano para clasificar las entradas, este claramente se puede separar con un linea vertical, o con líneas con pendiente 1 o -1 . Para esta función solo necesitamos de una entrada y un sesgo (bias), donde la entrada la combinaremos con un peso, este peso lo asignaremos a tanteo (por la sencillez de la operación). Así el peso $w_1 = -1$ y el peso asignado al bias sera $w_0 = 0.5$, ahora con esto datos podemos:

- Hacer la función de propagación donde $h(x) = (x * -1) + (0.5) * 1 = 0.5 - x$

- Hacer la función de activación escalón $\alpha(x) = \text{sgn}(h) = \text{sgn}(0.5 - x)$
- Dar la salida donde $s(1) = \text{sgn}(0.5 - 1) = 0$ y $s(0) = \text{sgn}(0.5 - 0) = 1$, en este caso la salida es la identidad de la activación.

x	h	s
0	0.5	1
1	-1.5	0

Algo similar va a pasar con la compuerta **and** y **or** donde al necesitar de dos entradas para la compuerta, asignaremos dos entradas para el perceptrón igualmente y las representaremos en el plano con cuatro puntos, donde cada punto representa una instancia y se le asigna valor positivo o negativo en el plano, así pues para el and tenemos los puntos $p_1 = (0, 0)$, $p_2 = (0, 1)$, $p_3 = (1, 0)$, negativos y $p_4 = (1, 1)$ el único positivo. Así nos damos cuenta que necesitamos una recta con pendiente negativa y fuera del origen, que nos separe estas clases de puntos. Por tanto para el bias le asignamos un peso de $w_0 = -1.5$, $w_1 = 1$ y $w_2 = 1$, con estos datos podemos:

- Hacer la función de propagación donde $h((x_1, x_2)) = (x_1 * 1) + (x_2 * 1) + (-1.5) * 1 = x_1 + x_2 - 1.5$
- Hacer la función de activación escalón $\alpha((x_1, x_2)) = \text{sgn}(h) = \text{sgn}(x_1 + x_2 - 1.5)$
- Dar la salida donde $s = \alpha(x)$, es la identidad de la activación.

x_1	x_2	h	s
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

Para la compuerta **or** es algo muy similar pues podemos igualmente representar la función con cuatro puntos en el espacio cada uno representando una instancia, solo que ahora tres de estos puntos serán positivos y solo uno negativo, los puntos positivos serían $p_2 = (0, 1)$, $p_3 = (1, 0)$ y $p_4 = (1, 1)$, mientras que $p_1 = (0, 0)$ negativo, el plano lo podemos dividir con una línea recta con pendiente negativa, así le asignamos $w_0 = -0.5$, $w_1 = 1$ y $w_2 = 1$, con esto hacemos los pasos que ya sabemos:

- Hacer la función de propagación donde $h((x_1, x_2)) = (x_1 * 1) + (x_2 * 1) + (-0.5) * 1 = x_1 + x_2 - 0.5$
- Hacer la función de activación escalón $\alpha((x_1, x_2)) = \text{sgn}(h) = \text{sgn}(x_1 + x_2 - 0.5)$

- Dar la salida donde $s = \alpha(x)$, es la identidad de la activación.

x_1	x_2	h	s
0	0	-0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	-1.5	1

Tomando el hecho que en la naturaleza las neuronas van pasando información en una estructura que forma niveles de abstracción, esto lo modelamos como capas de neuronas conectadas entre sí, cada capa haciendo su trabajo de abstracción. El perceptrón simple es un modelo neuronal unidireccional, una capa de entrada y otra de salida, que por si solo no puede separar todas la funciones lógicas pues tenemos el XOR, para resolver esto usaron perceptrones multicapa que se explicará más adelante en el curso.

Funciones de activación

Recordando que la forma de las funciones de activación es $y = f(x)$, donde x representa el potencial postsináptico e y el estado de activación de la neurona, es decir si va a lanzar un disparo o no. Las funciones de activación más empleadas son:

Funciones de error

Entonces tomando como base el perceptrón, una vez obtenidas las salidas con una primera iteración nos daremos cuenta de que tan lejos o que tan cerca estuvimos de la respuesta correcta, con esto darnos la oportunidad de que pesos ajustar respecto a sus entradas, en la segunda iteración. Ahora para facilitarnos esto y tomando en cuenta que el entrenamiento consiste en varias iteraciones hasta llegar a aprender la tarea T , hacemos uso de una función de error que nos ayude a minimizar la diferencia de error en las salidas.

Primero veamos el entrenamiento para una sola neurona, para esto haremos uso de la regla de aprendizaje del perceptrón (learning rule perceptron), donde para cada entrada, en la capa de salida se le calcula la desviación a la función objetivo. El cual utilizamos para ajustar los pesos del perceptrón (ver fig 4.5).

Usualmente al principio del entrenamiento se asignan pesos aleatorios, a medida que avance el entrenamiento, se van modificando con cada iteración, así $w_i \leftarrow w_i + \Delta w_i$. Esto con base a [la regla de aprendizaje](#) donde:

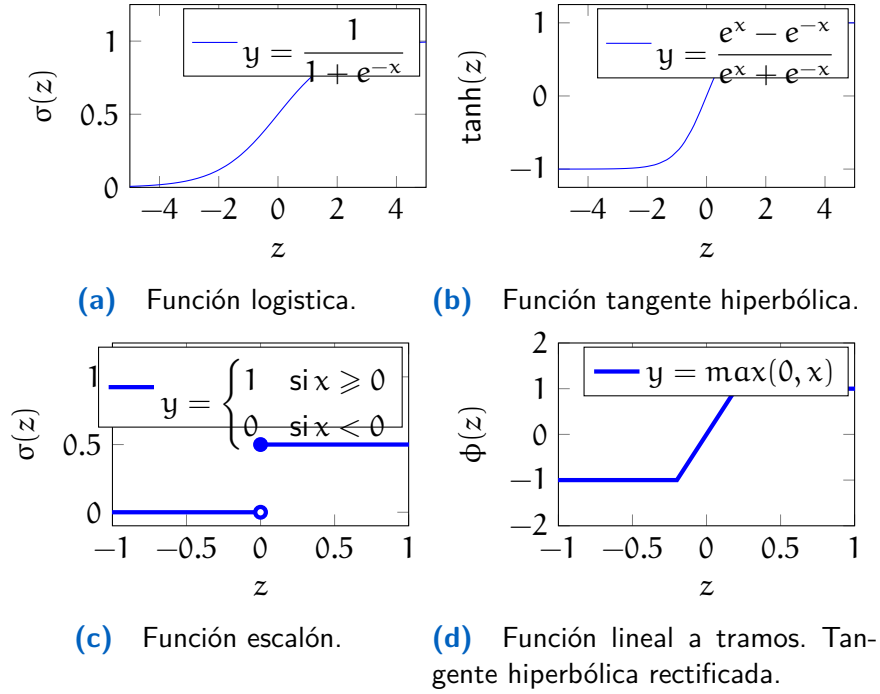


Figura 4.4 Las funciones de activación más usadas son la función sigmoide $\sigma(z)$ y la tangente hiperbólica $\tanh(z)$.

$$\Delta w_i = \alpha(y - y_{out})x_i \quad (4.1)$$

Con y es la salida deseada, y_{out} la salida generada, α la tasa de aprendizaje (learning rate) y x_i la entrada i . Lo que hace la tasa de aprendizaje es moderar el grado en que los pesos son cambiados con cada iteración, se le asigna un valor muy pequeño (0.1 o 0.2) y conforme se logran ajustar los pesos se minimiza aún más.

Para entrenar un perceptrón, utilizamos cualquier método de optimización de funciones para encontrar los parámetros w que minimizan el error con alguna de las siguientes funciones de error:

Diferencias al cuadrado, también conocida como regla aprendizaje delta, es la suma de cuadrados de errores que se tuvieron con cada ejemplar del conjunto de entrenamiento. Los podemos describir como:

$$\frac{1}{2m} \sum_{m=0}^{M-1} (y_m - a_m)^2 \quad (4.2)$$

con y_m y a_m , la salida obtenida dado un ejemplar m y la salida correcta del ejemplar m respectivamente.

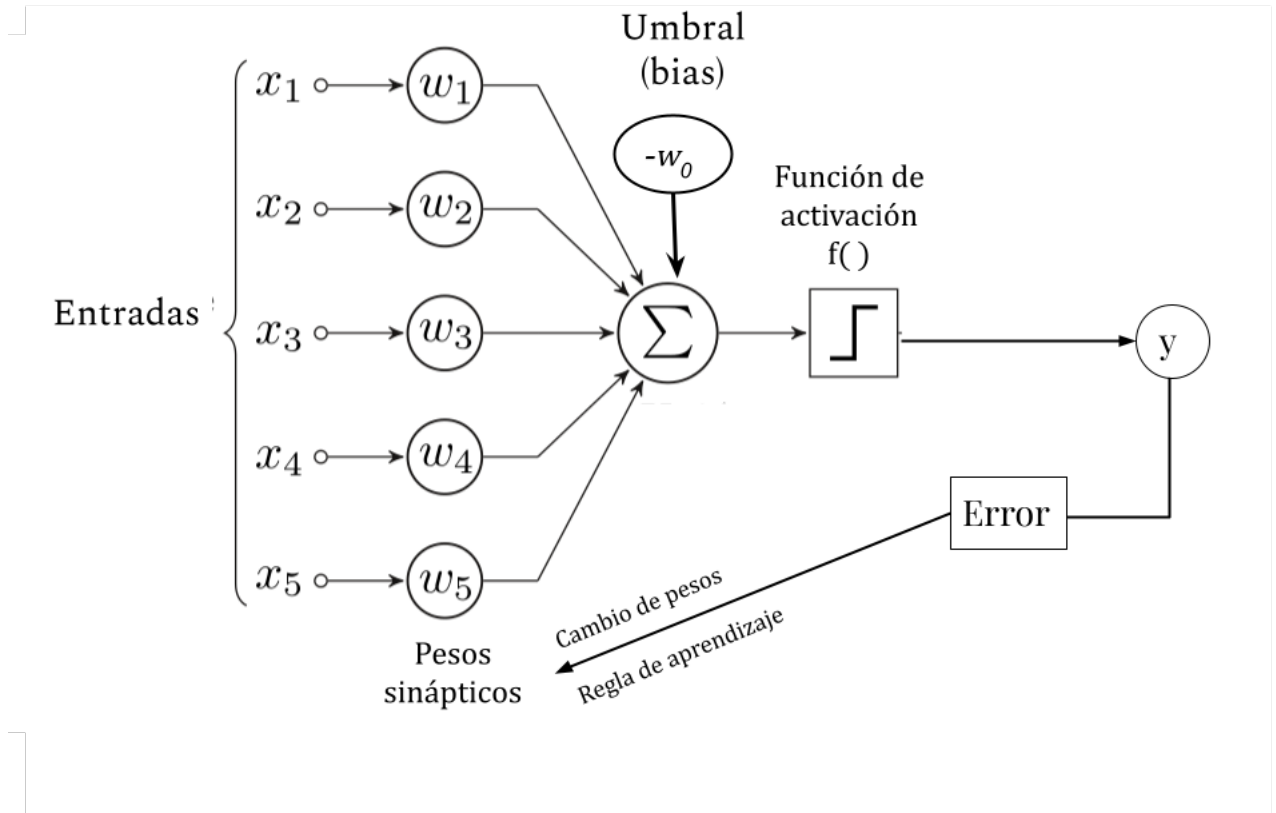


Figura 4.5 Modelo estándar de un perceptrón.

Entropía cruzada. Se usa para problemas de clasificación ya que su comportamiento es más suave (soft) y permite hacer clasificaciones más certeras. Se define como:

$$L(\Theta) = -\frac{1}{m} \sum_{m=1}^m (y^{(m)} \log(a_m) + (1 - y_m) \log(1 - a_m)) \quad (4.3)$$

Entonces juntando los conceptos que ya sabemos podemos describir el algoritmo de entrenamiento para el perceptrón de la siguiente forma:

1. Iteramos sobre todos los ejemplares.
2. Para cada ejemplar se calcula la función de propagación, es decir la suma ponderada.
3. Se calcula la función de activación con esta suma.

4. Se calcula la función de salida.
5. Actualización de pesos de acuerdo a la regla de aprendizaje.
6. Repetir de 1-6 hasta que los pesos nos satisfagan, un número de iteraciones establecidas.

Medidas de rendimiento

Las medidas de rendimiento de una red nos sirven para ver de manera concreta como se comportó nuestra red, durante el entrenamiento. Que fue lo que pudo aprender (clasificar). Si tuvo un sobreajuste, es decir, memorizo y por tanto, será incapaz de predecir la clase a la que pertenece realmente el ejemplar. Para esto se usan las siguientes herramientas:

Matriz de confusión : (Confusion matrix) Es una matriz, donde las celdas representan las predicciones que hizo nuestro modelo de clasificación de clases, respecto a las salidas esperadas. Así siendo las columnas las salidas y 's del modelo entrenado y las filas las salidas esperadas y_{true} . Nos facilita a ver cuando un clasificador está confundiendo clases, contabilizando a que clase etiqueto a los diferentes ejemplares. Ahora veamos, que puede estar representando cada celda en la matriz de confusión, estos pueden ser:

1. **VP** Verdaderos positivos (*TP, True Positive*): La clasificación de los ejemplares predichos, conciden con las etiquetas esperadas de los ejemplares.
2. **VN** Verdaderos negativos (*TN, True Negative*): El ejemplar que no es parte de clase, no son asignados a esa clase, son predichos correctamente.
3. **FP** Falso positivo (*FP, False Positive*): El ejemplar que **no** es parte de una clase i fue clasificado como tal.
4. **FN** False negativo (*FN, False Negative*): El ejemplar que es parte de una clase i no fue clasificado como tal.

Ejemplo 4.1. *Notemos un ejemplo sencillo, supongamos que tenemos una tarea binaria, donde queremos indicar que una persona está embarazada (de acuerdo a unos estudios). Ahora tenemos que:*

- *VP*, sería predecir que una mujer está embarazada y que en efecto esté embarazada. (**Correcto**)
- *VN*, sería con un hombre que no está embarazado y pues en efecto no está embarazado. (**Correcto**)
- *FP*, sería predecir que un hombre está embarazado y **no** este embarazado. (**Error, tipo 1**)

- **FN**, sería predecir que una mujer **no** está embarazada y esta embarazada. (**Error, tipo 2**)

Entonces dados los resultados binarios que nos entregó el modelo, los médicos nos dan las respuestas correctas a 10 estudios, representadas en la siguiente tabla.

Ejemplar	1	2	3	4	5	6	7	8	9	10
Sujeto	M	F	M	F	M	M	F	F	F	M
Etiqueta	No	Si	No	Si	No	No	No	Si	No	No
Clase	0	1	0	1	0	0	0	1	0	0
Predicho	0	0	0	1	0	1	0	1	0	0
Valores	VN	FN	VN	VP	VN	FP	VN	VP	VN	VN

Con estos datos, podemos construir nuestra matriz de confusión MC contabilizando las salidas obtenidas respecto a las deseadas. Quedando así de la siguiente manera:

		Salidas y	
		Si	No
Etiquetas	Si	VP = 2	FN = 1
	No	FP = 1	VN = 6

Tabla 4.1 Matriz de confusión binaria.

Ahora para una modelo que sea multiclase, en el que tengamos que clasificar varias clases de ejemplares. Tendremos una matriz M de $n * n$ donde n es el número de clases, así los valores **VP**, **VN**, **FP**, **FN**, son calculados para cada clase e identificados en la matriz M de la siguiente forma, también puedes ver la figura 4.6:

- **VP**, para la clase i será la celda $M[i][j]$ con la $i = j$.
- **VN**, para la clase i será la suma de los valores de toda la matriz menos los valores de la fila i ni los valores de la columna i .
- **FN**, para la clase i será la suma de los valores en la fila i , excepto la celda $M[i][j]$, con $i = j$ que es el **VP**.
- **FP**, para la clase i será la suma de los valores en la columna i , excepto la celda $M[i][i]$ que es el **VP**.

Valores para la clase i		Salidas y del clasificador			
		clase 1	clase 2	clase i	clase n
Etiquetas ytrue	clase 1	VN	VN	FP	VN
	clase 2	VN	VN	FP	VN
	clase i	FN	FN	VP	FN
	clase n	VN	VN	FP	VN

Figura 4.6 Matriz de confusión para clasificador multiclases.

Ejemplo 4.2. Tenemos un clasificador encargado de identificar las cinco vocales del español, escritas a mano. Entonces tenemos un total de 5 clases representadas por a, e, i, o, u, cada letra representada por un número del 0 al 4. Así la clase 0 = a, la clase 1 = e y así respectivamente. Este fue entrenado con 15 ejemplares etiquetados y se obtuvieron los siguientes resultados:

Ejemplar e	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Etiqueta	a	e	i	o	u	a	a	e	i	i	o	u	o	u	a
Clase y _{true}	0	1	2	3	4	0	0	1	2	2	3	4	3	4	0
Predicho y	0	1	3	3	2	1	1	1	2	2	4	4	3	2	0

Ahora para hacer la matriz de confusión MC notamos que tenemos una matriz de 5x5, donde $MC[i] = \text{Predicho } y$ y $MC[j] = \text{Etiquetas Reales}$. Entonces necesitamos contabilizar las predicciones y asignarlas a sus respectivas celdas, donde con la tabla anterior notamos que en el ejemplar e3 con $y_{\text{true}}(e3) = 2$, se predijo que $y(e3) = 3$, entonces $MC[2][3] += 1$, pues la etiqueta nos posiciona en la fila 2 y lo predicho en la columna 3. Ahora en el ejemplar e4 con $y_{\text{true}}(e4) = 3$ se predijo que $y(e4) = 3$, entonces $MC[2][2] += 1$. Así con cada ejemplar vamos a ir sumando su clasificación. Para construirla tendríamos un pseudocódigo siguiente:

```
def getMC (Ejemplares, y):
    """
    :param y: salidas obtenidas (int)
    """
    MC = [len(y)][len(Ejemplares)] # y == Ejemplares
    full_zeros(MC)
    for e in range (0, len(Ejemplares)):
        predicho = y[e]
        y_true = Ejemplares[e].etiqueta
```

```

        MC[predicho][y_true] += 1
    return MC

```

Dados los datos anteriores tenemos, la siguiente matriz de confusión:

		Salidas y				
		1	2	3	4	5
Etiquetas	1	2	2	0	0	0
	2	0	2	0	0	0
	3	0	0	2	1	0
	4	0	0	0	2	1
	5	0	0	2	0	1

Tabla 4.2 MC = Matriz de confusión multiclase.

Para calcular los valores *VP*, *VN*, *FP*, *FN*, por clase se propone el siguiente pseudo-código:

```

MC = getMC(y_salidas, y_true)

VP = VN = FP = FN = [0,0,0,0,0]

for i in range(len(MC)):
    for j in range(len(MC[0])):
        FN[i] = FN[i] + MC[i][j] if (i != j) else FN[i]
        FP[i] = FP[i] + MC[j][i] if (i != j) else FP[i]
        VP[i] = MC[i][j] if (i == j) else VP[i]
    VN[i] = sum(MC) - FN[i] - FP[i] - VP[i]

```

Si quisiéramos saber los valores *VP*, *VN*, *FP*, *FN*, para todo el desempeño total, simplemente sumamos lo obtenido en cada clase así:

```

VP_total = sum(VP)
VN_total = sum(VN)
FP_total = sum(FP)
FN_total = sum(FN)

```

Ahora los valores de cada celda nos representan lo siguiente:

- VP_{total} , toda la diagonal de la matriz. La salida del modelo coincide con lo etiquetado con el ejemplar.
- VN_{total} , todas las veces que no era la clase i y dijo que no era de la clase i .
- FN_{total} , todas las veces que era clase i y dijo que era clase x . (deseado $FN = 0$)
- FP_{total} , todas las veces que predijo que era clase i y era clase x . (deseado $FP = 0$)

Así que, si los valores que no están en la diagonal de la matriz son cero o todas nuestras clasificaciones están en la diagonal, podemos decir que nuestro modelo aprendió a clasificar correctamente todas las clases.

Precisión y recuperación : La precisión (*precision*) nos dice la proporción de ejemplares que se logró clasificar correctamente. Mientras que recuperación (*recall*) nos dice cuantas asignaciones de lo que nos interesa pudo clasificar correctamente en otras palabras donde del total de las respuestas correctas que se pueden tener, cuantas respuestas positivas acertadas se tuvo.

$$P = \frac{VP}{VP + FP} = \frac{\text{VerdaderosPositivos}}{\text{ValoresEsperados}} \quad (4.4)$$

$$R = \frac{VP}{VP + FN} = \frac{\text{VerdaderosPositivos}}{\text{ValoresPredichos}} \quad (4.5)$$

Valores para la clase i		Salidas y del clasificador			
		clase 1	clase 2	clase i	clase n
Etiquetas y true	clase 1	VN	VN	FP	VN
	clase 2	VN	VN	FP	VN
	clase i	FN	FN	VP	FN
	clase n	VN	VN	FP	VN

Recall = $VP / (VP + FN)$
Cuantos valores esperados fueron asignados realmente.

Precisión = $VP / (VP + FP)$
Cuanto de lo que clasifiqué fue correcto

- Cuando el modelo detecta los ejemplares, pero los incluye en otras clases también: P es bajo y R es alto.
- Cuando el modelo no detectó bien los ejemplares, pero tampoco los incluyó en otras clases: P es alto y R es bajo.

- Cuando el modelo detecta los ejemplares y no los incluye en otras clases: P y R es alto.
- Cuando el modelo no detecta los ejemplares: P y R es bajo.

En ocasiones le daremos más importancia al recall y en otras a la precisión. Retomando el ejemplo previo 4.1, no nos importa tanto los casos de error tipo 1, donde el modelo se equivoca con los ejemplares negativos, nos importa los errores de tipo 2, los Falsos Negativos, en este caso le tomaremos especial atención al Recall y que este sea alto. Pero si bien nuestra tarea es detectar cuando un correo es spam, no impacta tanto que aunque en ocasiones un correo sea spam, no lo clasifique como tal (error tipo 2 FN), nos es crucial que un correo que no sea spam nos lo clasifique como tal (error tipo 1 FP). Así deseando que los falsos positivos se acerquen a cero, dándonos como resultado una precisión alta aunque el recall sea bajo.

Exactitud y medida f : La exactitud (*accuracy*) es una medida de cuántas predicciones correctas en total hizo el modelo para el conjunto de datos completo (no se recomienda usar si tienes clases desbalanceadas, es decir, muchos elementos de una clase y poco de otra pues nos puede fallar totalmente con las clases pequeñas y aun así su valor sería alto). La medida f (*f score*) se utiliza para combinar las medidas de precisión y recall en un solo valor. Es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la recall. Se dan las ecuaciones a continuación:

$$\text{Accuracy} = A = \frac{VP + VN}{VP + VN + FP + FN} = \frac{VP + VN}{\text{TodosLosValoresClasificados}} \quad (4.6)$$

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} = 2 \frac{P * R}{P + R} \quad (4.7)$$

La medida f, también la podemos escribir (con un poco de aritmética) como:

$$F = \frac{2VP}{2VP + FP + FN} \quad (4.8)$$

5 | Perceptrón multicapa

Intro

Vacio, si vacio.

XOR

Vacio no vacio.

Propagación hacia adelante manual

Vacio no vacio.

Propagación hacia adelante vectorizada (con matrices)

Vacio no vacio.

Interpretación matemática del mapeo no lineal

Vacio no vacio, pero si vacio :P.

Propagación hacia adelante para el perceptrón multicapa

Cuasi vacio.

6 | Entrenamiento por retropropagación

Esquema general entrenamiento

Vacio pero no vacio.

Función de error: Entropía cruzada

Vacio pero no vacio.

Derivada de la función logística

Vacio, pero no vacio :P

Entrenamiento en la última capa

Vacio no vacio pero si vacio :P

Parcial con respecto a los pesos en la última capa

Vacio >:|

Vectorización

La vectorización.

7 | Optimización del entrenamiento

Problemas en redes profundas

Vacio.

Gradiente desvaneciente (o que explota)

Vacio.

Estilos de entrenamiento

Vacio.

Normalización y normalización por lotes

Vacio.

Regularización

Vacio.

8 | Caso de análisis e interpretación

Red Hinton árbol familiar con numpy (entrenamiento)

Vacio.

Red Hinton árbol familiar con pytorch

Vacio.

9 | Entrenamiento con genéticos

Algoritmos genéticos

Vacio.

Neuroevolución

Vacio.

Antecedentes: Aprendizaje por refuerzo en videojuegos

Vacio.

Arquitectura para estimar la función de recompensa

Vacio.

Entrenamiento

Vacio.

10 | Mapeos autoorganizados

Introducción

Vacio.

Aprendizaje no supervisado

Vacio.

Mapeos autoo-organizados

Vacio.

Kohonen

Vacio.

11 | Redes Neuronales Convolucionales

Convolución

Redes Convolucionales

Softmax

MNIST

PARTE III

Redes con ciclos

12 | Redes Neuronales Recurrentes

Derivadas ordenadas

Retropropagación en el tiempo

Sistemas dinámicos y despliegue del grafo

Arquitectura recurrente universal

Función de error

Forzamiento del profesor

13 | Atención

14 | LSTM

15 | GRU

16 | Casos de análisis: etiquetado de palabras y conjugación de verbos

PARTE IV

Redes no dirigidas

17 | Redes de hopfield

Entrenamiento

18 | Máquinas de Boltzman

Entrenamiento

Partículas y partículas de fantasía

Máquinas de Boltzman Restringidas

19 | Redes adversarias

GANs

A | Ecuaciones diferenciales

Bibliografía

- Waldeyer-Hartz, Wilhelm von (1891). «Forschungen im Gebiete der Anatomie des Zentralnervensystems. Deutsche medizinische Wochenschrift 1891; 17 (44)». En: (Sobre algunas investigaciones recientes en el campo de la anatomía del sistema nervioso central)volumen 17, número 44.
- Doidge, Norman (2007). *The Brain That Changes Itself: Stories of Personal Triumph from the Frontiers of Brain Science*. Penguin Life; Reprint edition.
- Francisco López-Muñoz, Jesús Boya y Cecilio Alamo (2006). «"Neuron theory, the cornerstone of neuroscience, on the centenary of the Nobel Prize award to Santiago Ramón y Cajal"». En: *Brain Research Bulletin*. 70 (4–6): 391–405. doi:10.1016/j.brainresbull.2006.07.010.
- John G. Nicholls, A. Robert Martin (2011). *From Neuron to Brain*. Sinauer Associates Inc.
- Kandel, Eric R. (2012). *Principles of Neural Science, Fifth Edition*. McGraw-Hill Education.
- Mesulam, M Marsel (1998a). *From sensation to cognition*. Brain 121.
- (1998b). «From Sensation to Cognition». En: *Brain* 121, págs. 1013-1052.
- Snell, Richard S. (2001). *Neuroanatomía Clínica*. En español. 3pp,50pp, varias más. Editorial medica panamericana.
- (2009). *Clinical Neuroanatomy*. 3pp. Lippincott Williams y Wilkins.
- Trappenberg, Thomas P. (2010). *Fundamentals of Computational Neuroscience*. Oxford University Press Inc.