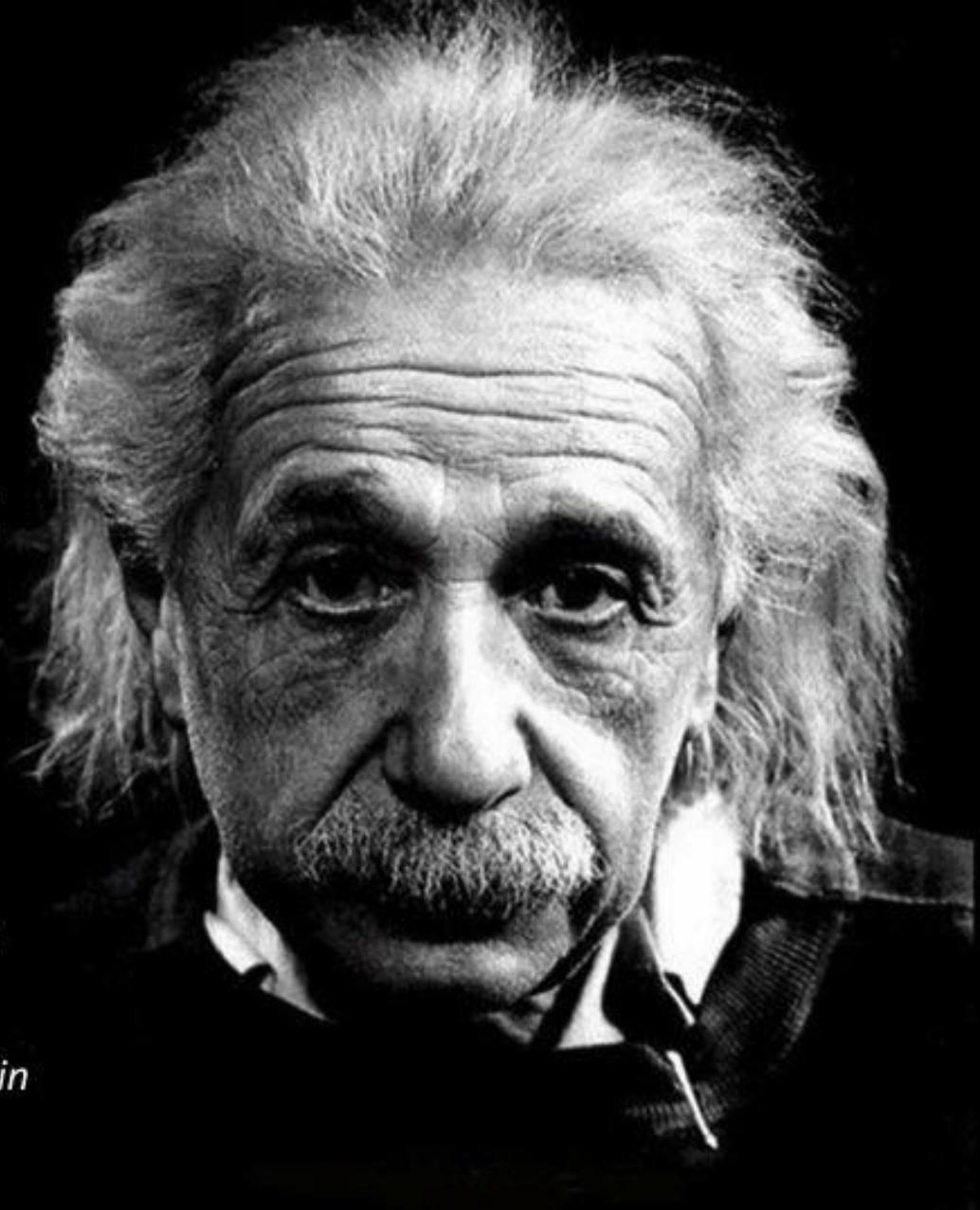


"If I had an hour to solve a problem and my life depended on the solution, I would spend the first 55 minutes determining the proper question to ask, for once I know the proper question, I could solve the problem in less than 5 minutes."

*- Albert Einstein*





ESCUELA DE INGENIERÍA  
FACULTAD DE INGENIERÍA

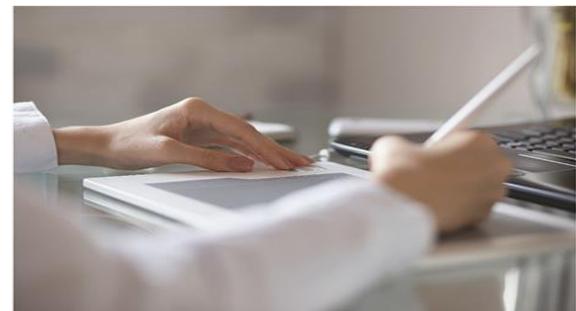
EDUCACIÓN  
PROFESIONAL

# Diplomado Big Data y Analítica de Datos 2021

## Curso: Fundamentos Machine & Deep Learning

Educación Profesional Escuela de Ingeniería UC

Profesor: Rodrigo Sandoval





ESCUELA DE INGENIERÍA  
FACULTAD DE INGENIERÍA

EDUCACIÓN  
PROFESIONAL

# 4. Deep Learning - Introducción

# Contenido

- 4.1. El diseño de las redes neuronales artificiales.
- 4.2. Motivación para desarrollar Deep Learning.
- 4.3. El impacto del Deep Learning en la industria.
- 4.4. Los conceptos esenciales en Deep Learning.
  - 4.4.1. Álgebra lineal
  - 4.4.2. Funciones de activación
  - 4.4.3. Entrenamiento de redes neuronales profundas.
  - 4.4.4. Overfitting y regularización
  - 4.4.5. Algunas decisiones de arquitectura

Repasemos ...

## **4.1. REDES NEURONALES ARTIFICIALES**

# Redes Neuronales Artificiales (ANN)

Se conoce generalmente a McCulloch & Pitts (1943) como los diseñadores de la primera red neuronal.

Muchas de esas ideas iniciales aún se usan hoy en día (por ej, muchas unidades simples se combinan para lograr un poder computacional aumentado y la idea de un umbral – threshold)

Hebb (1949) desarrolló la primera regla de aprendizaje (en la premisa que si dos neuronas estaban activas al mismo tiempo, la fuerza entre ellas se incrementa).

En los '50s y '60s varios investigadores trabajaron en “el perceptrón” con mucha expectativa.

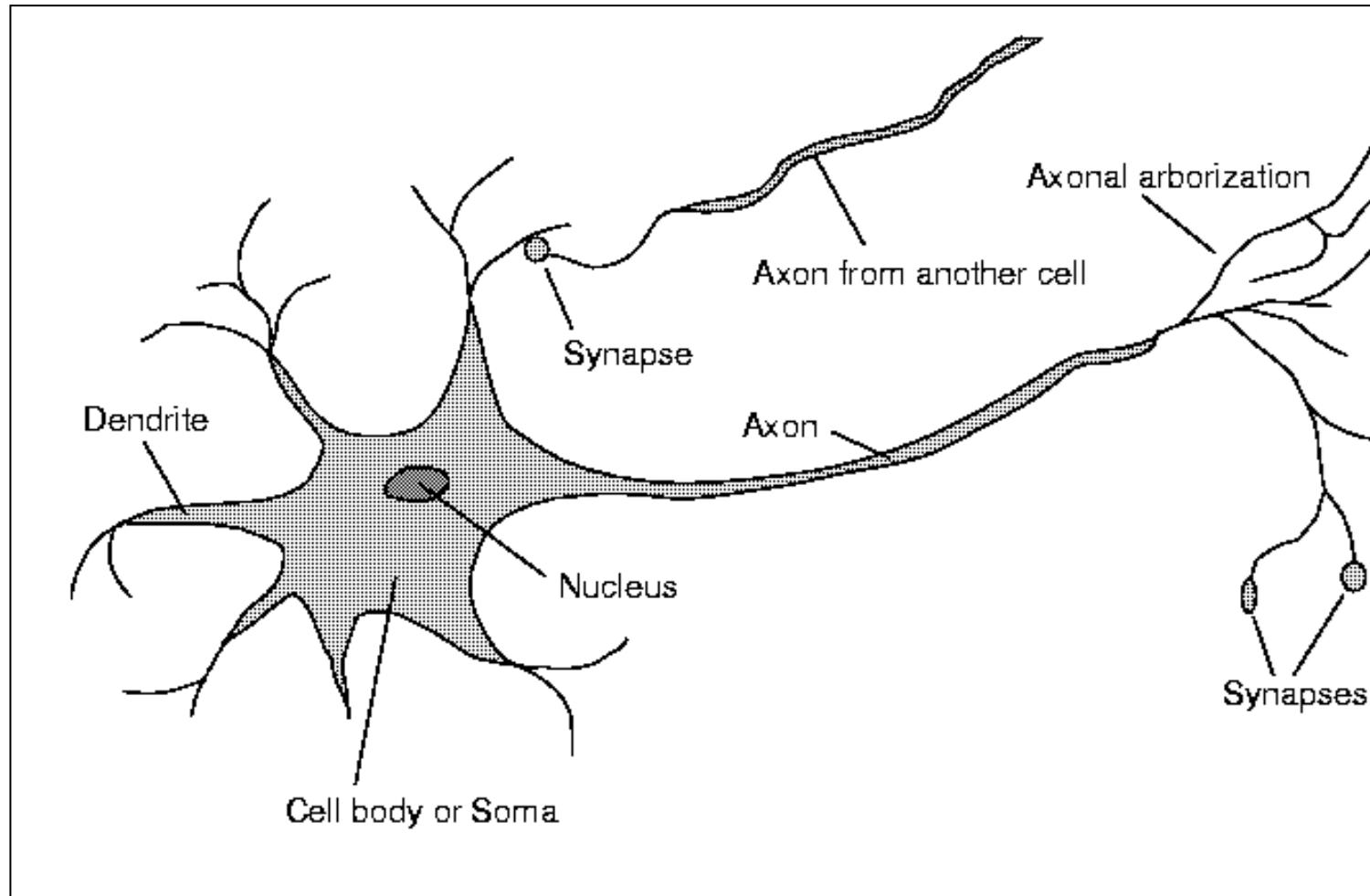
En los '70s vino el invierno de la IA y detuvo la investigación por 15 años.

En los '80s gracias a Parker y Yann LeCun se revivió el interés.  
El algoritmo de entrenamiento sale de Werbos en 1974.



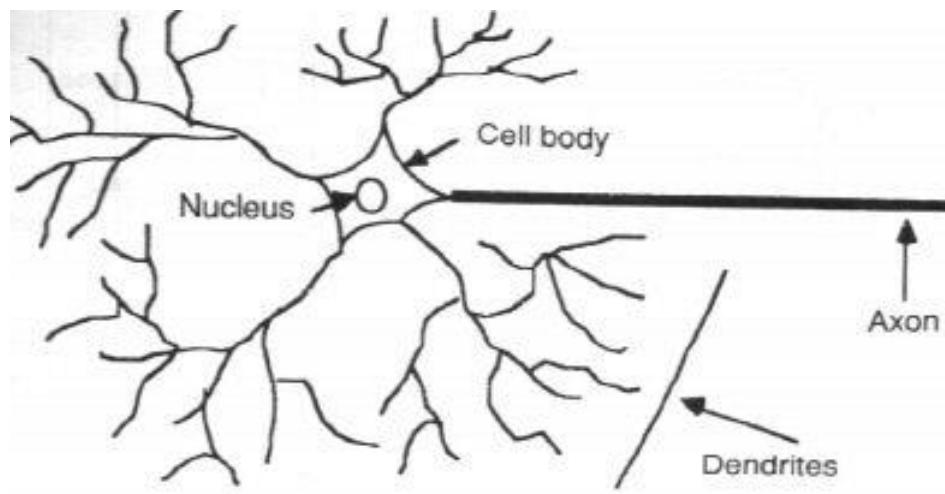
Imagen del primer Perceptrón,  
desarrollado por la oficina de  
investigación de Fuerza Naval  
de EEUU, en julio de 1958.  
“La primera máquina capaz de  
tener una idea original”.

# Redes Neuronales (biológicas)

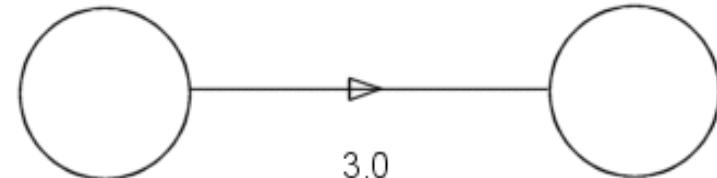
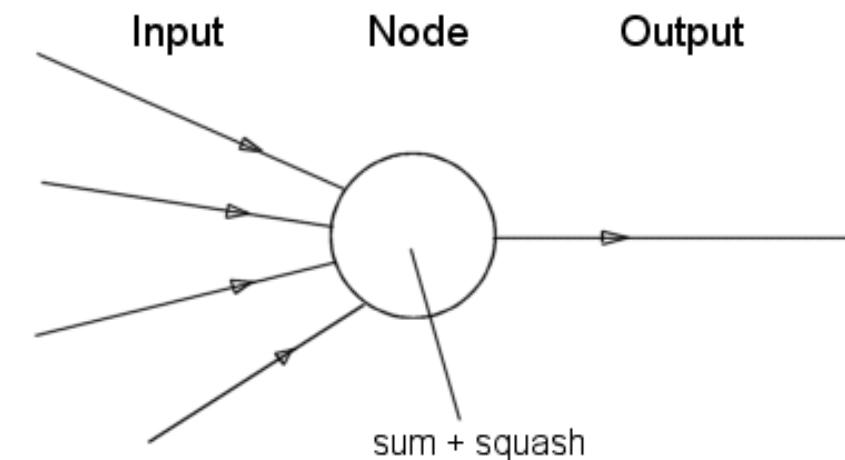
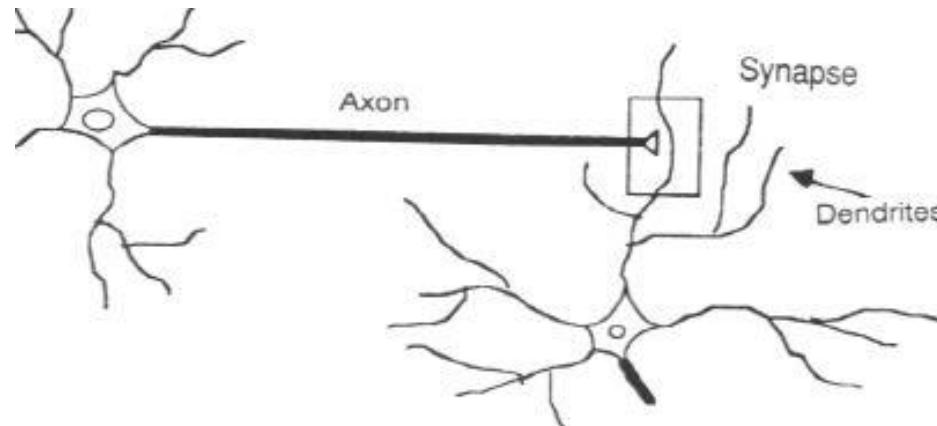


# Equivalencias biológica-artificial

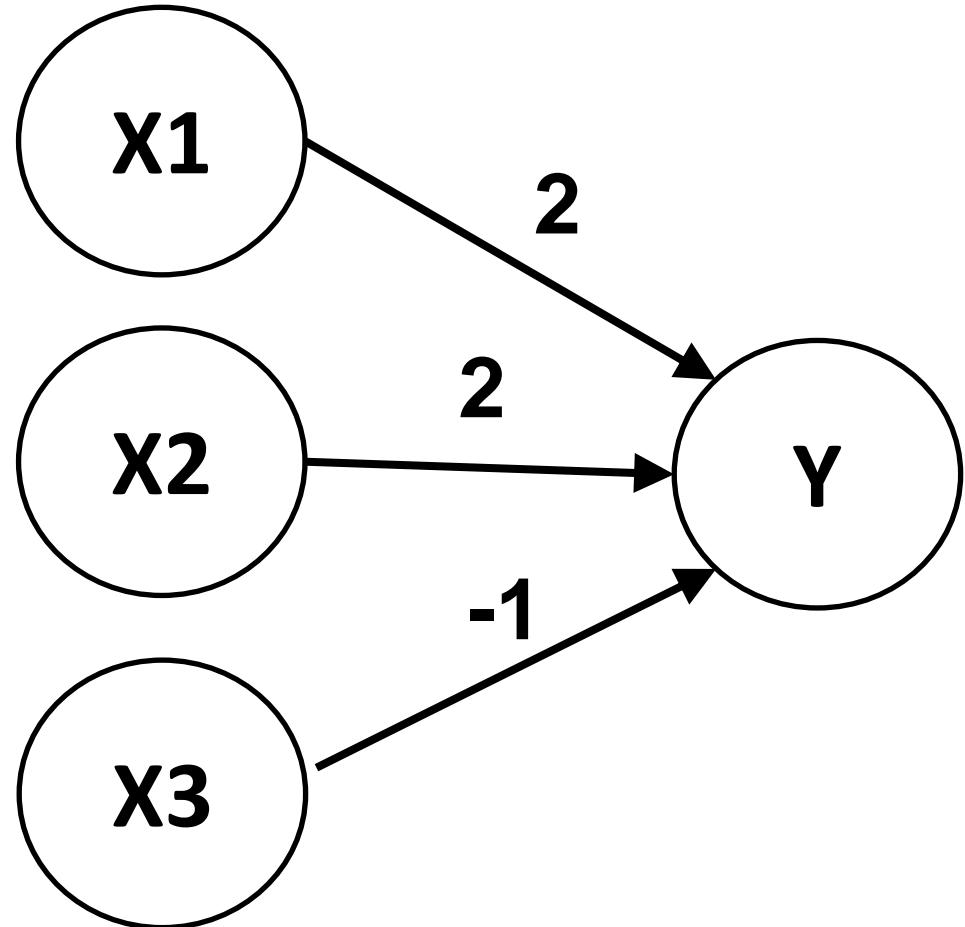
Neurona  
vs. Nodo



Sinapsis  
vs. Peso

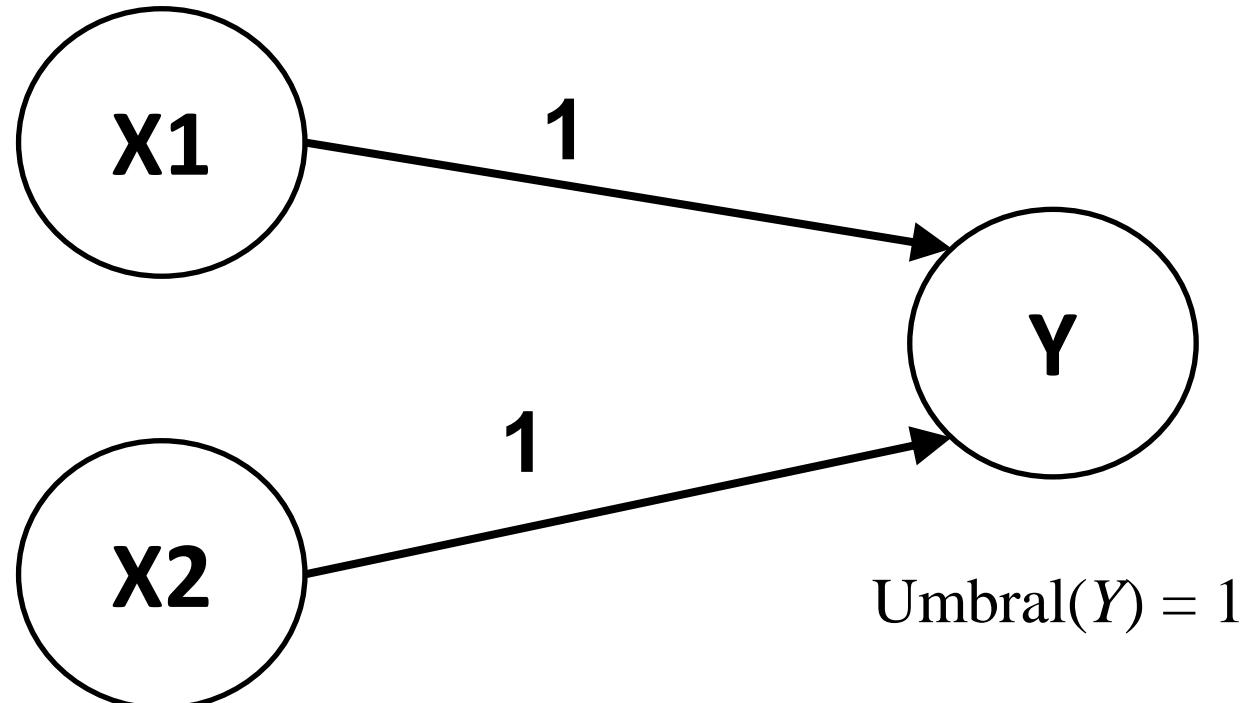


# Red Neuronal Artificial Simple



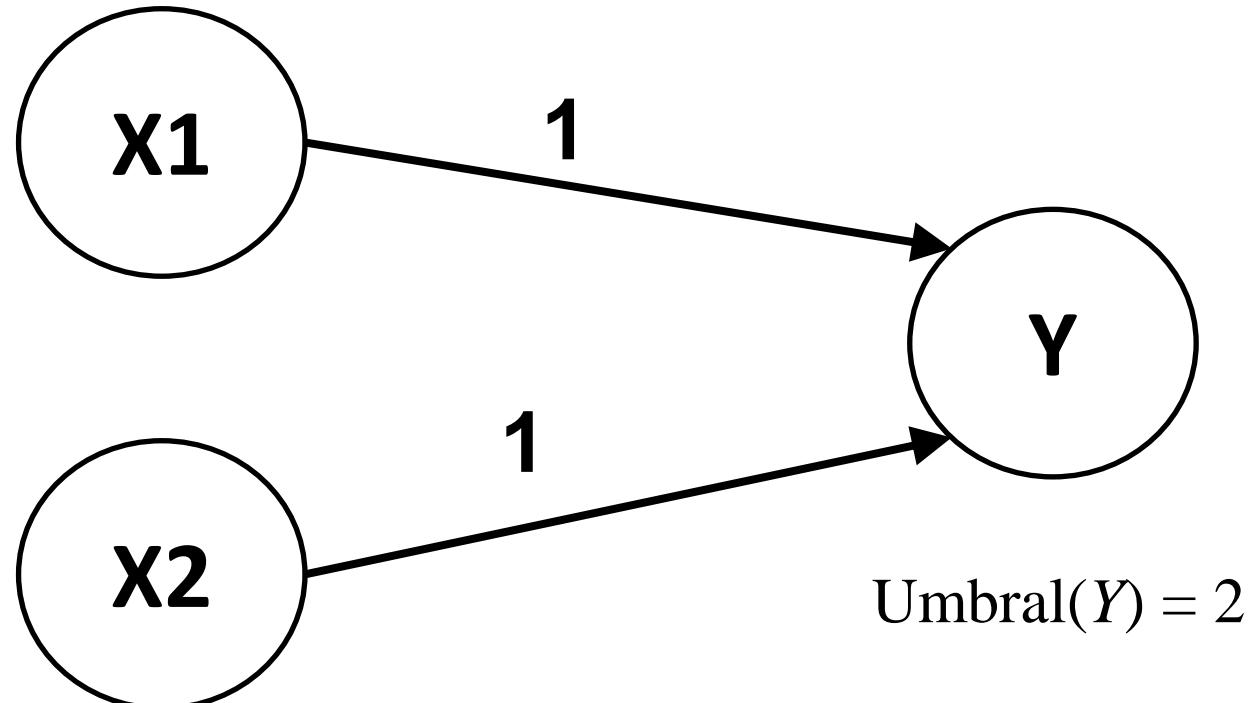
- Las primeras neuronas de McCulloch-Pitts se conectan por rutas ponderadas (con peso) directas.
- La activación de la neurona es binaria: se activa o no.
- Si el peso de una ruta es positivo, excita, si no, es inhibitoria.
- Todas las conexiones excitadoras a una neurona en particular tienen el mismo peso, aunque diferentes conexiones ponderadas pueden ser entrada para diferentes neuronas.
- Cada neurona tiene un umbral de activación fijo. Si la entrada combinada desde la red es mayor que el umbral, esa neurona se activa.

# Red Neuronal Artificial Simple



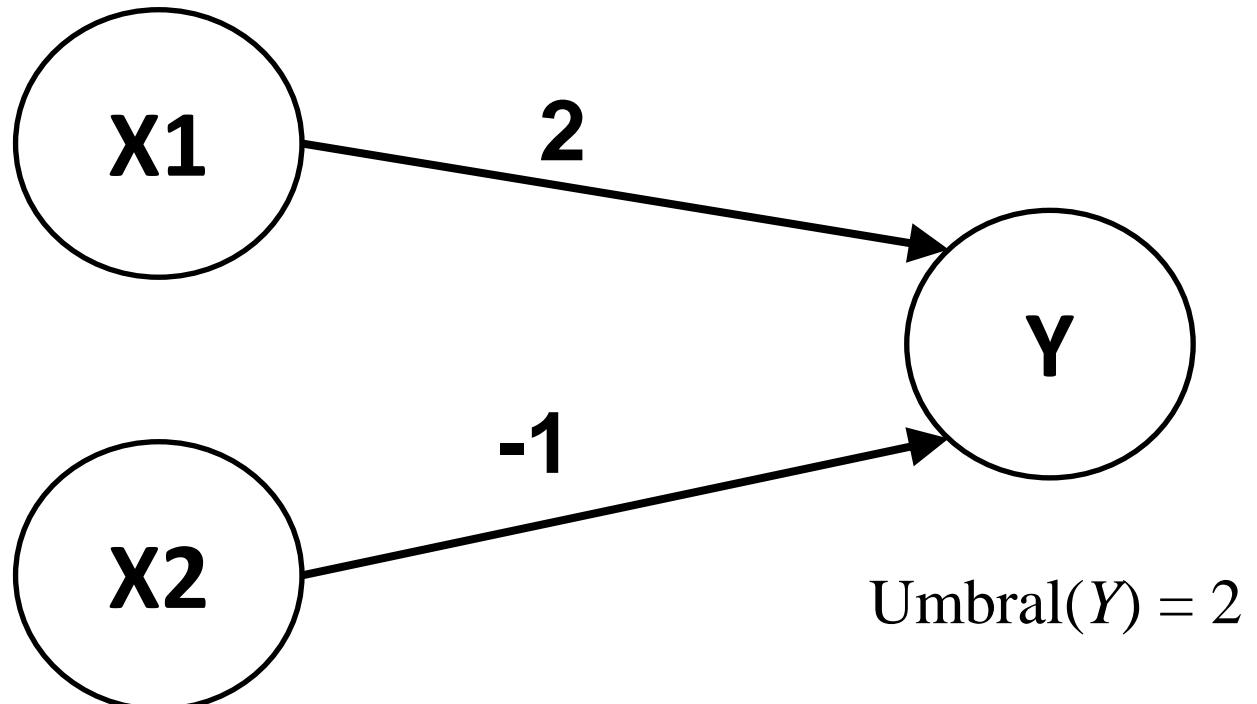
$x_1$	$x_2$	OR
1	1	1
1	0	1
0	1	1
0	0	0

# Red Neuronal Artificial Simple



$x_1$	$x_2$	AND
1	1	1
1	0	0
0	1	0
0	0	0

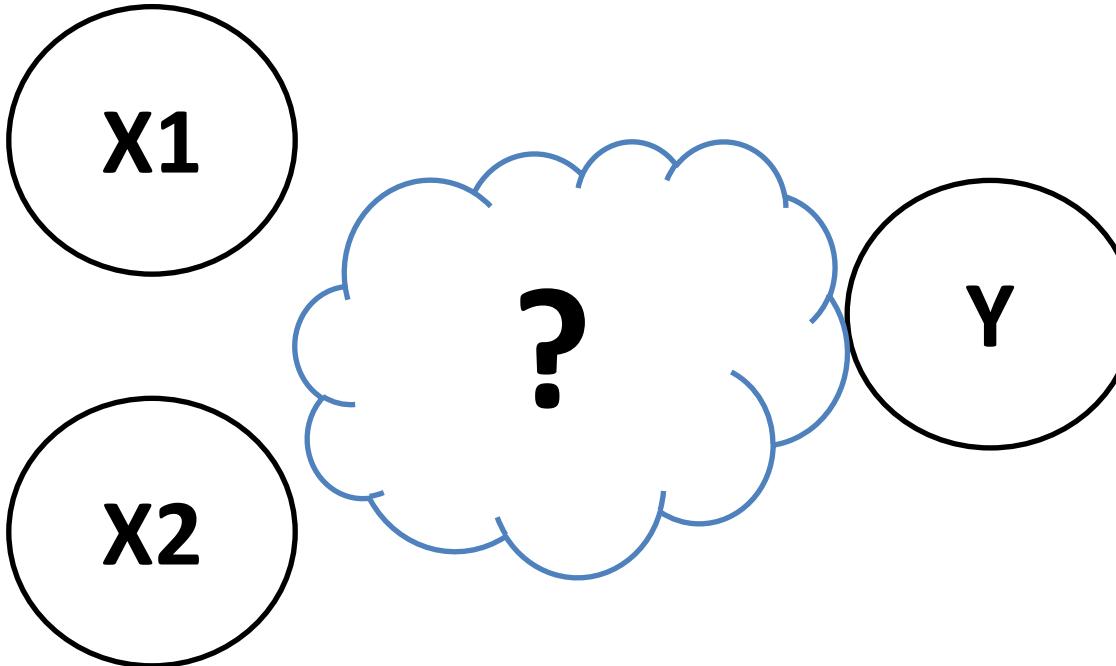
# Red Neuronal Artificial Simple



<b>X1</b>	<b>X2</b>	<b>AND NOT</b>
1	1	0
1	0	1
0	1	0
0	0	0

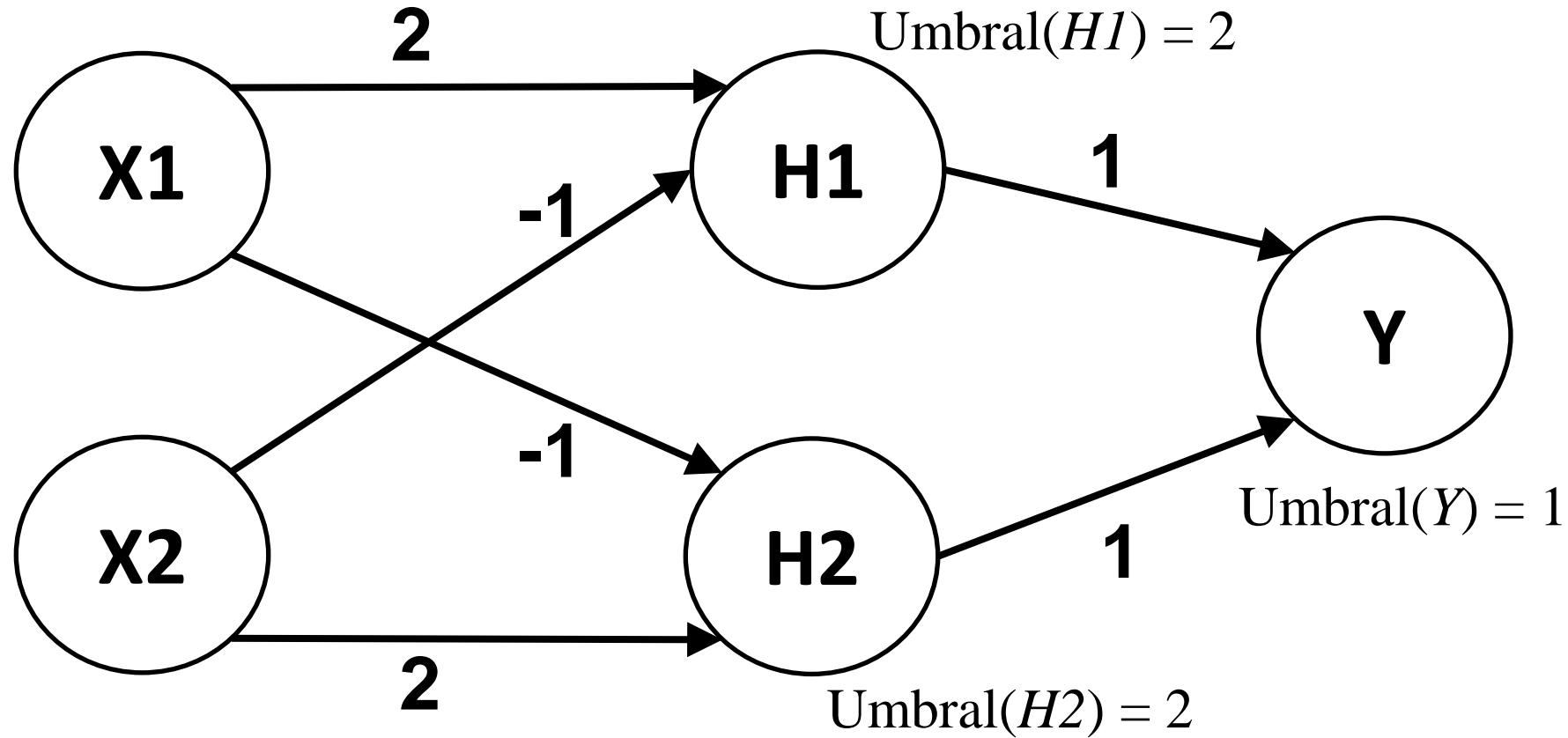
# Red Neuronal Artificial Simple

¿Cómo se implementa XOR?



<b>X1</b>	<b>X2</b>	<b>XOR</b>
1	1	0
1	0	1
0	1	1
0	0	0

# Red Neuronal Artificial Simple - XOR

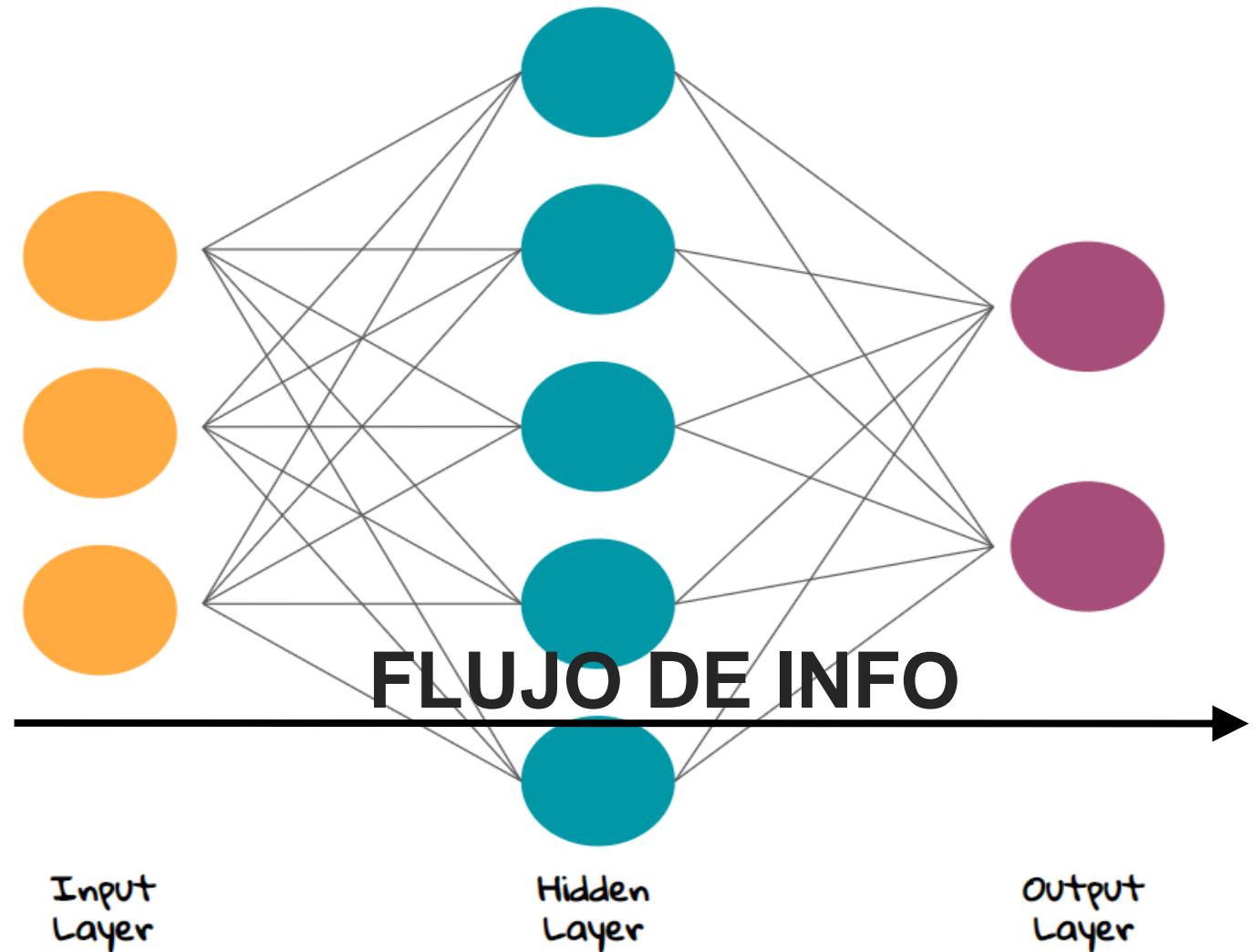


$$(X_1 \text{ AND NOT } X_2) \text{ OR } (X_2 \text{ AND NOT } X_1) = X_1 \text{ XOR } X_2$$

X1	X2	XOR
1	1	0
1	0	1
0	1	1
0	0	0

# Estructura y Flujo en Redes Neuronales

- Como entrada recibe datos en una dimensión pre-definida (*Input Layer* tiene una cantidad fija de nodos)
- Como salida, entrega una de N posibles clases. (*Output Layer* tiene una cantidad fija de nodos = clases)
- La capa interna o escondida (*Hidden Layer*), permite operaciones y relaciones más complejas. Al ser entrenada, conecta los datos de entrada con la clase que corresponda.
- Flujo de Información unidireccional
  - Los datos se presentan al *Input Layer*
  - Luego pasan al *Hidden Layer*
  - y Finalmente pasan al *Output Layer*



# Redes Neuronales – Ejemplo con Pixeles de Imagen

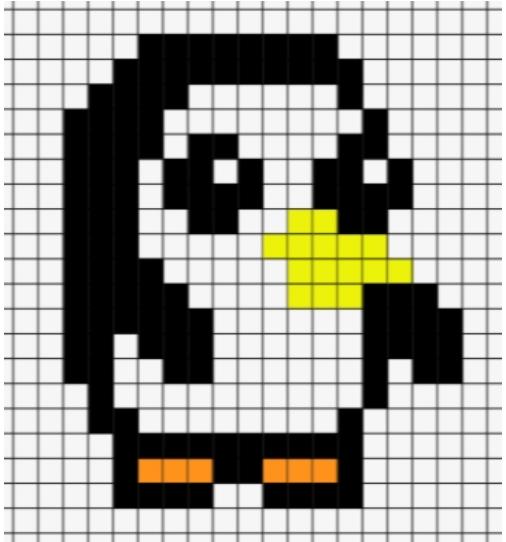
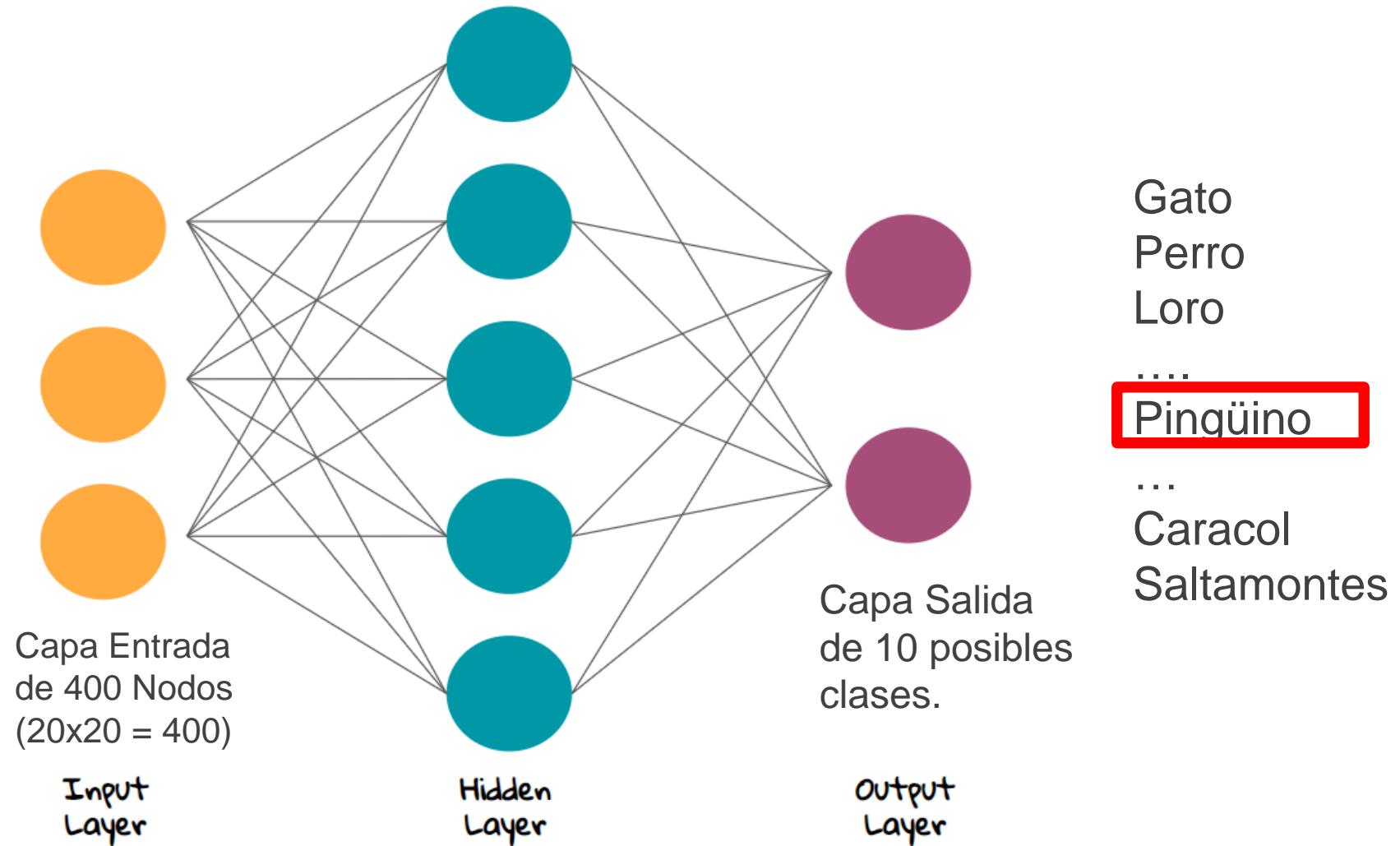
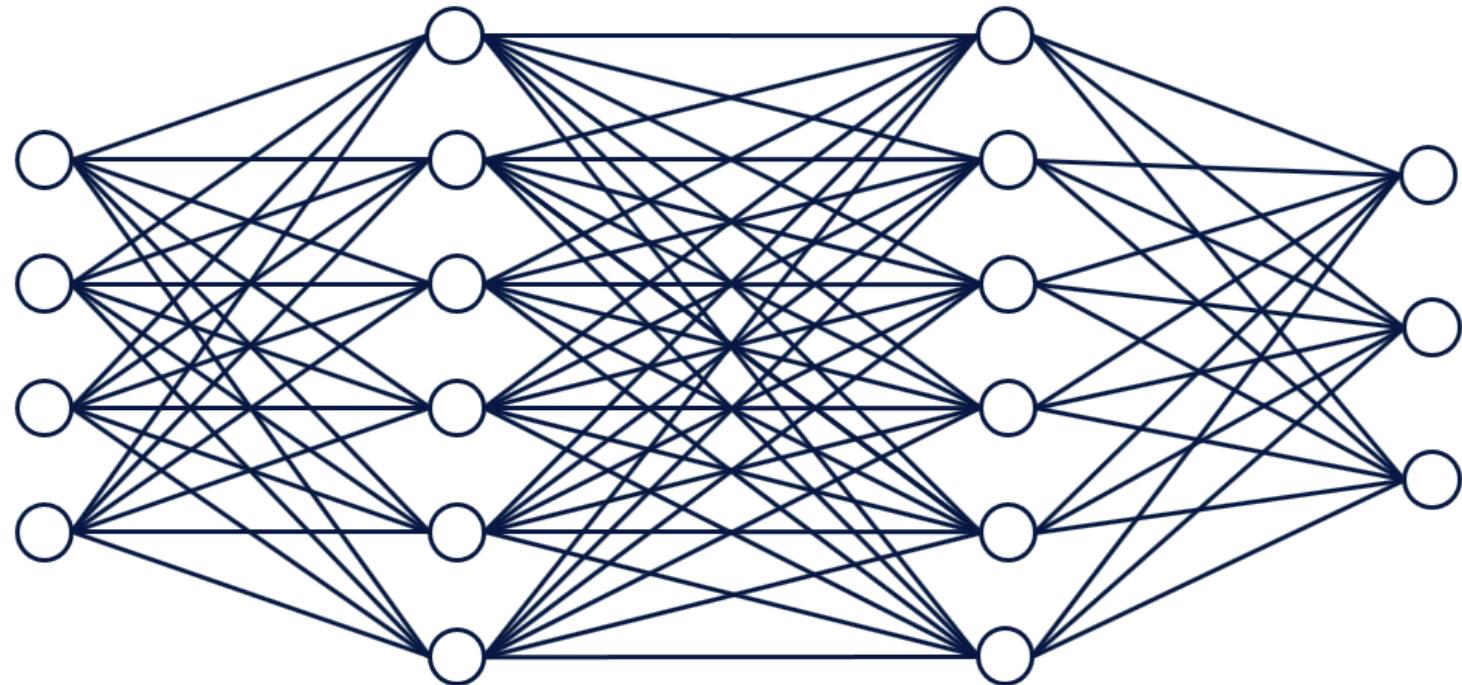


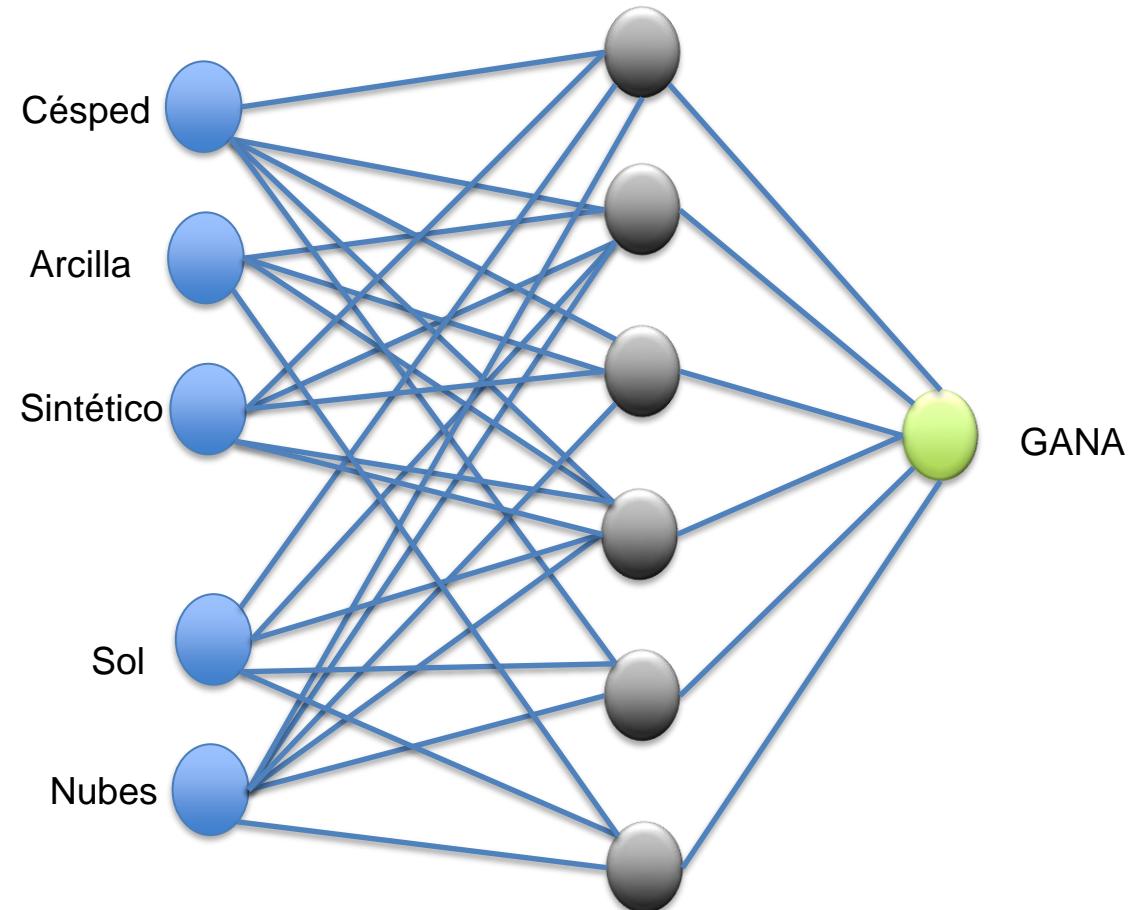
Imagen de 20x20



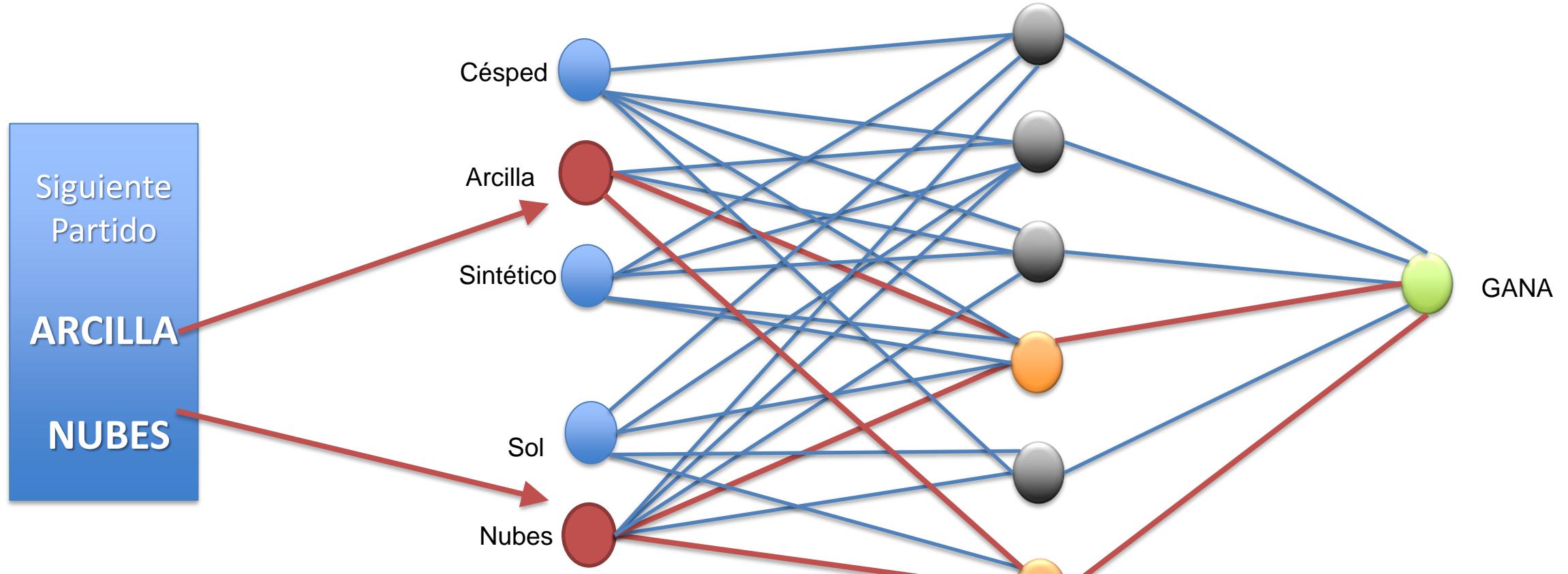


# Volviendo al tenista ...

# Partido	Tipo Cancha	Tiempo	Resultado
1	Arcilla	Soleado	GANÓ
2	Arcilla	Soleado	GANÓ
3	Césped	Nublado	PERDIÓ
4	Sintética	Nublado	GANÓ
5	Arcilla	Nublado	GANÓ
6	Sintética	Soleado	PERDIÓ
7	Arcilla	Nublado	GANÓ
8	Césped	Soleado	PERDIÓ
9	Arcilla	Soleado	PERDIÓ
10	Sintética	Nublado	GANÓ
11	Sintética	Soleado	GANÓ
12	Sintética	Nublado	GANÓ



# Volviendo al tenista ...



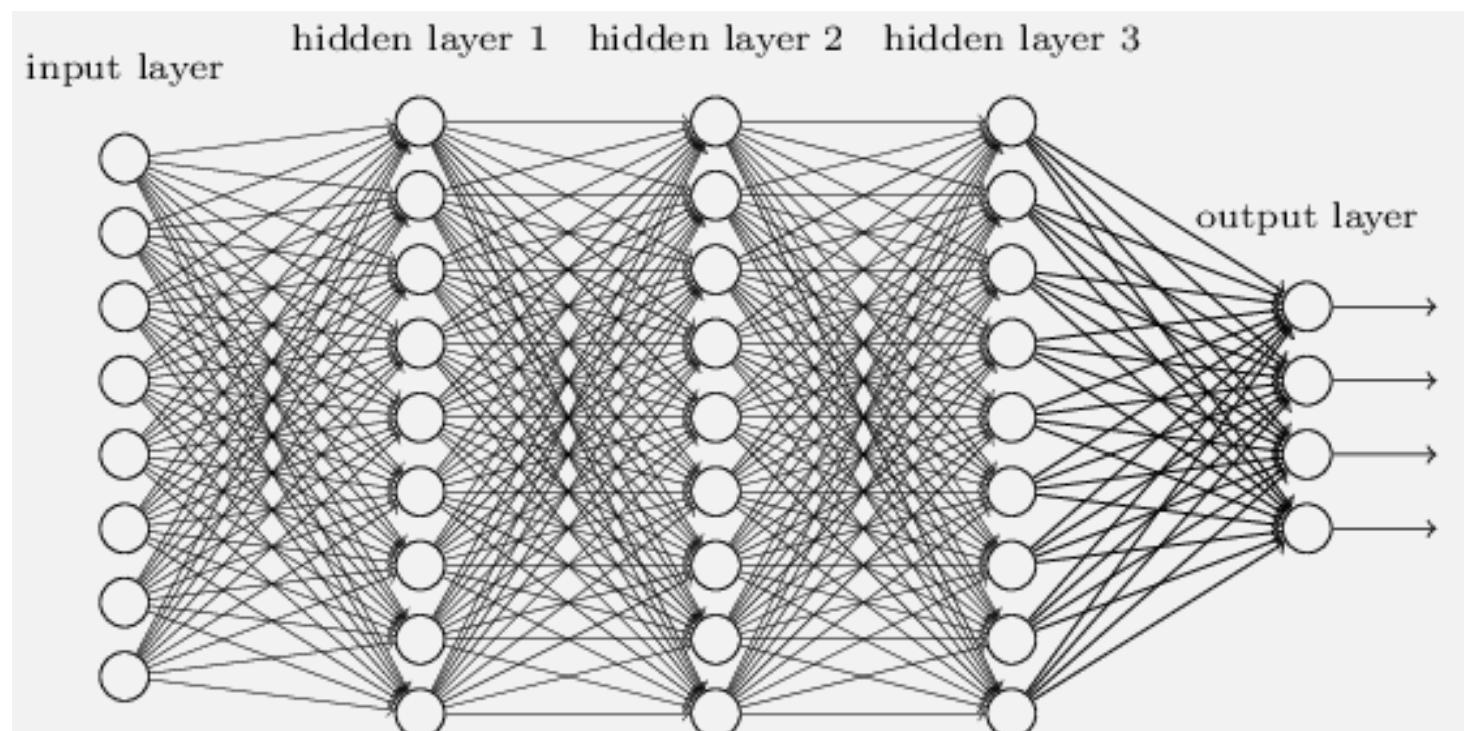
¿Por qué necesitamos redes neuronales profundas?

## 4.2. MOTIVACIÓN DEEP LEARNING

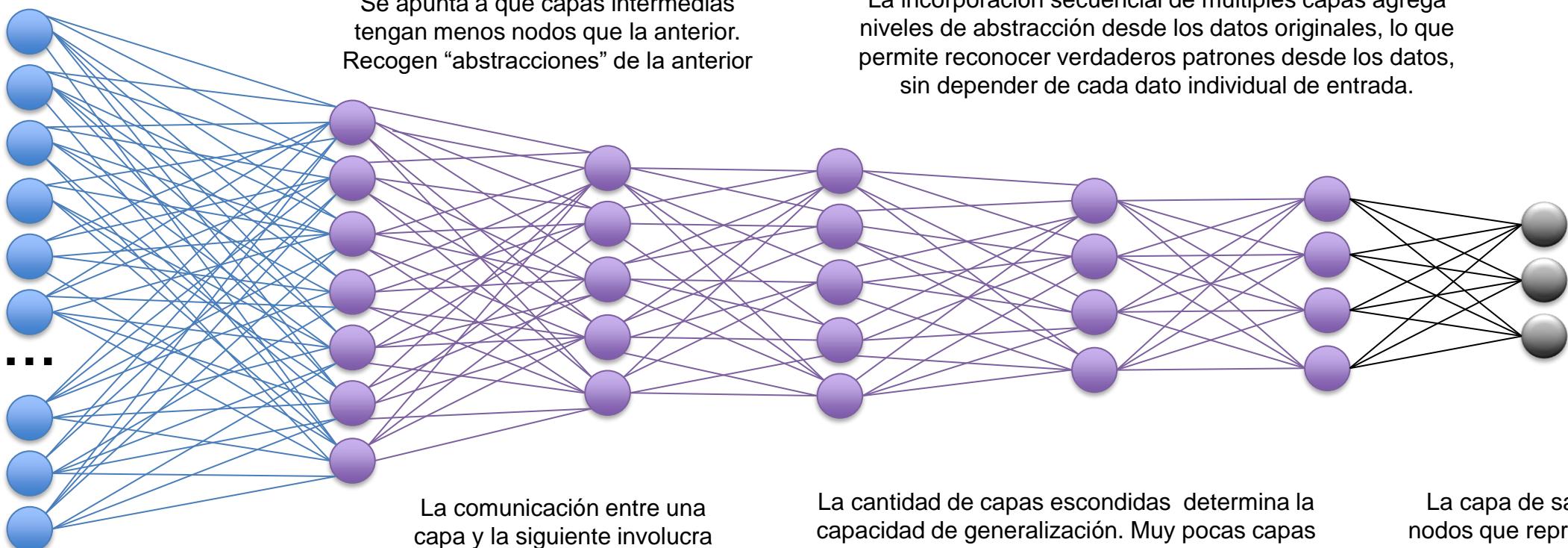
# ¿Por qué redes neuronales “profundas”?

En el contexto de la clasificación supervisada se ve que diferentes modelos tienen diversas ventajas y limitaciones, dependiendo del contexto del problema y de los datos. En general, las Redes Neuronales demuestran tener gran capacidad para resolver problemas de distintos tipos y contextos, con buen desempeño, especialmente gracias a la combinación de capas entrada-escondidas.

**Las redes neuronales profundas son aquellas que utilizan varias capas escondidas**, relacionando nodos de capas anteriores en forma agregada, para lograr – potencialmente – mejores resultados de predicción debido a un análisis de mejor generalización de los datos de entrada.



# Anatomía de una red neuronal profunda



La capa de entrada típicamente es de varios cientos, miles, e incluso millones de nodos. Estas redes tienen ventajas en contexto de datos no estructurados, como texto e imágenes.

Se apunta a que capas intermedias tengan menos nodos que la anterior. Recogen “abstracciones” de la anterior

La incorporación secuencial de múltiples capas agrega niveles de abstracción desde los datos originales, lo que permite reconocer verdaderos patrones desde los datos, sin depender de cada dato individual de entrada.

La comunicación entre una capa y la siguiente involucra funciones de activación no-lineales (**sigmoide, tanh, ReLU**, entre otras).

En redes tan enormes, no todos los arcos son necesarios. Adicionalmente, podrían tender a producir *overfitting*, por lo que hay técnicas de regularización eliminan nodos y conexiones, simplificando la red y mejorando su generalización.

La cantidad de capas escondidas determina la capacidad de generalización. Muy pocas capas limitan la posibilidad de interpretar, en forma más general, datos de enorme dimensión. Muchas capas pueden llevar a la “confusión” de la red, evitando que reconozca clases correctas.

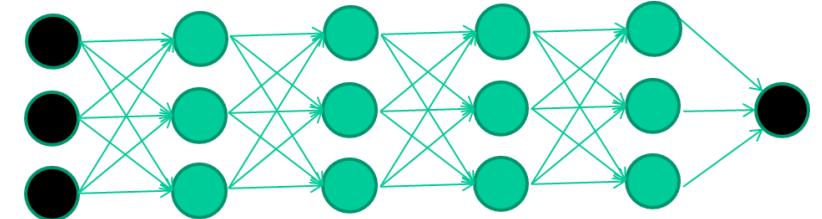
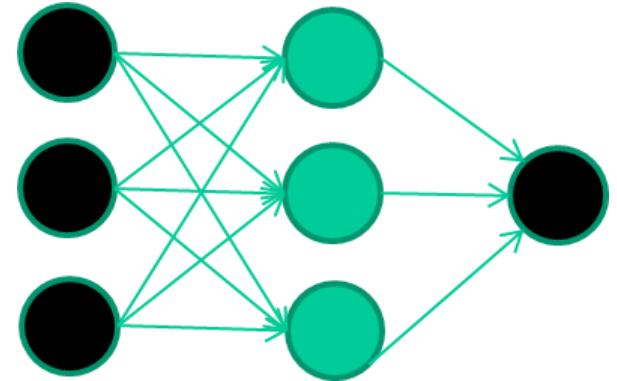
La capa de salida tiene nodos que representan las pocas clases a determinar. Incluso pueden ser cientos o miles, pero típicamente son una pequeña fracción de la cantidad de nodos de entrada.

# La ventaja de las capas intermedias o escondidas

- Estas capas, al conectarse entre los nodos de entrada y de salida, generan la identificación de características (*features identification/extraction*), que son patrones de información, que no siempre son evidentes en los datos de entrada y su relación con las clases de salida.
- Este reconocimiento de patrones ocurre en etapas, que es muy similar a la forma en que el cerebro humano realiza la identificación de características.
- Por ello, las redes neuronales profundas (*Deep Learning*) han demostrado capacidades sorprendentes en contextos de información extremadamente compleja (típicamente datos no-estructurados).
- Pero las redes neuronales multi-capa han existido por casi 3 décadas. **¿Dónde está la novedad actual?**

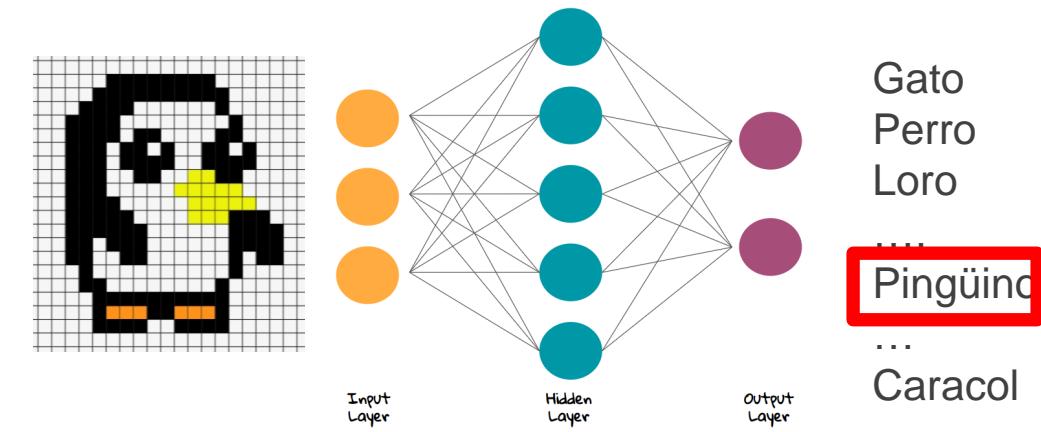
# \*El algoritmo de entrenamiento es la clave\*

- Entrenar una red neuronal simple (no-profunda) se reduce a probar diferentes combinaciones de pesos en las conexiones entre nodos, y de umbrales de activación de las neuronas, buscando calzar al máximo con los datos de entrenamiento.
- Pero los algoritmos tradicionales de “aprendizaje” para redes neuronales no son eficientes en el caso de una gran cantidad de pesos con muchas capas escondidas.
- **Backpropagation** es un algoritmo que permite entrenar redes neuronales con este nivel de complejidad, sin limitaciones.



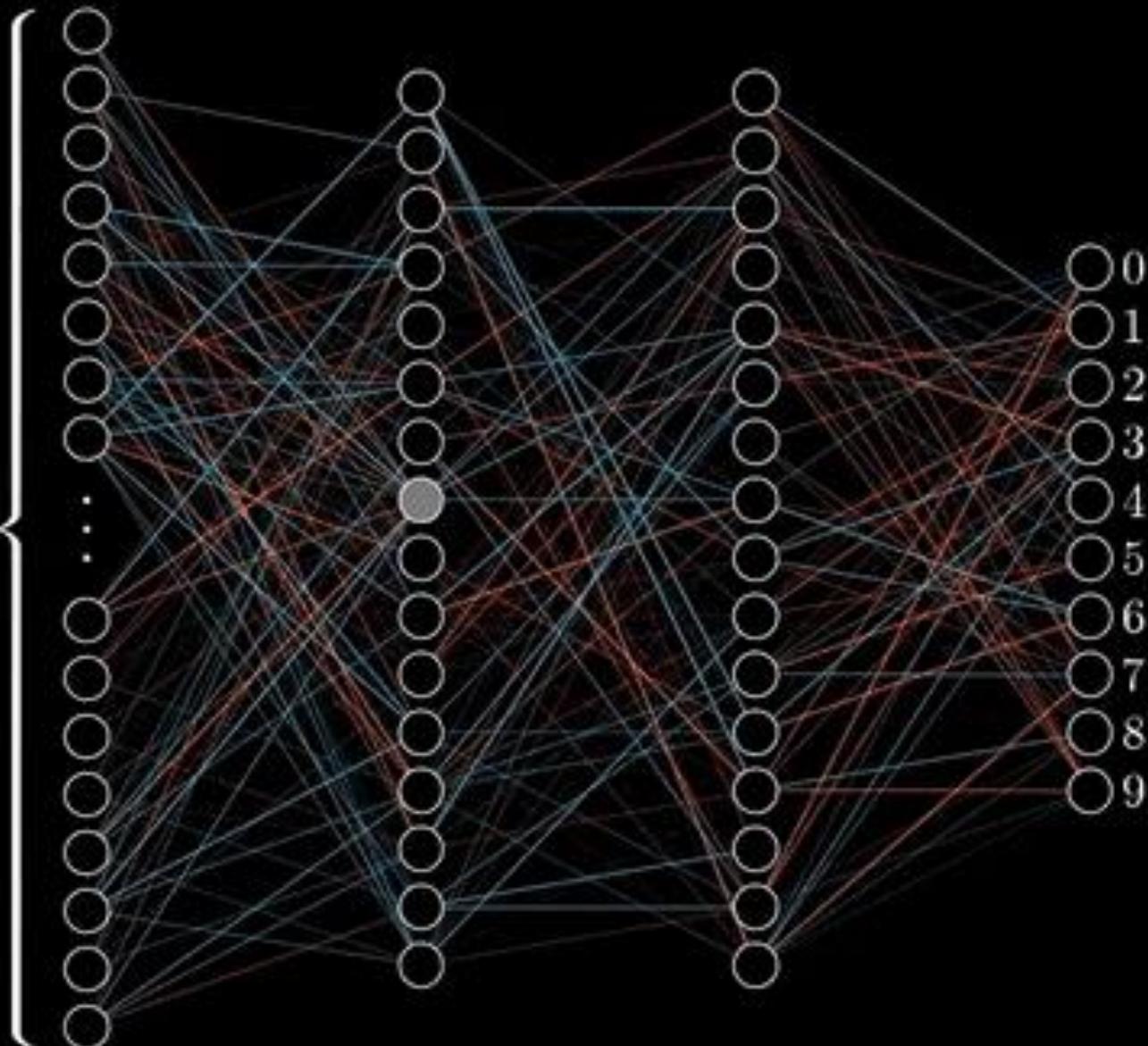
# “Aprendizaje no-supervisado de *features*”

- Por ej, en el análisis de imágenes, que son representadas por listas de píxeles, el reconocer que en la posición N hay un pixel negro, es un dato demasiado específico para ser relevante en un análisis generalizado de imágenes. Sin embargo, reconocer “cuerpos” con áreas blancas y otras negras, implica reconocer “features” en las imágenes que se pueden repetir en otras de la misma clase.
- De tal forma, las capas 2, 3, etc. van “sumando” áreas de píxeles para ir formando las características recurrentes en las imágenes de una clase particular, a veces independientemente de la posición en que estos píxeles aparecen en las imágenes.





784



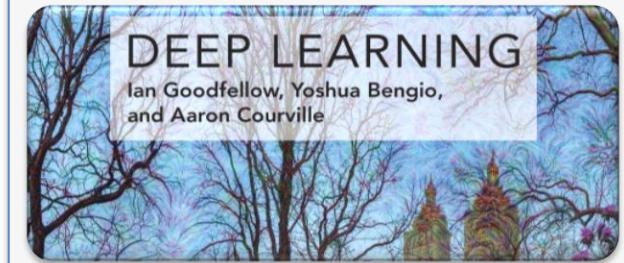
# Deep Feedforward Networks

**Deep Feedforward Networks  
Feedforward Neural Networks  
Multilayer Perceptrons (MLP)**

Objetivo: Aproximar una función  $f^*$ .

Ejemplo: un clasificador  $y = f^*(x)$  mapea una entrada  $x$  a una salida  $y$ . La DFN define un mapeo  $y = f(x; \Theta)$  y aprende el valor de los parámetros de  $\Theta$  que logran la mejor aproximación a la función.

**Feedforward** se debe a que el flujo va de  $x$  a  $y$ . (No hay feedback)  
**Network** implica una composición de funciones  
**Neuronal** por inspirada por la neurociencia.



## Deep Learning

- Ian Goodfellow, Yoshua Bengio, Aaron Courville
- 2015,  
[deeplearningbook.org](http://deeplearningbook.org)

# Redes Neuronales en *Deep Learning*

La estrategia está en “aprender” el mapeo (o función) de entrada-a-salida no-lineal.

Esta función de mapeo está parametrizada y un algoritmo de optimización encuentra los parámetros que corresponden a una buena representación para el conjunto de datos de entrenamiento.

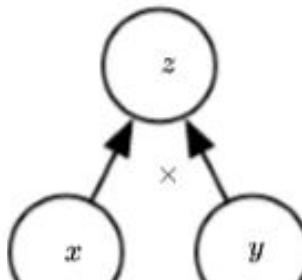
Un diseñador humano sólo necesita encontrar la familia de funciones general adecuada y no necesariamente la función específica más adecuada.

# Grafos computacionales

- Permiten entender más fácilmente la lógica, arquitectura de una red, y en particular, el algoritmo de back-propagation.
- Cada nodo el grafo representa una variable, la cual puede ser un escalar, vector, matriz, o tensor, o inclusive una variable de otro tipo.
- Una operación es una función simple de una o más variables.
- El grafo computacional define una serie acotada de operaciones permitidas.

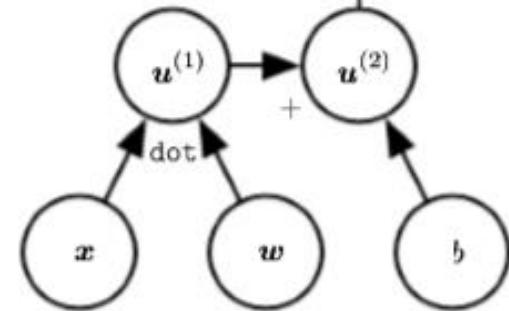
# Ejemplos

a) Grafo con la operación X para computar  $z = xy$



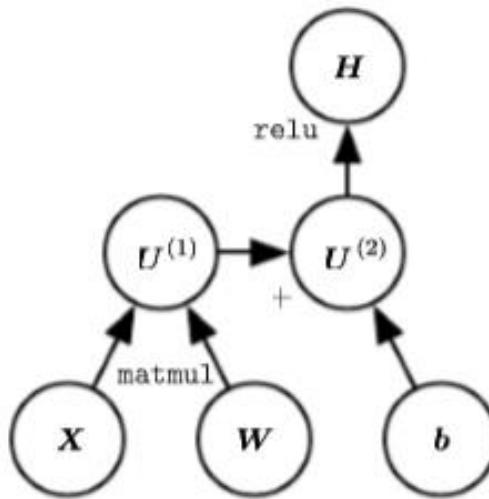
(a)

b) Grafo para la predicción de la regresión logística  $\hat{y} = \sigma(\mathbf{x}^\top \mathbf{w} + b)$  variables intermedias son  $u^{(n)}$



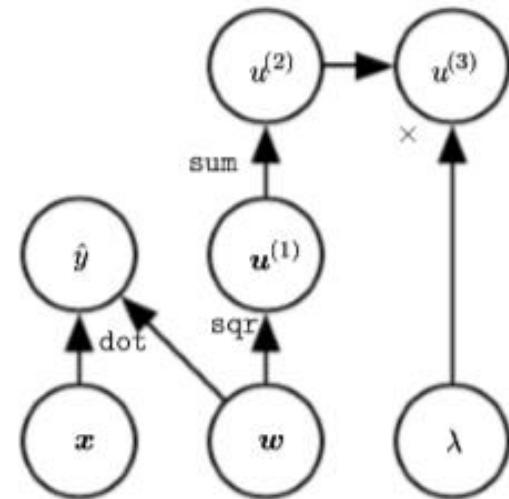
(b)

c) Grafo de la expresión  $H = \max\{0, \mathbf{X}\mathbf{W} + \mathbf{b}\}$  computando una matriz de diseño con ReLU  $H$



(c)

d) Grafo con más de una operación a los pesos  $w$  de una regresión lineal.



(d)

¿Por qué se ha vuelto tan popular en los últimos pocos años?

## 4.3. IMPACTO DE DEEP LEARNING

# The New York Times

## Scientists See Promise in Deep-Learning Programs

John Markoff

November 23, 2012

**Rich Rashid** in Tianjin, October, 25, 2012



La “**promesa de Deep Learning**” es destacada por New York Times en 2012

MIT  
Technology  
Review

# 10 BREAKTHROUGH TECHNOLOGIES 2013

[Introduction](#)[The 10 Technologies](#)[Past Years](#)

## Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

## Temporary Social Media

Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.

## Prenatal DNA Sequencing

Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?

## Additive Manufacturing

Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.

## Baxter: The Blue-Collar Robot

Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.

## Memory Implants

## Smart Watches

## Ultra-Efficient Solar Power

## Big Data from Cheap Phones

## Supergrids

MIT Technology Review, April 23<sup>rd</sup>, 2013

# Courant's LeCun to Lead Facebook's New Artificial Intelligence Group

December 9, 2013

143

Facebook has named New York University Professor Yann LeCun the director of a new laboratory devoted to research in artificial intelligence and deep learning.

"As one of the most respected thinkers in this field, Yann has done groundbreaking research in deep learning and computer vision," said Mike Schroepfer, Facebook's chief technology officer. "We're thrilled to welcome him to Facebook."

Facebook is building the team across three locations: Facebook's headquarters in Menlo Park, Calif., New York City, and London.

Machine learning is a branch of artificial intelligence that involves computers "learning" to extract knowledge from massive data sets and rendering informed analyses and judgments, often predicting outcomes.

LeCun, a professor at NYU's Courant Institute of Mathematical Sciences, is a pioneer in this growing field. In the 1980s, LeCun proposed one of the early versions of the back-propagation algorithm, the most popular method for training artificial neural networks. In the late 1980s and early 1990s at AT&T Bell Laboratories, he developed the convolutional network model—a pattern-recognition model whose architecture mimics, in part, the visual cortex of animals and humans. AT&T eventually deployed a check-reading system—based on this breakthrough—that by the late 1990s was reading about 20 percent of all the checks written in the U.S.



ENTERPRISE | [research](#) | uncategorized

**Facebook Taps 'Deep Learning' Giant for New AI Lab**

BY CADE METZ 12.09.13 3:14 PM

[Follow @cademetz](#)



Yann LeCun. Photo: Josh Valcarcel/WIRED

Facebook is building a research lab dedicated to the new breed of artificial intelligence, after hiring one of the preeminent researchers in the field: New York University professor Yann LeCun.

## NYU "Deep Learning" Professor LeCun Will Head Facebook's New Artificial Intelligence Lab, Dec 10, 2013

# Google Acquires Artificial Intelligence Startup DeepMind For More Than \$500M

Catherine Shu @catherineshu / 10:20 pm -03 • January 26, 2014

Comment



Google will buy London-based artificial intelligence company DeepMind. The Information reports that the acquisition price was more than \$500 million, and that Facebook was also in talks to buy the startup late last year. DeepMind confirmed the acquisition to us, but couldn't disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.

Google's hiring of DeepMind will help it compete against other major tech companies as they all try to gain business advantages by focusing on deep learning. For example, Facebook recently hired NYU professor Yann LeCunn to lead its new artificial intelligence lab, IBM's Watson supercomputer is now working on deep learning, and Yahoo recently acquired photo analysis startup LookFlow to lead its new deep learning group.

DeepMind was founded by neuroscientist Demis Hassabis, a former child prodigy in chess, Shane Legg, and Mustafa Suleyman. Skype and Kazaa developer Jaan Tallin is an investor.

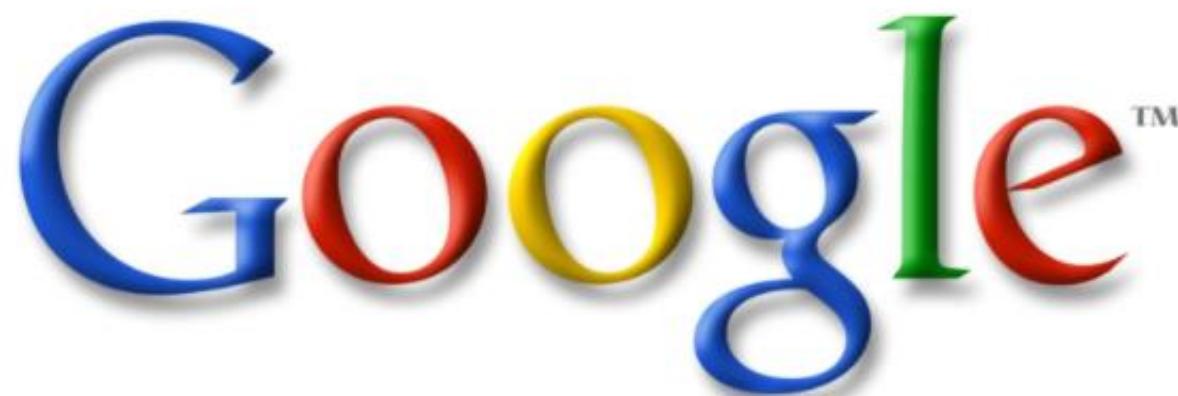
This is the latest move by Google to fill out its roster of artificial intelligence experts and, according to Re/code, the acquisition was reportedly led by Google CEO Larry Page. If all three of DeepMind's founders work for Google, they will join inventor, entrepreneur, author, and futurist Ray Kurzweil, who was hired in 2012 as a director of engineering focused on machine learning and language processing.



# Google Scoops Up Neural Networks Startup DNNresearch To Boost Its Voice And Image Search Tech

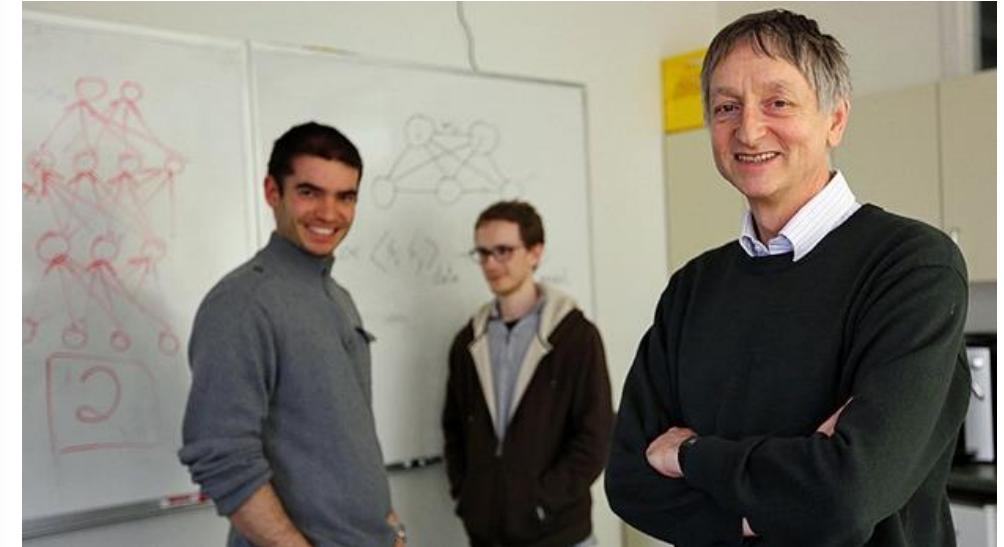
Rip Empson @rlpemp / 8:37 pm -03 • March 12, 2013

Comment



Well, Google's M&A strategy is nothing if not diverse in focus. In November, it acquired package delivery startup Bufferbox. Last month, Google it made its first acquisition of the year, buying eCommerce startup Channel Intelligence. Today, Google dug into the Computer Science department at The University of Toronto to acquire DNNresearch, a young startup founded by professor Geoffrey Hinton and two of his grad students, Alex Krizhevsky and Ilya Sutskever.

Incorporated last year, the startup's website is conspicuously devoid of any identifying information — just a blank black screen. While the financial terms of the deal were not disclosed, Google was eager to acquire the startup's research on neural networks — as well as the talent behind it — to help it go beyond traditional search algorithms in its ability to identify pieces of content, images, voice, text and so on. In its announcement today, the



**DNNResearch: startup de 3 personas (incluyendo Geoff Hinton) adquirida por Google.**

# The Great A.I. Awakening

How Google used artificial intelligence to transform Google Translate, one of its more popular services — and how machine learning is poised to reinvent computing itself.

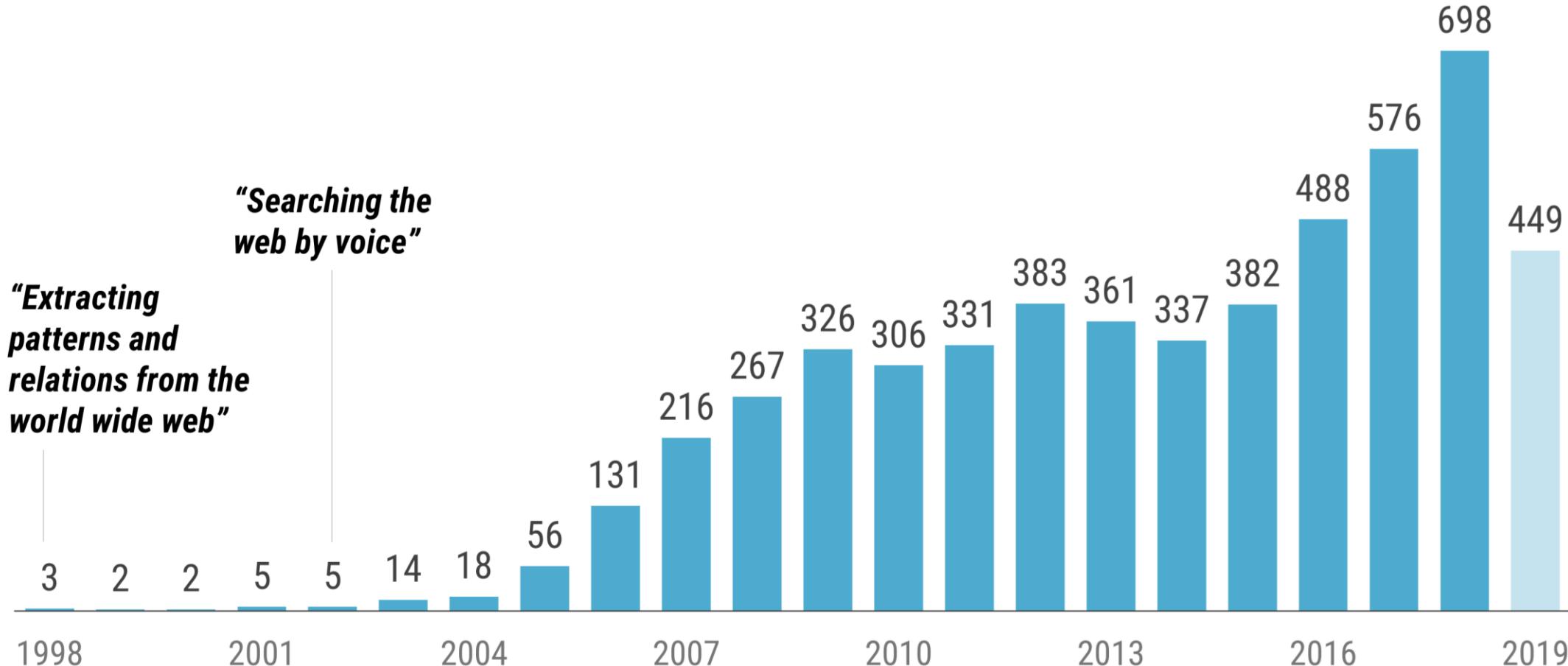
BY GIDEON LEWIS-KRAUS DEC. 14, 2016





# Google's AI research dates back to its founding year

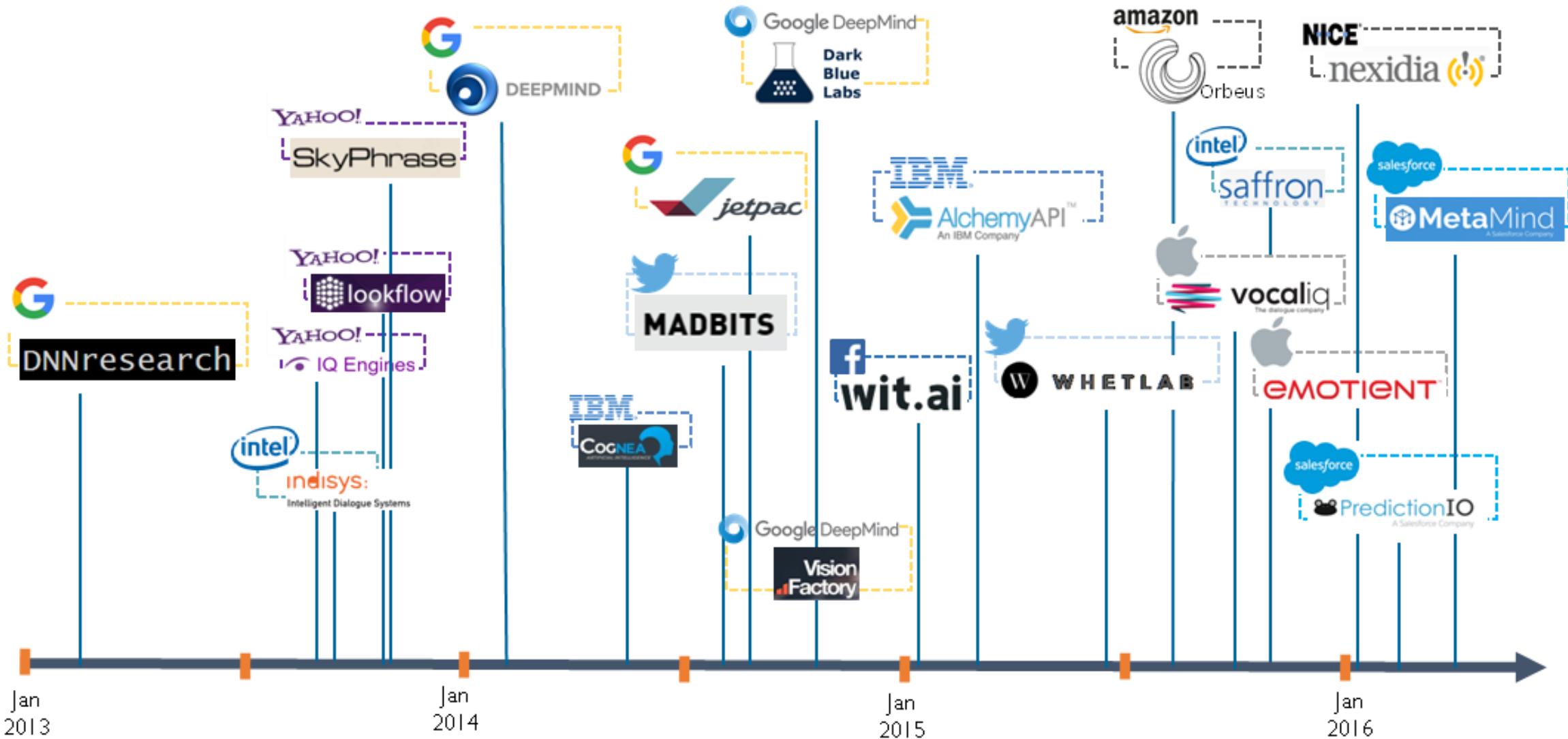
AI-related publications, 1998 – 2019 (as of Aug. 5)



Source: Google AI

CB INSIGHTS

# Race To AI: Major Acquisitions In Artificial Intelligence



# Deep Learning – De la Investigación a la Aplicación

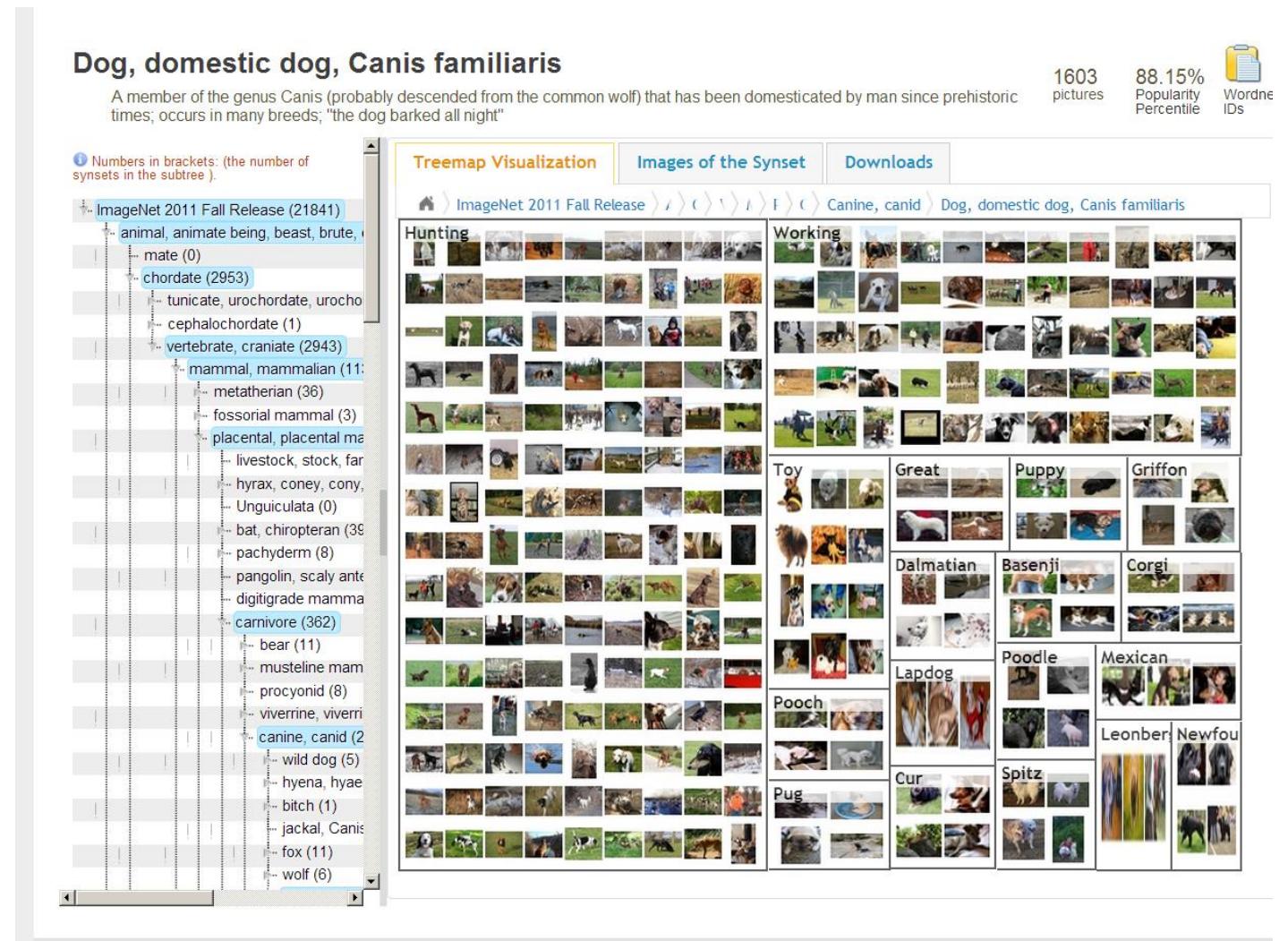


Deep Learning

Particularmente relevante en visión computacional y reconocimiento de voz

# Caso de éxito en CNN: ILSVRC 2012

- ImageNet Large Scale Visual Recognition Competition
  - BD con 14 millones de imágenes etiquetadas
  - Más de 20.000 categorías



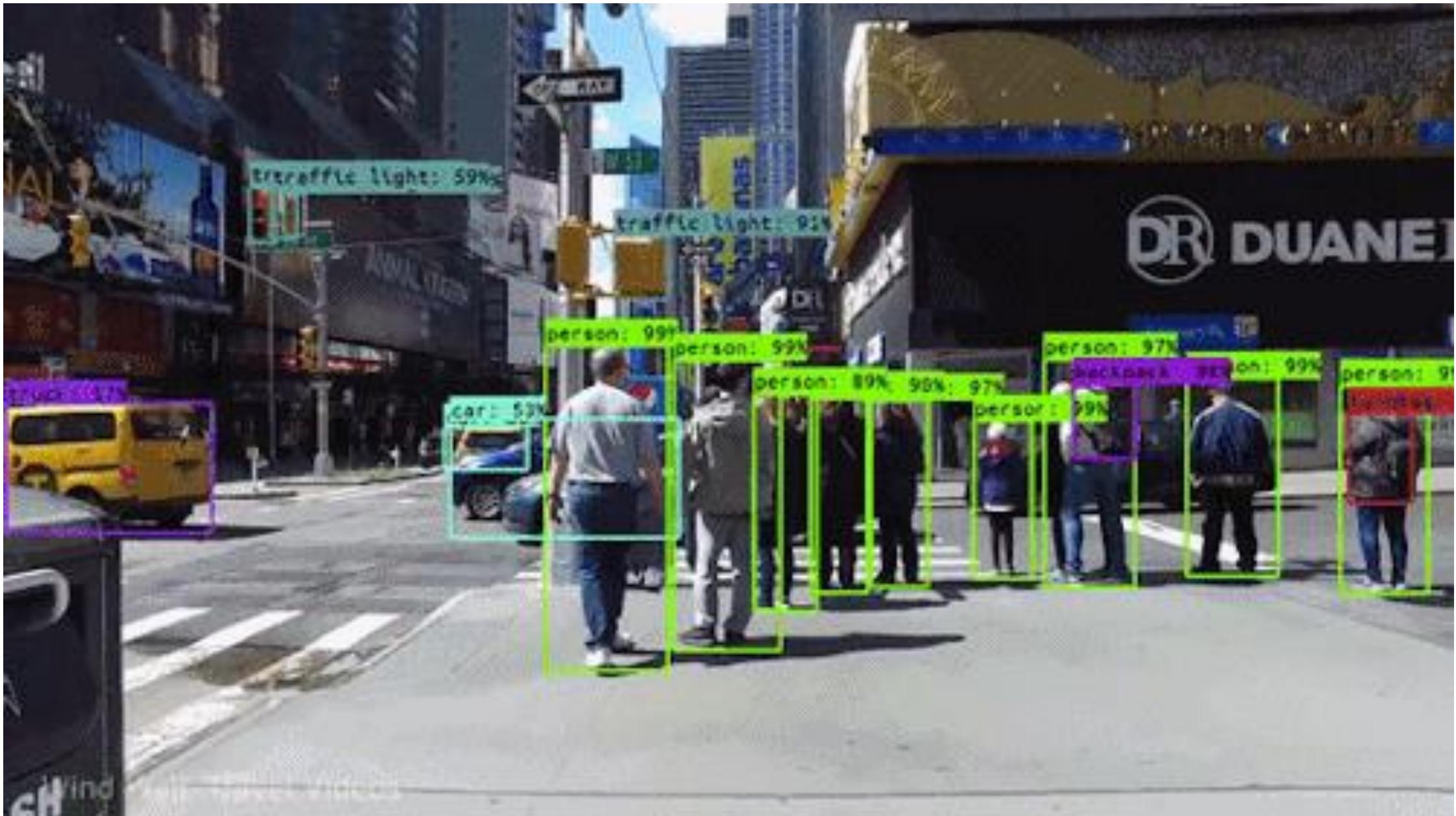
# Los mejores reconocedores de imágenes

2012

N	Error-5	Modelo	Equipo	Autores
1	0.153	<b>Deep Convolutional Neural Network</b>	Univ. of Toronto	Krizhevsky et al
2	0.262	Features + Fisher Vectors + Linear classifier	ISI	Gunji et al
3	0.270	Features + FV + SVM	OXFORD_VGG	Simonyan et al
4	0.271	SIFT + FV + PQ + SVM	XRCE/INRIA	Perronnin et al
5	0.300	Color desc. + SVM	U. Amsterdam	van de Sande et al

2013

N	Error-5	Algoritmo	Equipo	Autores
1	0.117	<b>Deep Convolutional Neural Network</b>	Clarifi	Zeiler
2	0.129	<b>Deep Convolutional Neural Networks</b>	Nat. Univ. Singapore	Min LIN
3	0.135	<b>Deep Convolutional Neural Networks</b>	NYU	Zeiler Fergus
4	0.135	<b>Deep Convolutional Neural Networks</b>		Andrew Howard
5	0.137	<b>Deep Convolutional Neural Networks</b>	Overfeat NYU	Pierre Sermanet et al



## **4.4. LOS CONCEPTOS BÁSICOS EN DEEP LEARNING**

Fuente parcial: Mingyue Tan, UBC

# ¿Entonces, qué cambia de las ANN simples?

El tratamiento de la información en nodos

- Procesamiento de información en forma de matrices (**álgebra lineal**)

Función de Activación

- Lineal, Step, No-Lineal, Sigmoide, Tan Hiperbólica, ReLU.

Entrenamiento (aprendizaje) de la red

- Aprendizaje basado en gradiente: Optimización, Función de Costo, Unidades de Salida
- Algoritmos de diferenciación: Back-propagation y otros

Regularización

- Reduciendo Overfitting con Drop-out, L1, L2, etc.

Procesamiento (entrenamiento) en la nube

- La gran demanda de recursos que implica entrenar redes neuronales profundas se resuelve gracias a servicios en la nube

## **4.4.1 CONCEPTOS BÁSICOS DE ÁLGEBRA LINEAL**

Fuente parcial: Mingyue Tan, UBC

# Álgebra Lineal

Rama de las matemáticas ampliamente usada en ciencia e ingeniería

Una adecuada comprensión de AL es esencial para entender y trabajar con algunos algoritmos de Machine Learning, especialmente en Deep Learning

Referencia complementaria:

“The Matrix Cookbook, Petersen and Pedersen, 2006.

# Objetos de Álgebra Lineal

## Escalares

- Simplemente un número
- Típicamente se declaran como “sea  $s \in \mathbb{R}$  la pendiente de la recta” o “sea  $n \in \mathbb{N}$  la cantidad de unidades”.

## Vectores

- Arreglos o listas ordenadas de elementos/números
- Cada elemento se puede identificar por su índice
- Se pueden declarar explícitamente, como  $\{1, 2, 3\}$  o bien “sea  $X \in \mathbb{R}^N$ ”

$$X = [x_1, x_2, \dots, x_n]$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

# Objetos de Álgebra Lineal

## Matrices

- Arreglo 2D de números. Cada elemento se identifica por 2 índices, como  $A_{1,1}$
- Una matriz de  $m$  filas y  $n$  columnas, se declara como “sea  $A$  s  $\in \mathbb{R}^{m \times n}$ ”

## Tensores

- Son matrices de más de 2 ejes. En general se definen como un arreglo de números en una grilla o tabla regular, con un número variable de ejes.
- Por ej, si son 3 ejes, el tensor se puede declarar como  $A_{i,j,k}$

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow A^\top = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

La transpuesta de una matriz se puede declarar como la imagen refleja sobre la diagonal principal

La transpuesta se puede declarar como:

$$(A^\top)_{i,j} = A_{j,i}.$$

Se puede ver un vector como una matriz de una sola columna. Por eso, su Transpuesta es una matriz de una sola fila.

Un escalar puede verse como una matriz de una sola entrada. Por lo mismo, su transpuesta es el mismo escalar.

# Multiplicación de Matrices y Vectores

- Dos matrices A y B multiplicadas, deben cumplir con:
  - A debe tener la misma cantidad de columnas que filas de B
  - Su resultado es una matriz C de las filas de A y columnas de B

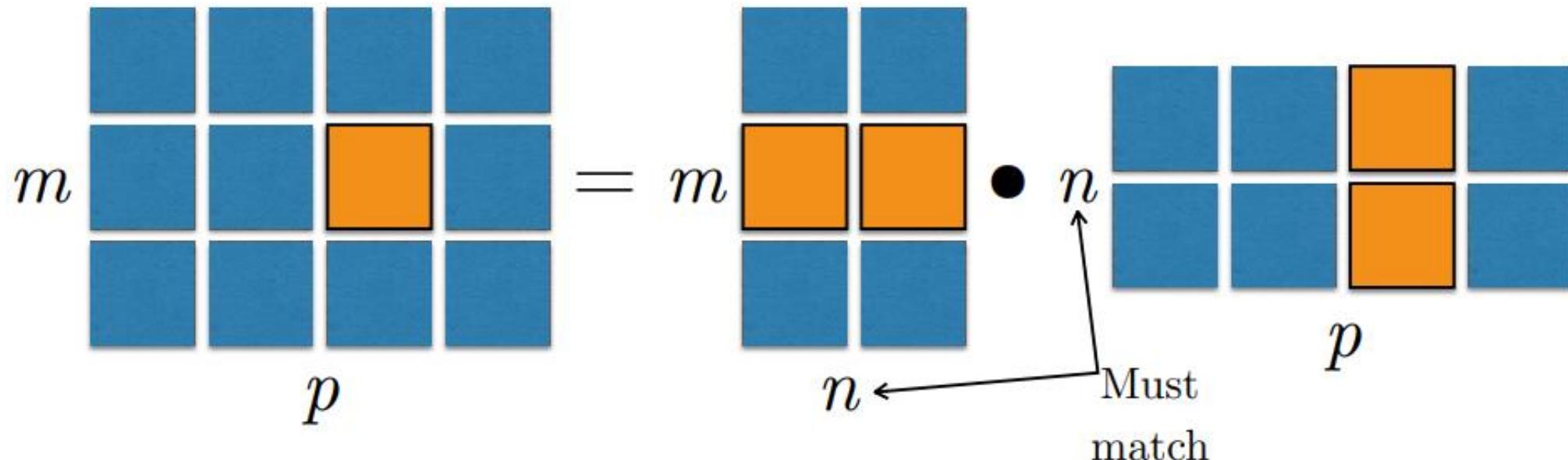
$$C = AB \quad \text{definida por} \quad C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

- La multiplicación es distributiva  $A(B + C) = AB + AC.$
- Y también asociativa  $A(BC) = (AB)C.$
- ... pero no comutativa.

# Multiplicación de Matrices

$$C = AB. \quad (2.4)$$

$$C_{i,j} = \sum_k A_{i,k}B_{k,j}. \quad (2.5)$$



## **4.4.2. FUNCIONES DE ACTIVACIÓN**

Fuente parcial: **Mingyue Tan, UBC**

# Función de Activación o Función de Transferencia

Objetivo: introducir propiedades no-lineales a una red.

Calcula la suma ponderada y agrega dirección, decidiendo si “excitar” una neurona en particular o no.

El propósito fundamental es convertir una señal de entrada de un nodo en una Red Neuronal en una señal de salida. Esa señal de salida se usa como entrada en una siguiente capa.

La función de activación no-lineal ayudará al modelo a entender la complejidad y obtener resultados precisos.

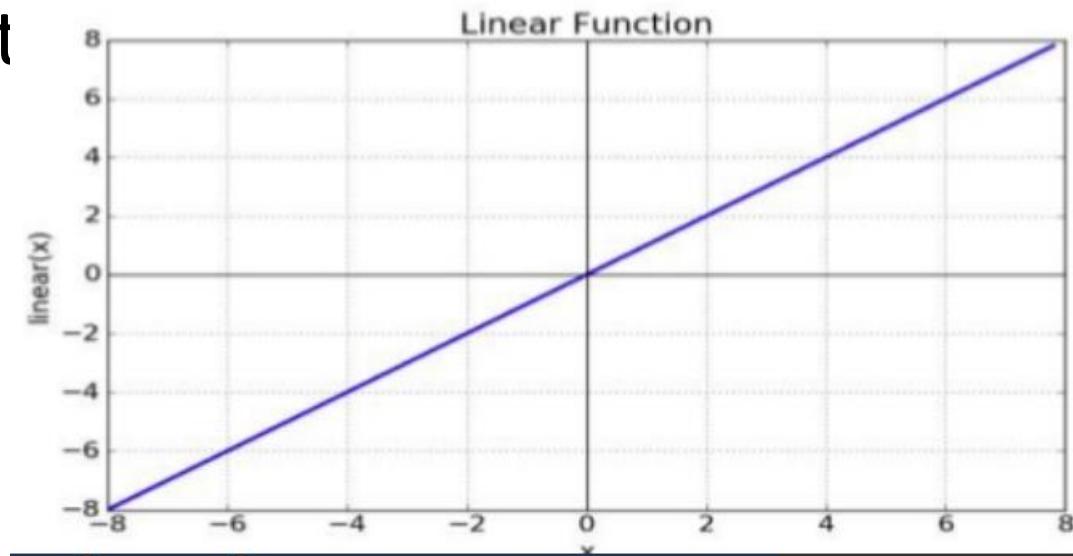
Provoca que la salida no sea una simple función lineal (la ANN sería una regresión lineal). Es una función polinomial de grado 1.

Gracias a esta función abre el mundo a problemas de dimensiones complejas, como imágenes, video, texto, audio, etc.

Referencia: Astha Jain

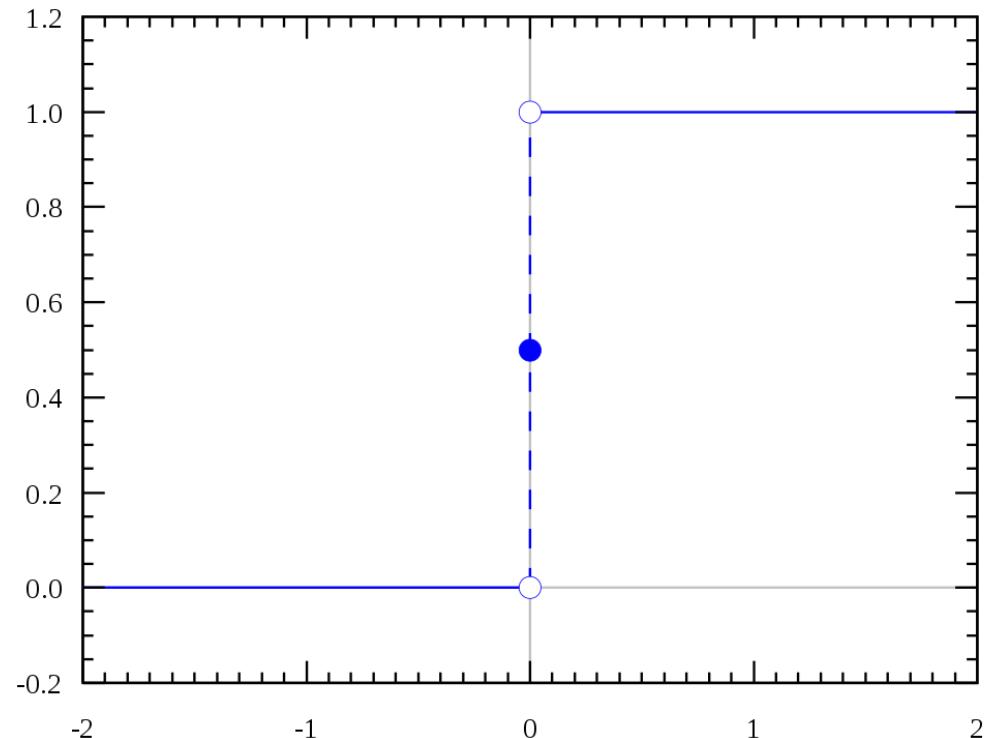
# Función de activación lineal

- A ser una línea, la salida no queda restringida a un rango de valores.
  - Ecuación:  $f(x) = x$
  - Rango:  $[-\infty, \infty]$
- Por su simplicidad, no siempre es útil para dimensiones más complejas.



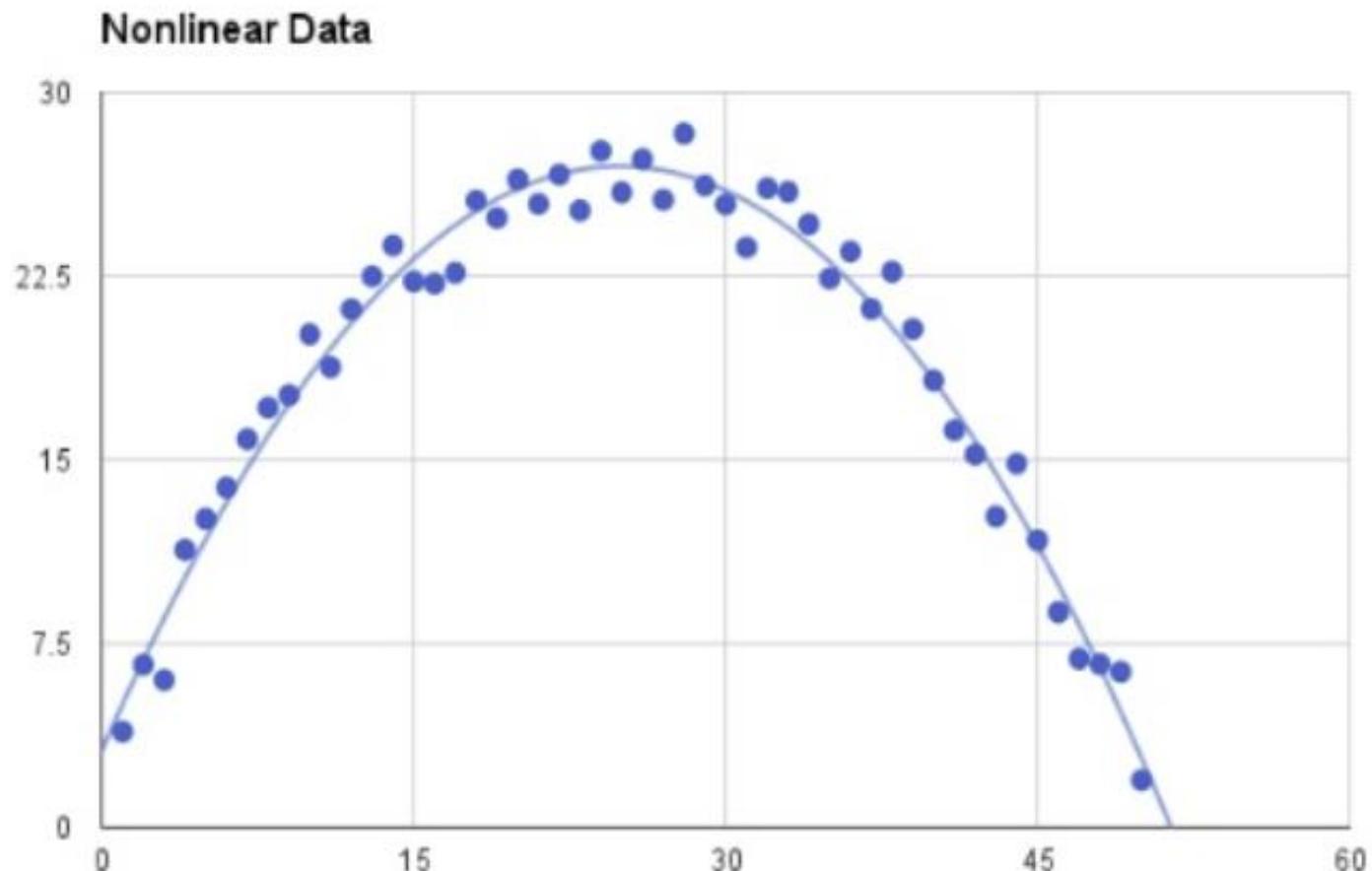
# Función Escalón o *Step*

- Es una de las funciones de activación más simples.
- Se considera un valor umbral y si el valor que viene en la entrada de la red es mayor que ese umbral, la neurona se activa.
- $F(x) = 1, \text{ si } x \geq 0$



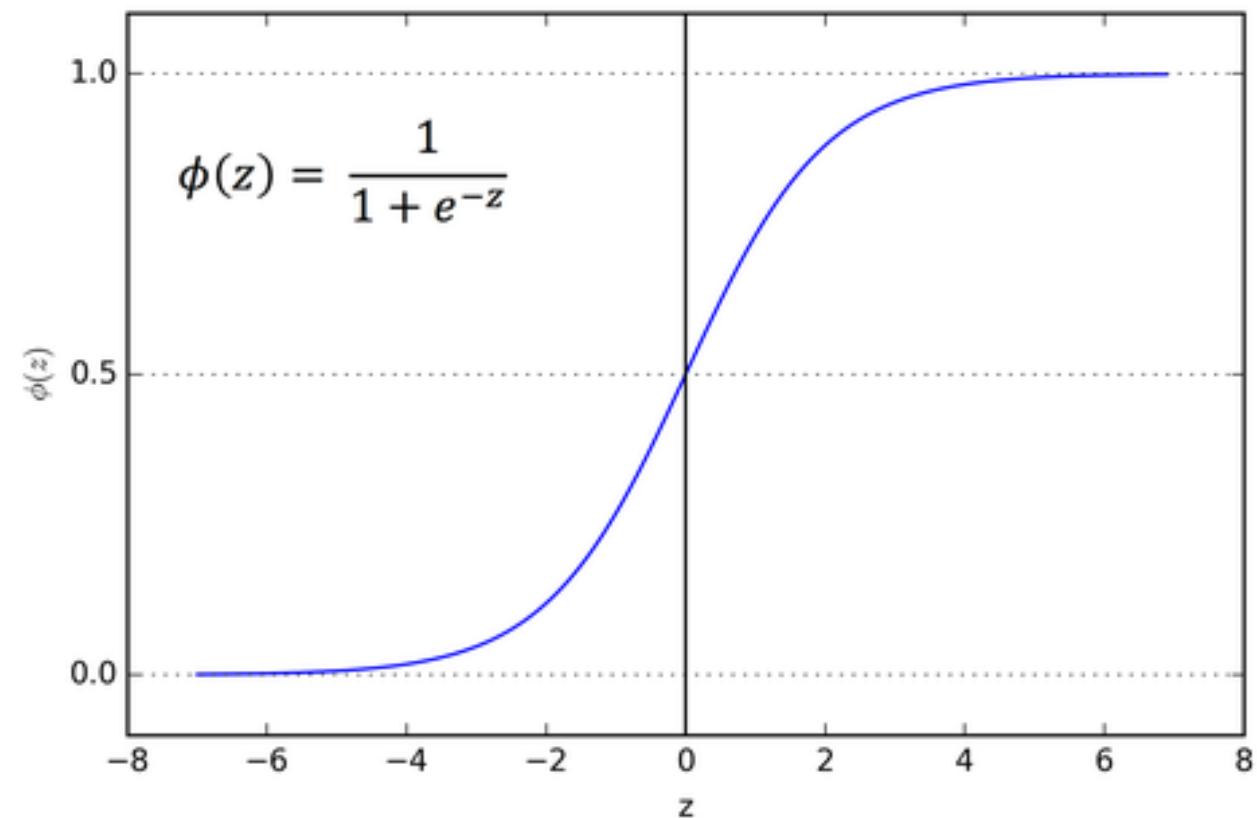
# Función no-lineal

- Permite que el modelo se adapte a datos más variados
- Derivativa o diferencial.
  - Cambios en el eje Y con respecto a cambios en el eje x. También conocida como pendiente (*slope*).
  - Puede encontrar la pendiente de la función sigmoide en 2 puntos cualquiera.
- Función monotónica.
  - Una función que es enteramente no-creciente o no-decreciente.



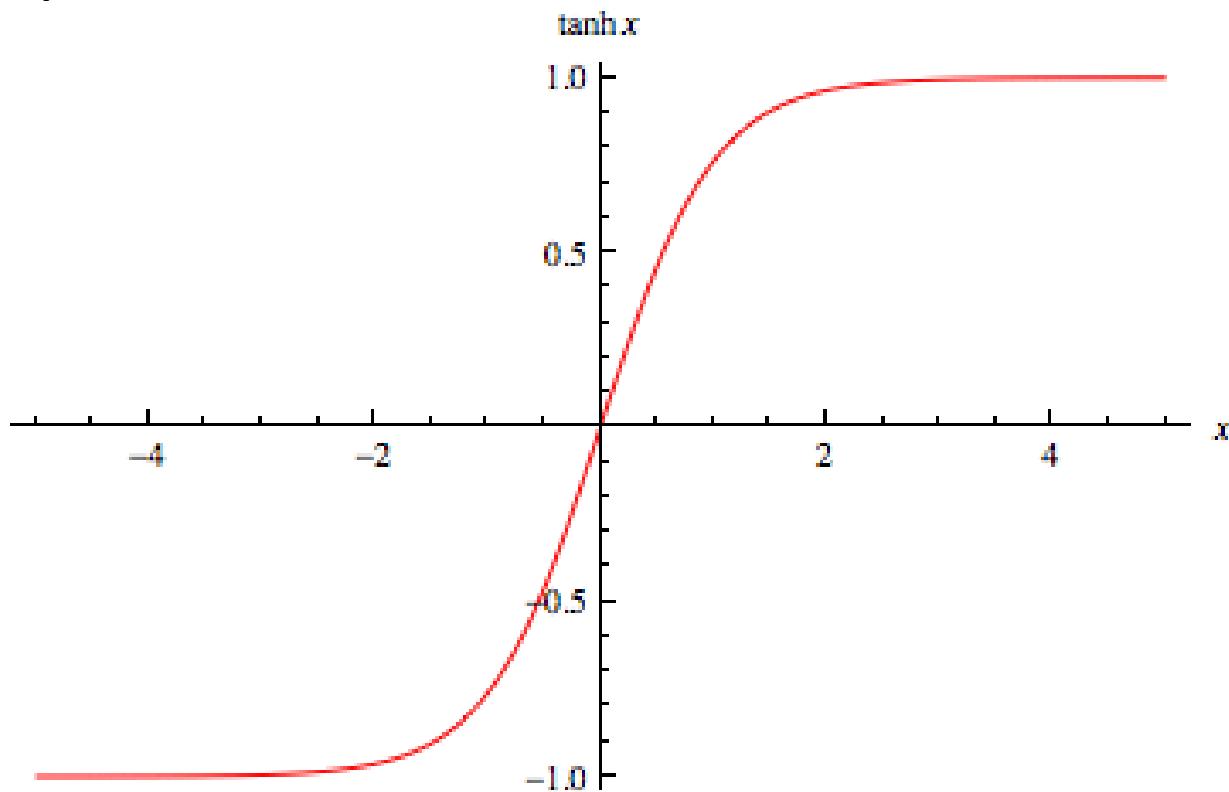
# Funciones de activación según su rango o curvas

- **Función Sísmoide o Logística**
- Su rango de valores resultantes es  $[0, 1]$
- Ventajas: funciona perfecto en el rango de valores, lo que permite, inclusive, escalar valores negativos grandes hacia 0, y los positivos grandes hacia 1.
- Desventajas: el problema de la gradiente que desaparece. Su salida no está centrada en 0, lo que hace que su gradiente se vaya muy lejos en diferentes direcciones, haciendo la optimización más difícil.



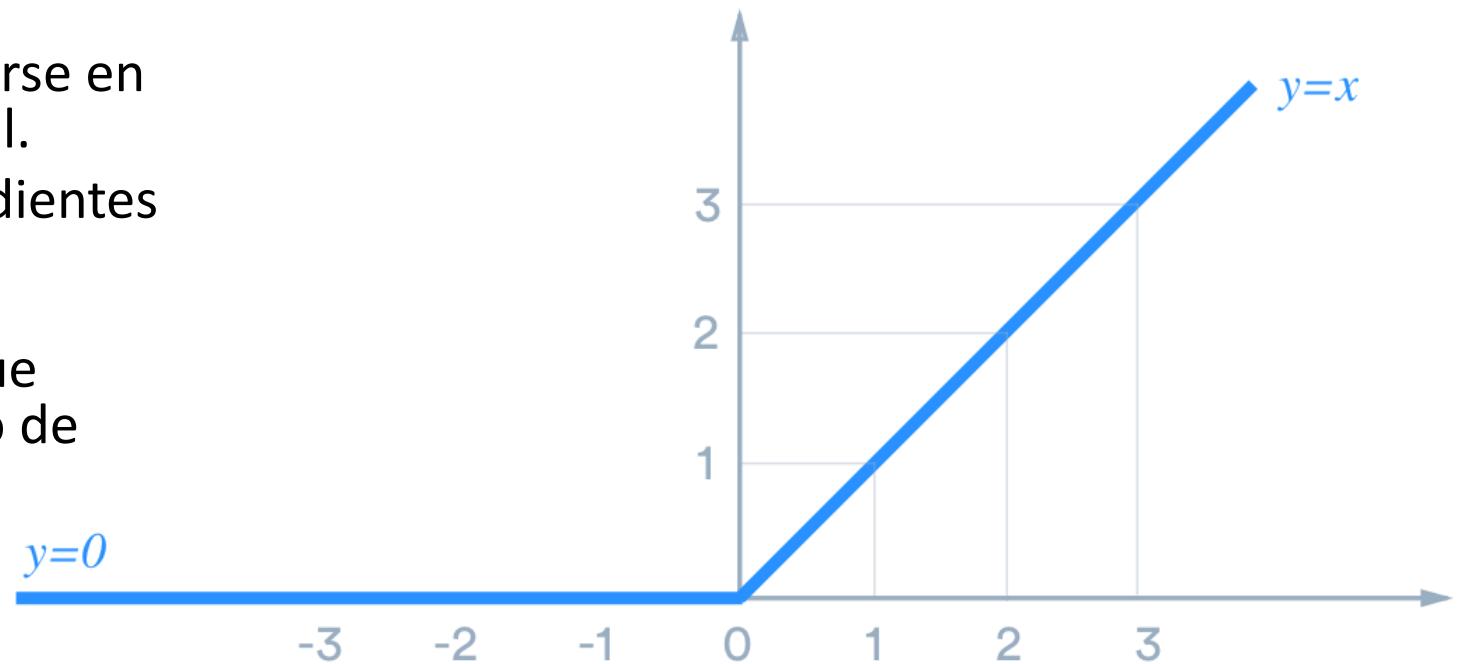
# Funciones de activación según su rango o curvas

- **Función de Tangente Hiperbólica (*tanh*)**
- $F(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$
- Su salida es centrada en 0 y su rango  $[-1, 1]$
- La optimización es más fácil, por lo que se prefiere a la sigmoid, aunque también tiene el problema de la gradiente que desaparece.



# Funciones de activación según su rango o curvas

- **Función ReLU (Rectified Linear Units)**
- $R(x) = \max(0, x)$  i-e si  $x < 0$ ,  $R(x) = 0$ , si  $x \geq 0$ ,  $R(x) = x$
- Resuelve el problema de la gradiente que desaparece.
- Pero su limitación es que debe usarse en capas internas de una red neuronal.
- Otro problema es que algunas gradientes pueden ser frágiles durante el entrenamiento. Esto provoca una actualización de pesos que hará que nunca se active en cualquier punto de nuevo (“neuronas muertas”)



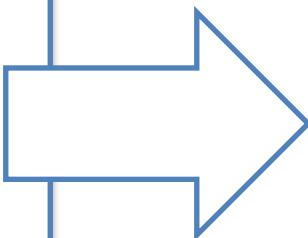
### **4.4.3. ENTRENAMIENTO**

Fuente parcial: **Mingyue Tan, UBC**

# Receta de Deep Learning

**1. Dado un conjunto de datos de entrenamiento**

$$\{x_i, y_i\}_{i=1}^N$$



No es Cara



**2. Elegir una de las siguientes:**

- Función de decisión

$$\hat{y} = f_{\theta}(x_i)$$



- Función de pérdida

$$\ell(\hat{y}, y_i) \in \mathbb{R}$$



Ejemplos:  
Regresión Lineal, Regresión Logística, Red Neuronal

Ejemplos:  
Mean-squared error, Cross Entropy

# Receta de Deep Learning

**1. Dado un conjunto de datos de entrenamiento**

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$$

**2. Elegir una de las siguientes:**

- Función de decisión

$$\hat{\mathbf{y}} = f_{\theta}(\mathbf{x}_i)$$

- Función de pérdida

$$\ell(\hat{\mathbf{y}}, \mathbf{y}_i) \in \mathbb{R}$$

**3. Definir objetivo**

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^N \ell(f_{\theta}(\mathbf{x}_i), \mathbf{y}_i)$$

**4. Entrenar con SGD (pequeños pasos opuestos a la gradiente)**

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \ell(f_{\theta}(\mathbf{x}_i), \mathbf{y}_i)$$

# Receta de Machine Learning

1. Dado un conjunto de datos de entrenamiento

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$$

2. Elegir una de las siguientes:

- Función de decisión

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

- Función de pérdida

$$\ell(\hat{\mathbf{y}}, \mathbf{y}_i) \in \mathbb{R}$$

3. Definir objetivo

¡Back-propagation puede computar este gradiente!

Es un caso especial de un algoritmo más general, llamado *reverse-mode automatic differentiation*, que puede computar eficientemente la gradiente en cualquier función diferenciable.

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i)$$

# Aprendizaje basado en gradiente

Diseñar y entrenar una red neuronal no es muy diferente de entrenar otros modelos de Machine Learning con descenso de gradiente.

La principal diferencia con los modelos lineales es que la no-linealidad de la NN logra que las funciones de pérdida más interesantes se vuelvan no-convexas.

Las NN se entranan – usualmente – con optimizadores iterativos basados en gradiente que llevan la función de pérdida a cero (minimizan).

Pero en funciones de pérdida no-convexas, no hay garantía de convergencia.

# Función de Costo

La elección de una función de costo es importante.

- Máxima similitud
- Funcional (estadística condicional) como el MSE.

Se relaciona con la unidad de la salida

- Unidades lineales para distribuciones Gaussianas
- Unidades Sigmoid para distribuciones Bernoulli
- Unidades Softmax para distribuciones Multinoulli
- Otras.

# Back-propagation y otros algoritmos de diferenciación

El flujo en una red se basa en recibir una entrada X y producir una salida Y.

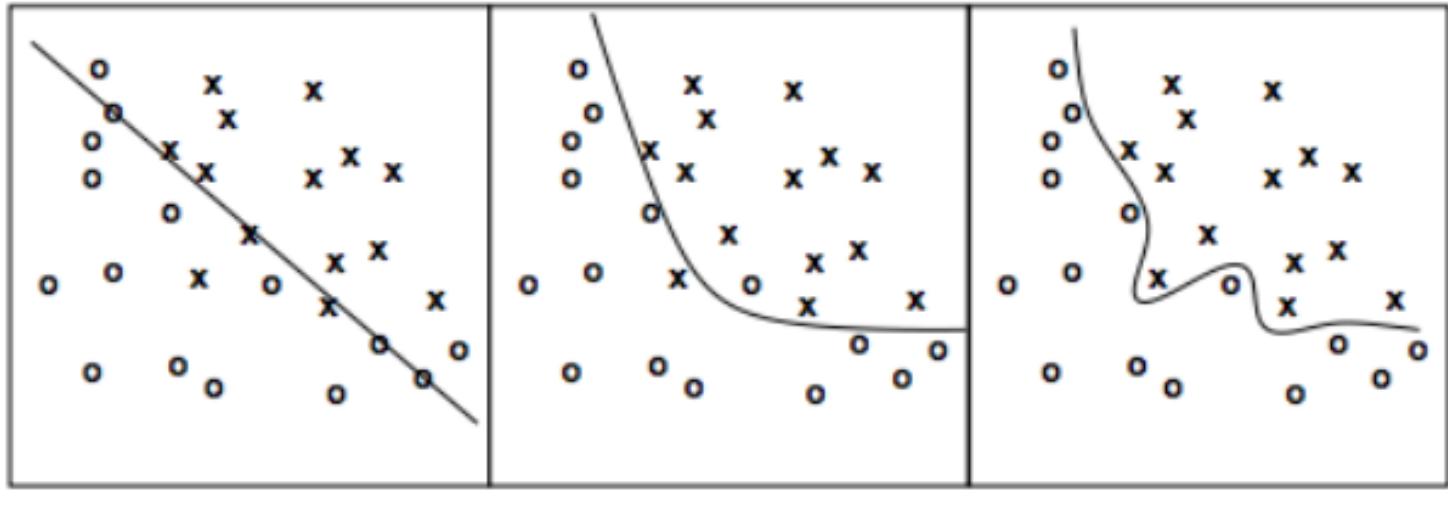
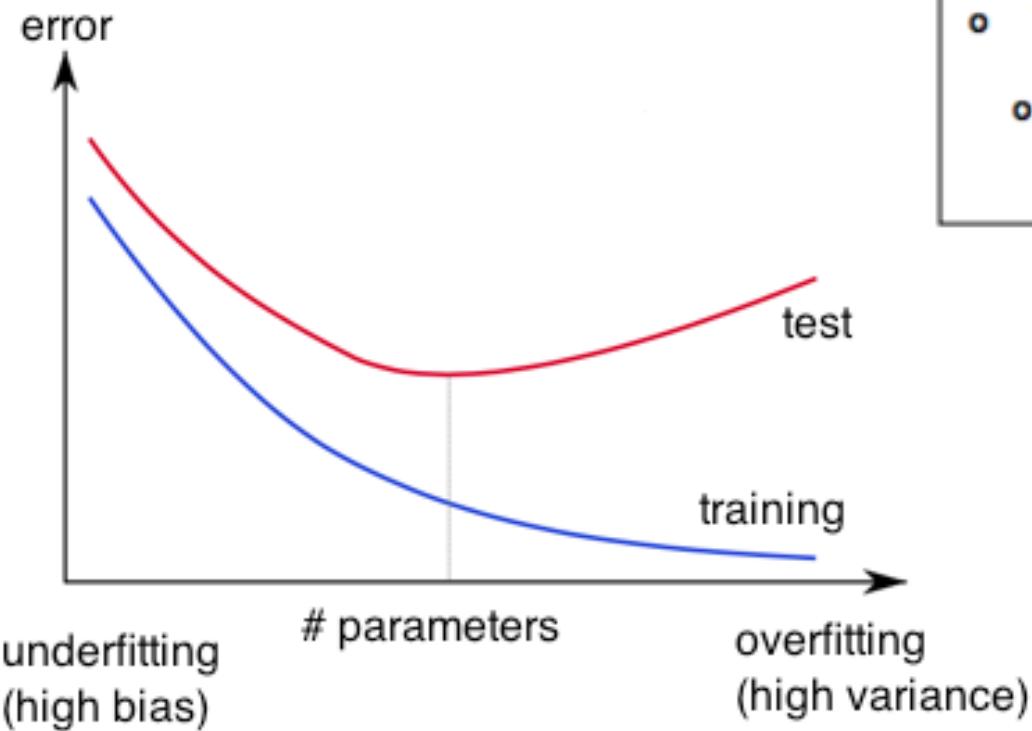
Pero en el entrenamiento, el algoritmo de back-propagation permite que la información fluya desde el costo al flujo en reversa a través de la red para poder computar el gradiente.

Entonces, el gradiente se usa para realizar el entrenamiento, usualmente por un Descenso de Gradiente Estocástico, o Stochastic Gradient Descent (SGD)

## **4.4.4. OVERFITTING Y REGULARIZACIÓN**

# Overfitting

<http://wiki.bethanycrane.com/overfitting-of-data>



La hipótesis aprendida puede calzar con datos de entrenamiento muy bien, incluso con outliers (ruido), pero fallar en la generalización de nuevos ejemplos (datos de test)

[https://www.neuraldesigner.com/images/learning/selection\\_error.svg](https://www.neuraldesigner.com/images/learning/selection_error.svg)

# Regularización

*“Regularización es cualquier modificación realizada al algoritmo de aprendizaje con la intención de reducir su error de generalización, pero no su error de entrenamiento.”*

*Ian Goodfellow 2016*

- En particular, para las redes neuronales, las técnicas apuntan a reducir los efectos del sobreajuste en nodos y pesos, de modo de evitar que una red neuronal, sea demasiado influenciada por las características del conjunto de entrenamiento.
- Las técnicas a considerar son: Dropout, L1 & L2, Early Stopping

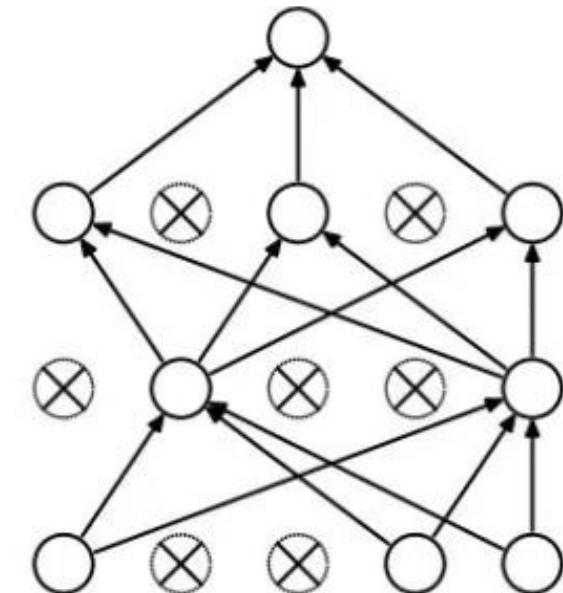
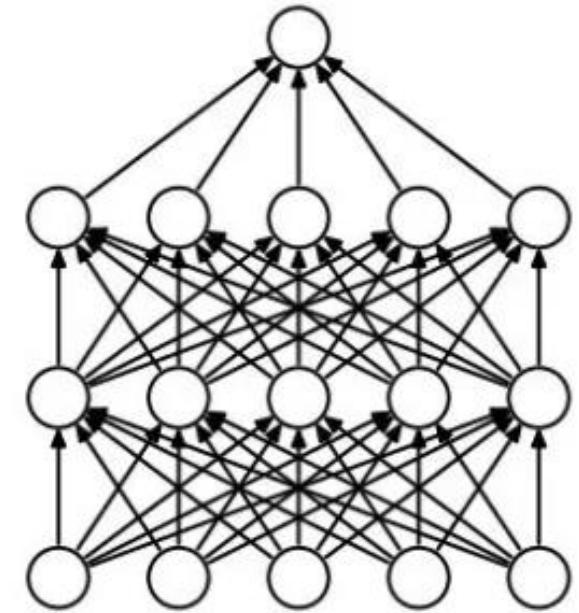
# Regularización - Dropout

- Se eliminan neuronas aleatoriamente durante entrenamiento (incluyendo todas sus conexiones)
- Cada unidad (neurona) es retenida con una probabilidad  $p$ , independientemente de otras unidades.
- La probabilidad  $p$  pasa a ser un hiper-parámetro a ser ajustado en la implementación.

## OBJETIVO

- El beneficio es forzar al modelo a entrenarse de una forma más ruidosa, evitando que se sobreajuste demasiado a puntos que podrían ser ruido en un contexto más generalizado.
- Esto trae la noción de romper situaciones donde las capas de la red se co-adaptan a corregir errores de capas anteriores, logrando que el modelo sea más robusto (generalice mejor).

<https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/>

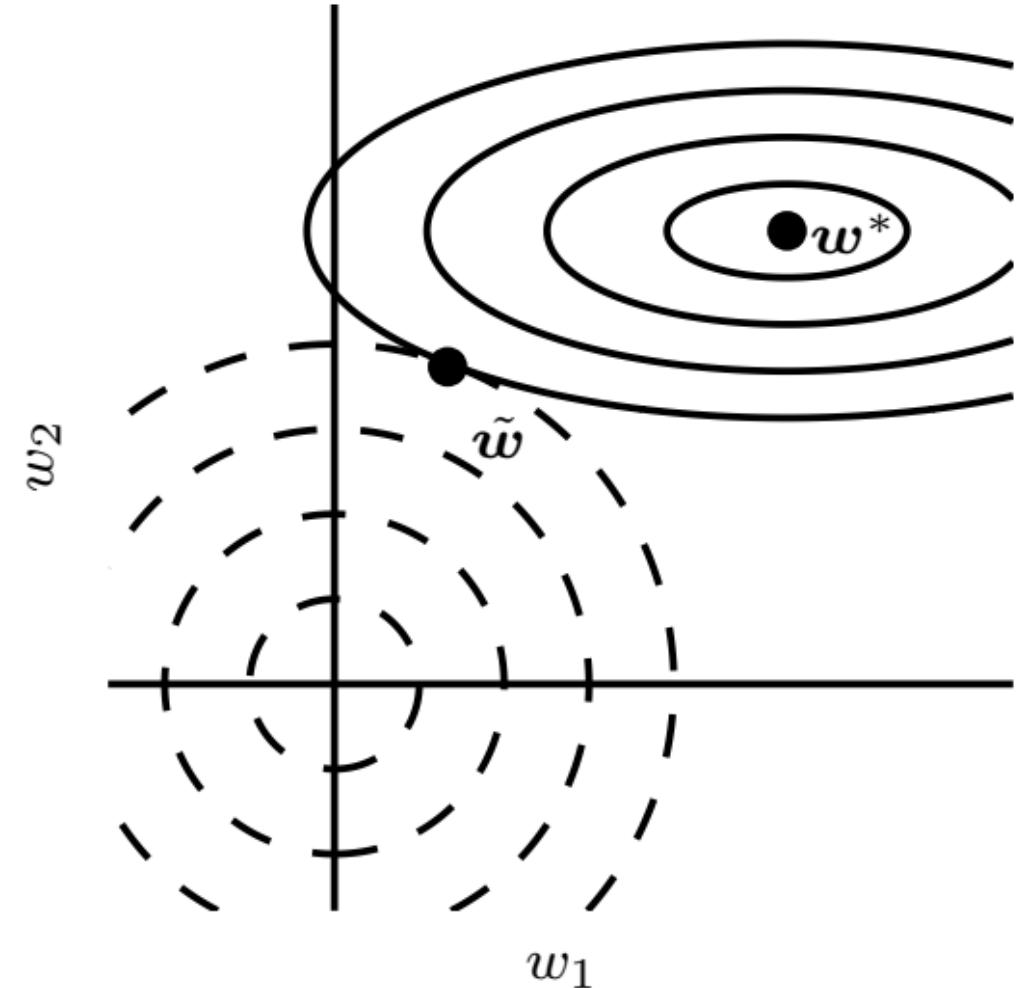


Srivastava, Nitish, et al. *"Dropout: a simple way to prevent neural networks from overfitting."* Journal of machine learning research (2014)

# Regularización – Penalización de Norma

- L1: Fomenta la diversidad, lo que es equivalente a la estimación MAP Bayesiana con Laplace.
- L2 (cuadrado): fomenta pesos pequeños, equivalente a la estimación MAP Bayesiana con Gauss.
  - Penaliza los grandes pesos, agregando a la función objetivo (entrenamiento).
  - El valor de disminución de peso determina cuán dominante es la regularización durante el cálculo del gradiente.
  - Coeficiente de disminución de peso grande → gran penalización para grandes pesos

$$J_{reg}(\theta) = J(\theta) + \lambda \sum_k \theta_k^2$$

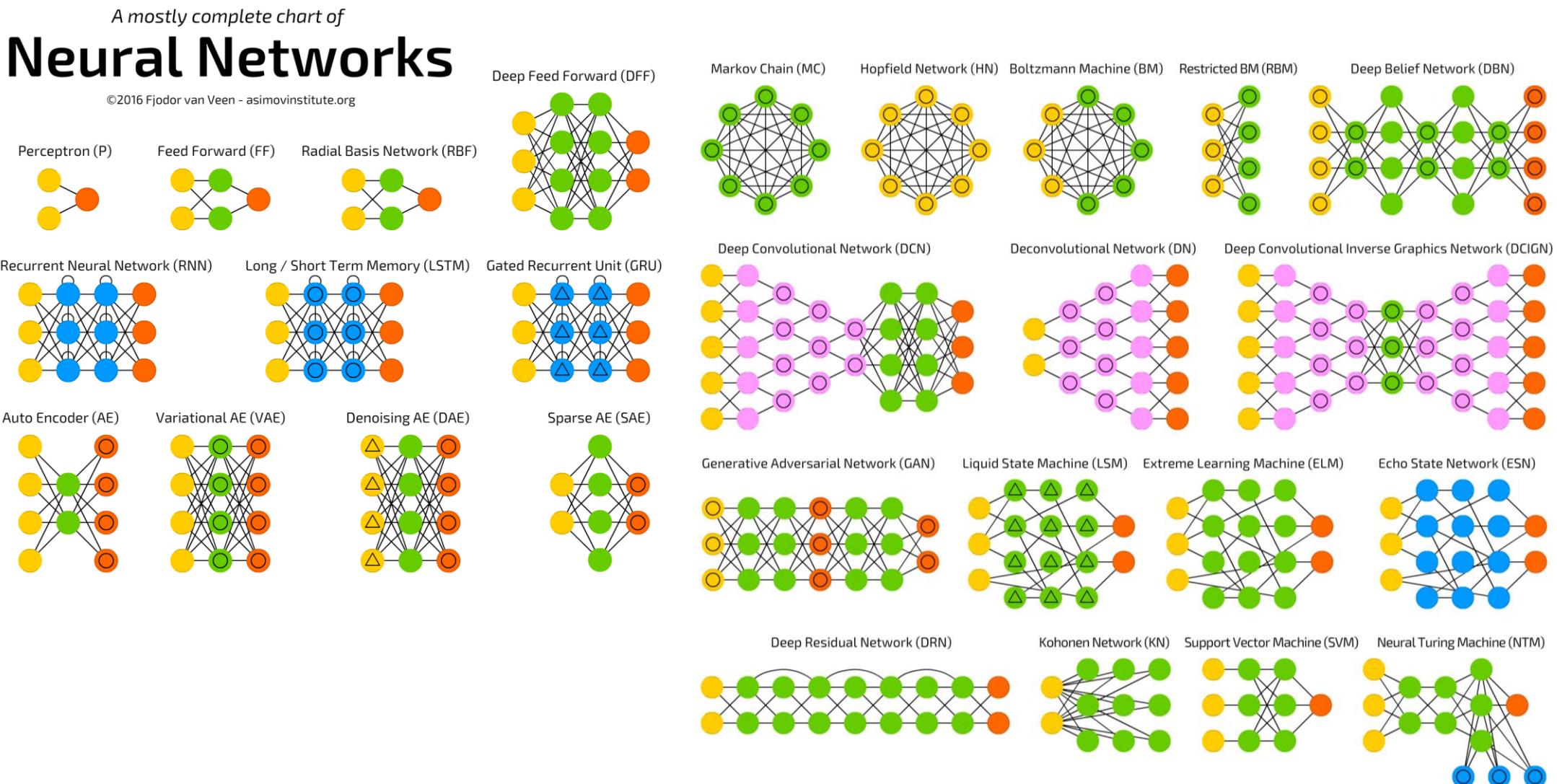


# Regularización - Early Stopping

- Se monitorea el error de validación comparado con el error en entrenamiento. Mientras el 2º no sea notoriamente mejor al de validación, todo bien.
- Cuando el error en el conjunto de validación no ha disminuido por N epochs, o se desvía (aumenta) muy por sobre el de entrenamiento, detener el algoritmo.
- N: “pacienza”

## **4.4.5. ALGUNAS DECISIONES DE ARQUITECTURA**

# Diferentes Topologías de Redes



<https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>

# Diseño Arquitectural

## La estructura de la red

- Cantidad de capas internas o escondidas
- Cantidad de unidades por capa
- Las capas se organizan en cadena (feedforward)
- Cada capa es una función de la capa predecesora

Arquitectura ideal para una tarea en particular debe ser encontrada en forma de pruebas (experimentales), guiadas por un set de validación

# Unidades Escondidas

## La elección del tipo de unidad escondida

- ReLU: no-diferenciable en  $z = 0$   
... porque en entrenamiento no llegará a gradiente  
 $= 0$
- Logística sigmoid o tangente hiperbólica
- Otras.

# Propiedades y Profundidad Aproximación Universal

## Teorema de la aproximación universal

- Independiente de la función que es está aprendiendo, un MLP la puede representar
- En la práctica, una capa escondida única puede representar cualquier función.
- Pero no hay garantía de que el algoritmo de entrenamiento será capaz de aprender esa función.

## El aprendizaje puede fallar por dos razones:

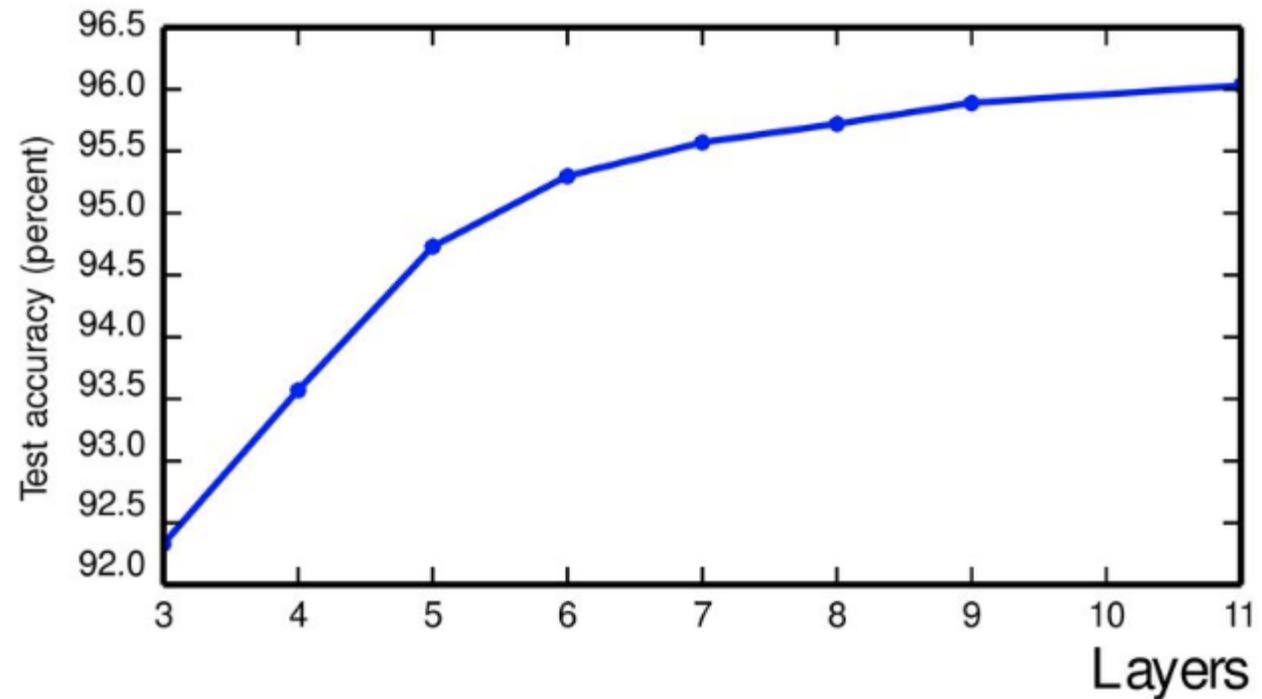
- El algoritmo de entrenamiento puede no encontrar parámetros para la solución
- El algoritmo de entrenamiento puede elegir la función equivocada, producto de overfitting.

# Otras consideraciones arquitecturales

Arquitecturas diferentes  
diseñadas para tareas  
diferentes

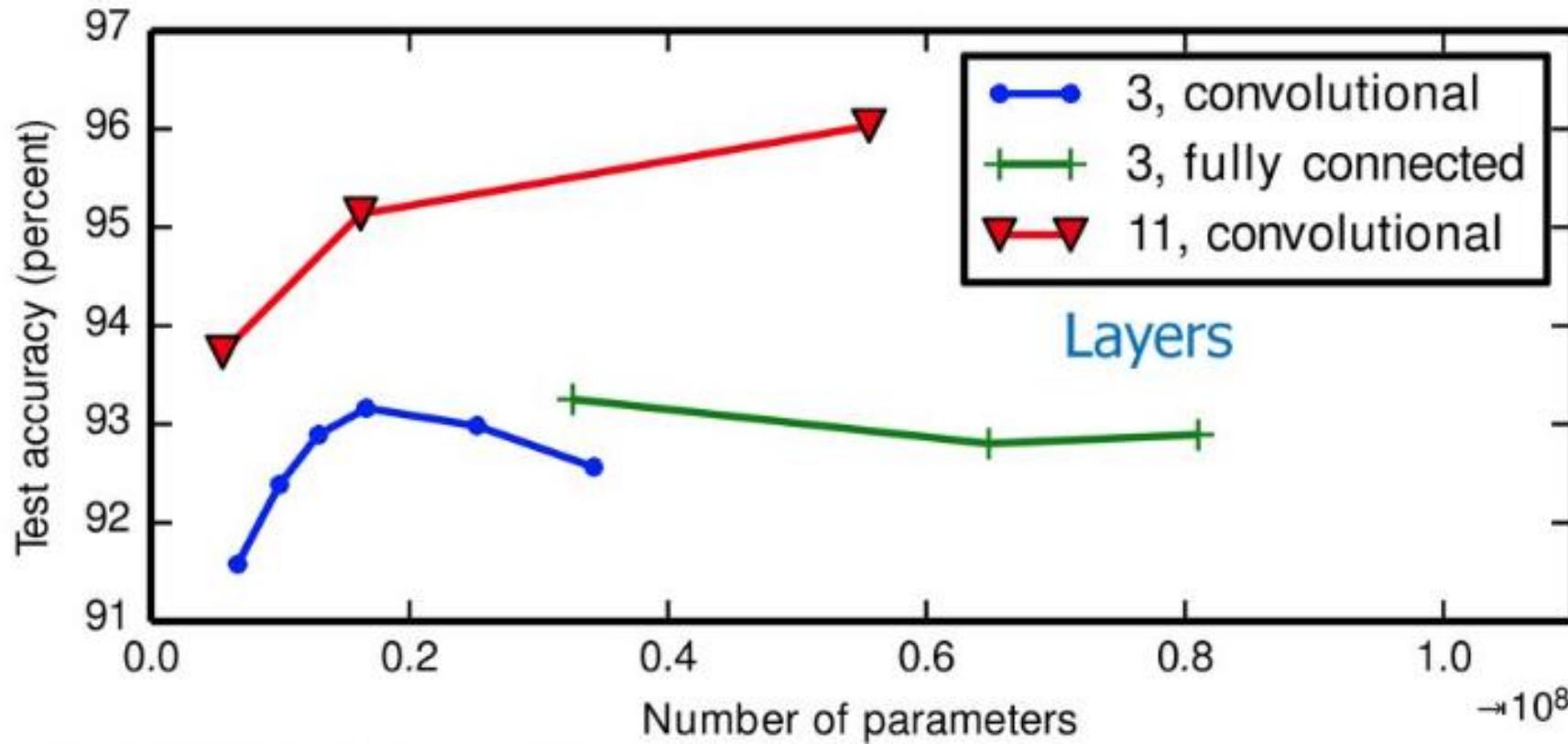
- Hay muchas formas de conectar (mapear) un par de capas
  - Totalmente conectadas
  - Menos conexiones, como las CNN

Las redes más profundas  
tienden a generalizar mejor



(Goodfellow 2016)

# Modelos grandes, pero planos, tienden al Overfitting



- Modelos más planos tienden al sobreajuste (overfitting) cerca de los 20 millones de parámetros.
- Los más profundos se benefician con 60 millones de parámetros.

(Goodfellow 2016)

# Revisando con un ejemplo

# Ejemplo: “Aprendiendo” XOR

X1	X2	X1 XOR X2
SI	SI	NO
SI	NO	SI
NO	SI	SI
NO	NO	NO

El desafío es calzar con un conjunto de entrenamiento acotado. No hay necesidad de generalizar.

Se trata esto como un problema de regresión, utilizando la función de error cuadrado medio (MSE)

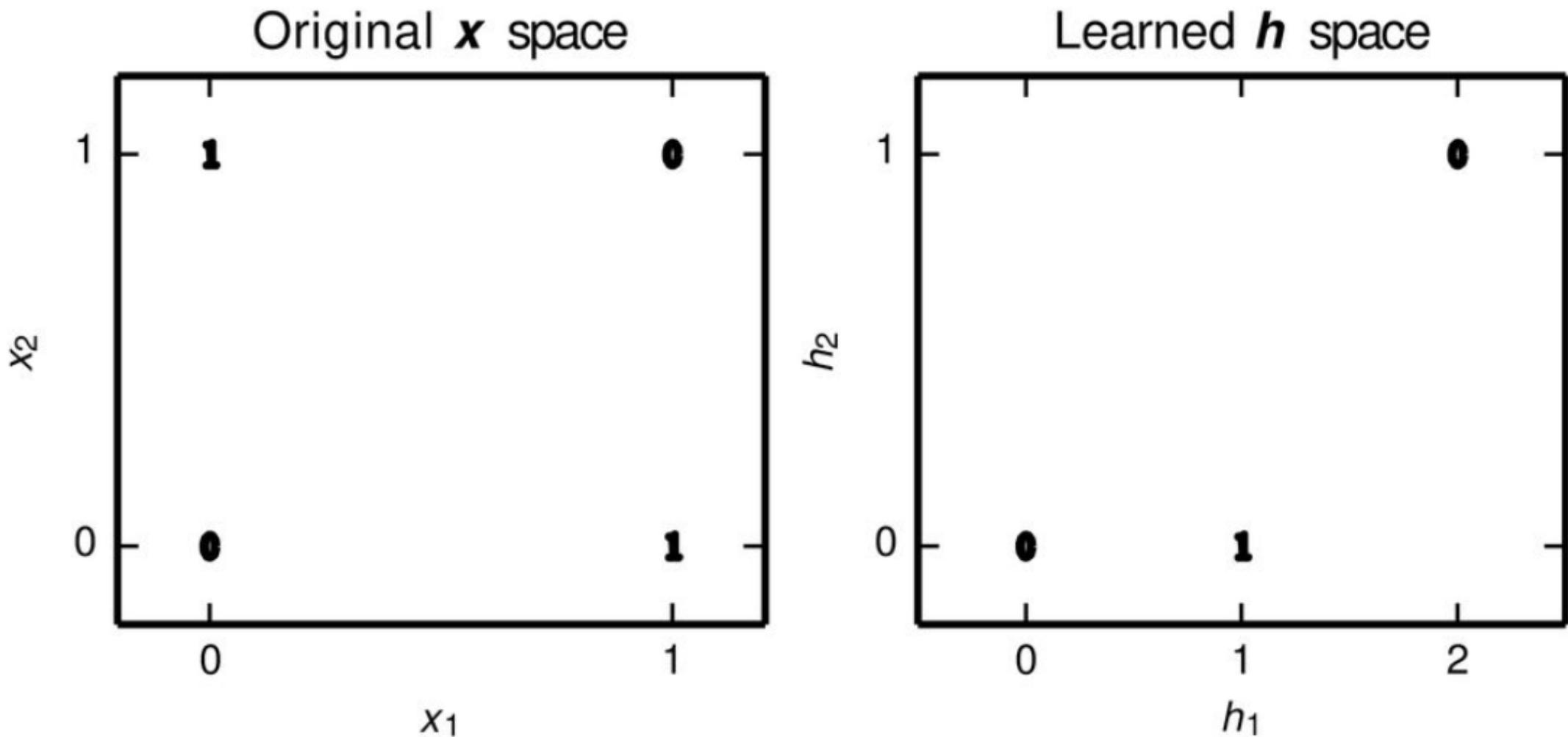
Un modelo lineal no puede representar la función XOR

Entonces, se utiliza un modelo con un espacio de características diferente.

Específicamente, se usa una *Deep Feedforward Network* con una única capa escondida de dos unidades.

Para crear la no-linealidad, se usa una función de activación, aquella “por defecto” en las redes neuronales modernas: **ReLU**

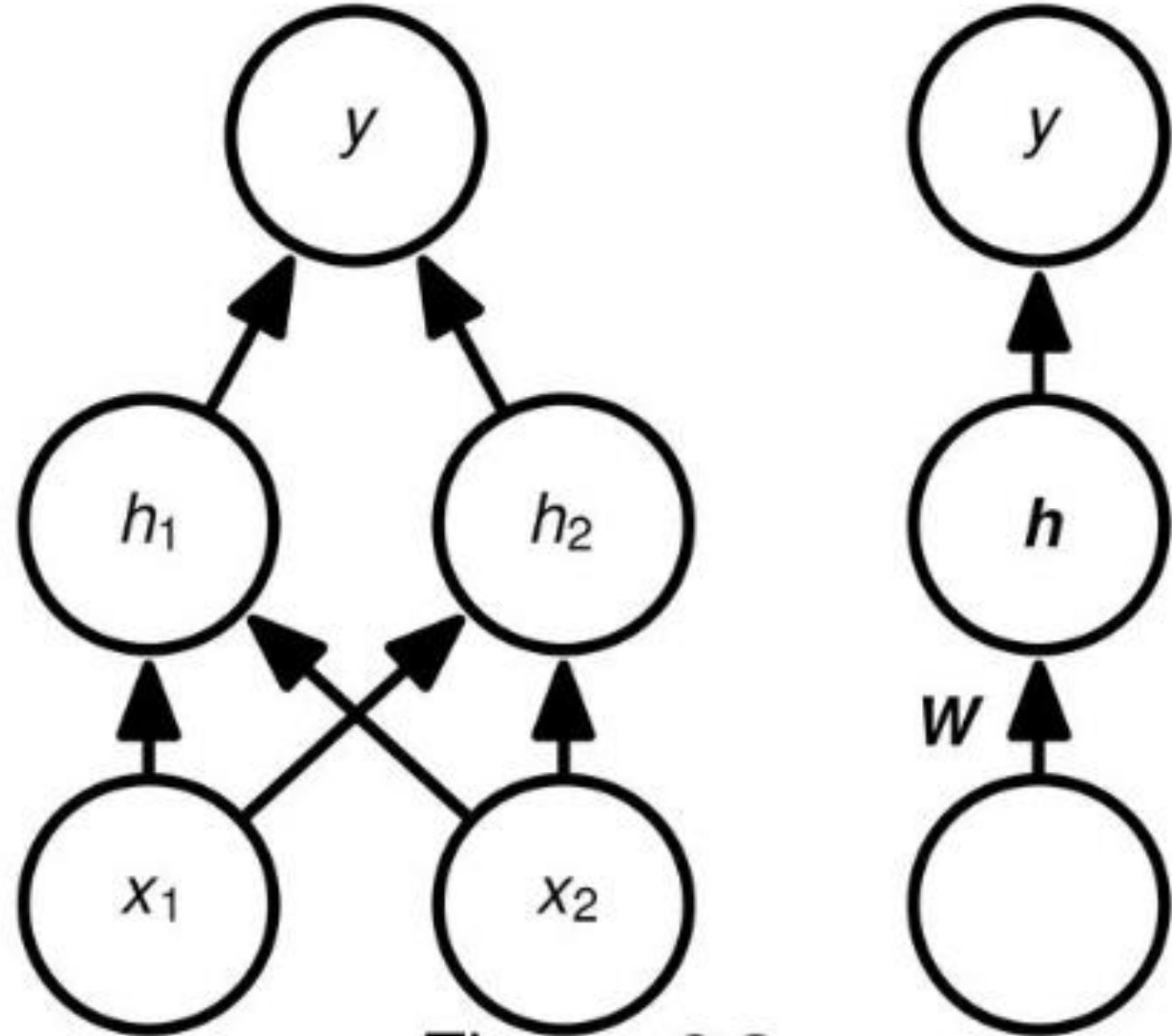
# Resolviendo XOR



X1	X2	X1 XOR X2
SI	SI	NO
SI	NO	SI
NO	SI	SI
NO	NO	NO

- La matriz  $W$  describe el mapeo de  $x$  a  $h$  y el vector  $w$  describe el mapeo  $h \rightarrow y$  (salida)
- La función de activación  

$$h_i = g(\mathbf{x}^\top \mathbf{W}_{:,i} + c_i).$$



# Especificación

- La red completa se especifica:

$$f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, b) = \mathbf{w}^\top \max\{0, \mathbf{W}^\top \mathbf{x} + \mathbf{c}\} + b.$$

- Sean  $\mathbf{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ ,  
 $\mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ ,  $\mathbf{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$   
y  $b=0$

# Proceso: el modelo procesa un lote de entradas

$X$  es la matriz de diseño que contiene 4 puntos en el espacio binario, un ejemplo por fila

$$\mathbf{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

El primer paso en la red neuronal es multiplicar la matriz de entrada por la matriz de pesos de la primera capa

$$\mathbf{XW} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

Se agrega un vector de bias  $c$ , para obtener:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

# Proceso (continuación)

En este espacio todos los ejemplos se mueven en una línea de gradiente 1. Mientras se avanza, la salida debe comenzar en 0 y luego saltar a 1. Para obtener el valor de  $h$  para cada ejemplo, se aplica ReLU

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Esta transformación cambia la relación entre los ejemplos: ya no van en una línea, sino que en un espacio donde un modelo lineal puede resolver el problema.

Así se llega al resultado multiplicando por el vector  $w$

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

# Gracias

 **rsandova@ing.puc.cl**

rodrigo@RSolver.com

 **@RSandovalSolver**

 **/in/RodrigoSandoval**

**www.RodrigoSandoval.net**

www.RSolver.com

# Clases Restantes

## Clase 4

- Reducción dimensional
- Visión Computacional con DL

## Clase 5

- Introducción NLP

## Clase 6

- NLU y modelos avanzados

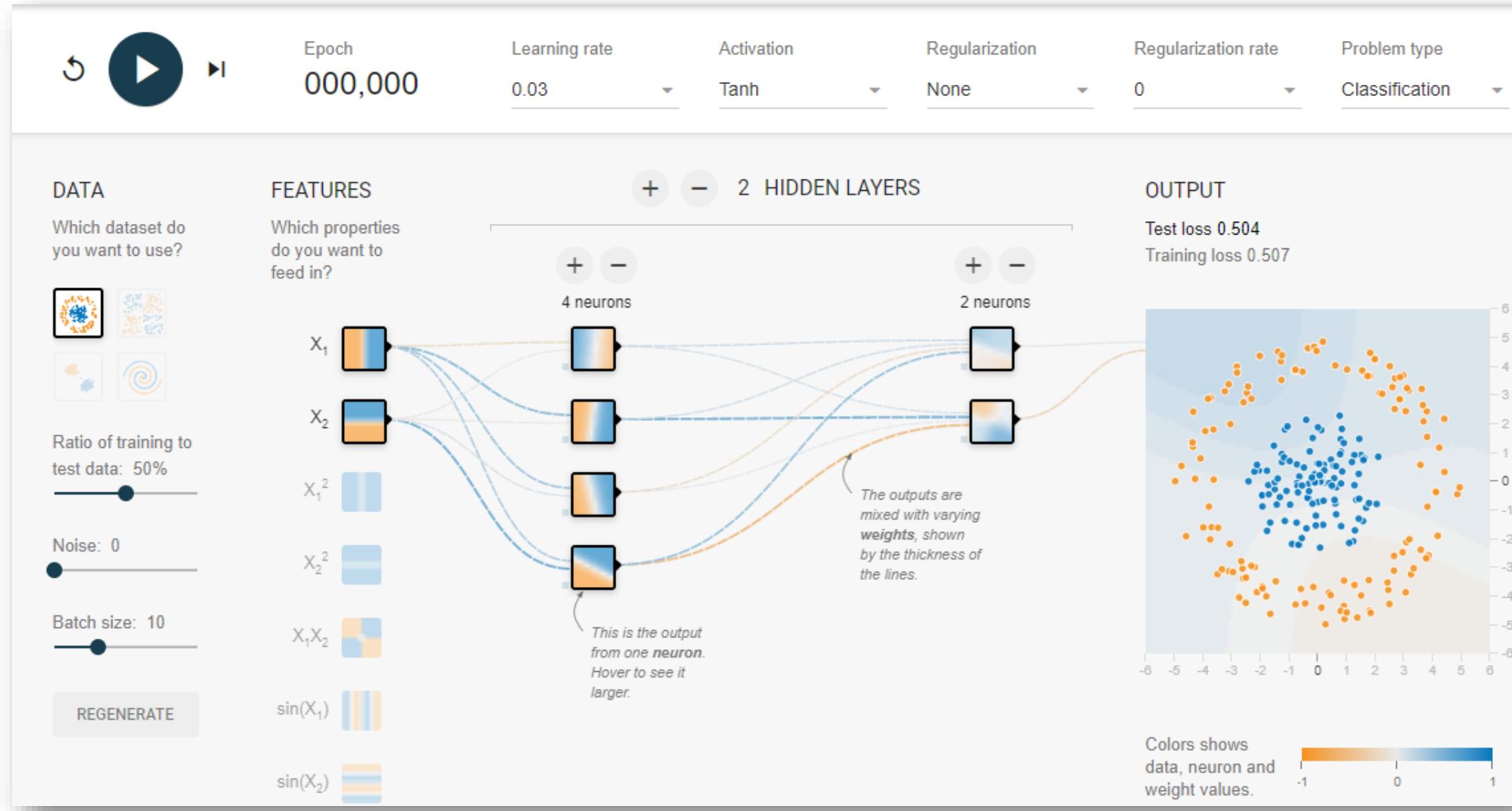
## Clase 7

- Clasificación No-Supervisada

## Clase 8

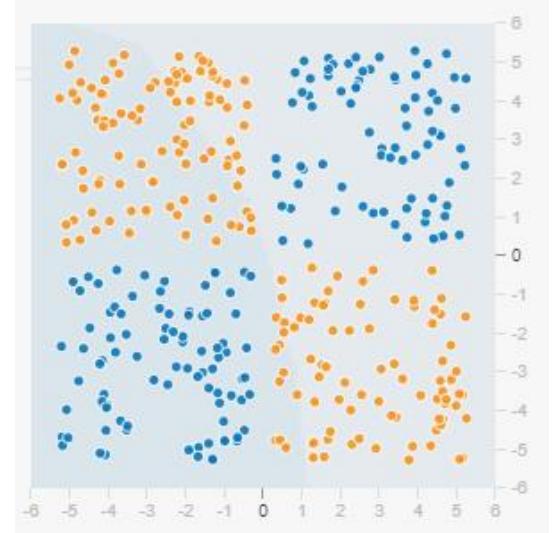
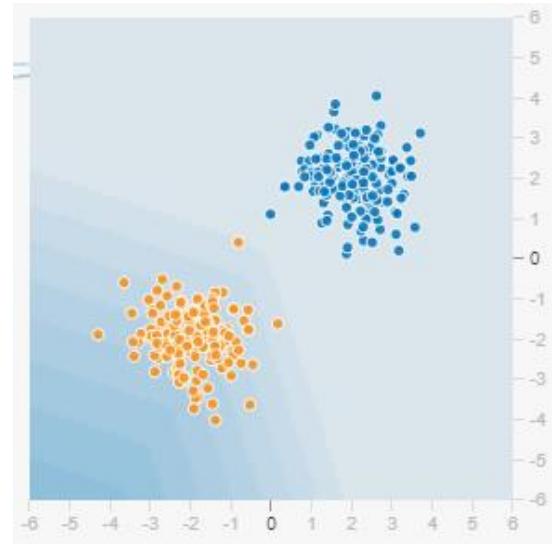
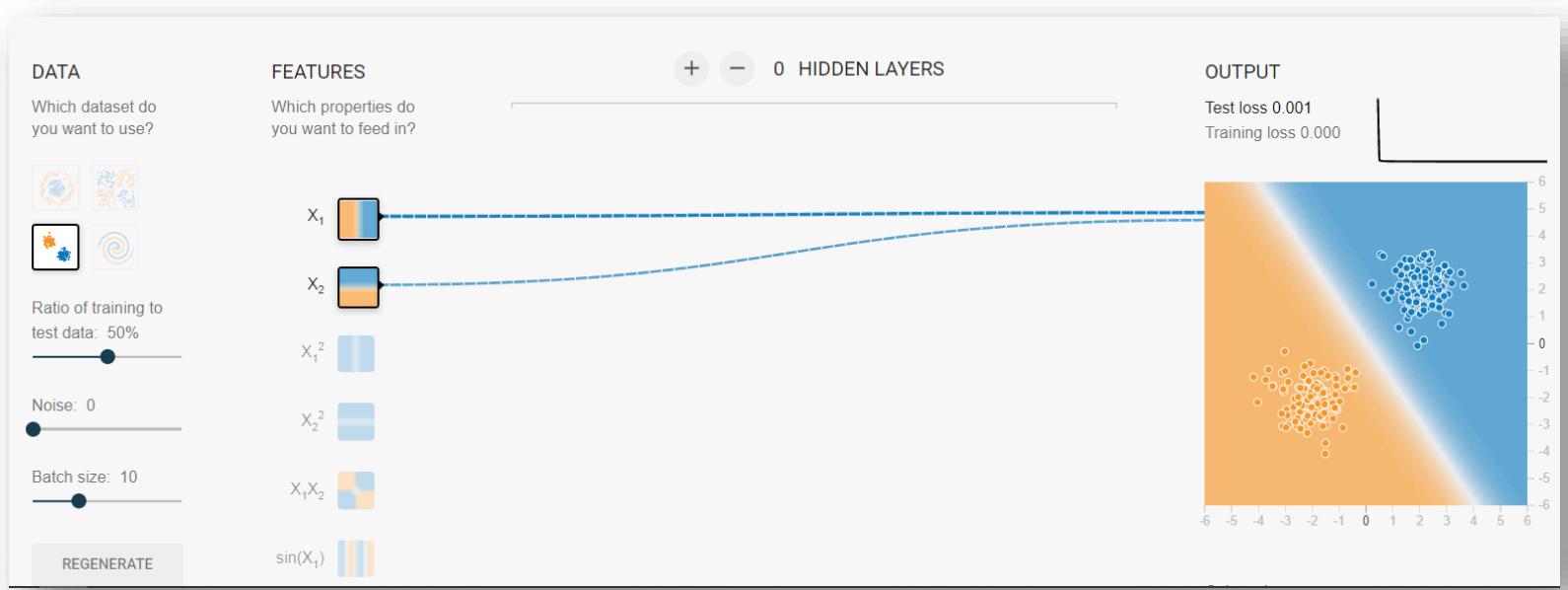
- Proyectos Machine Learning y estado del arte

# Playground.TensorFlow.org



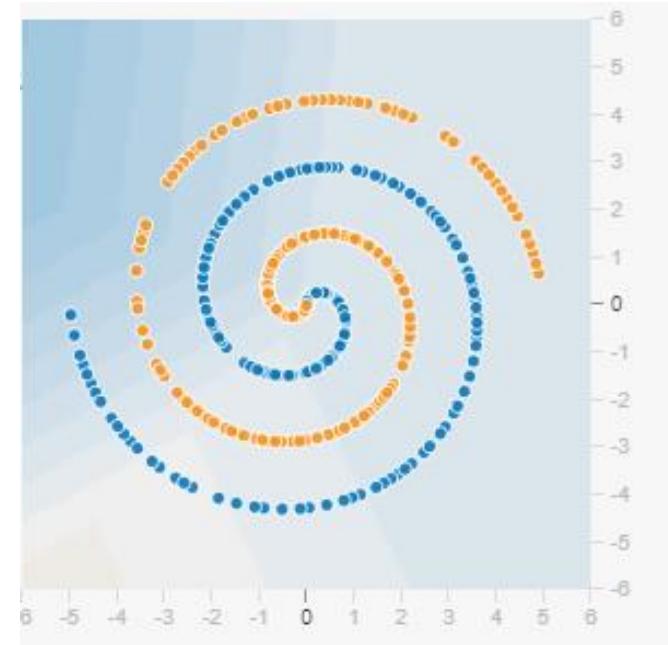
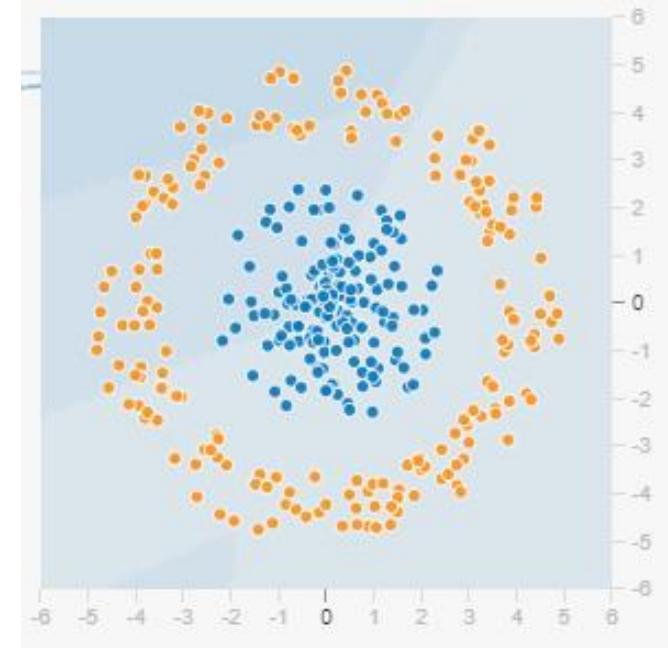
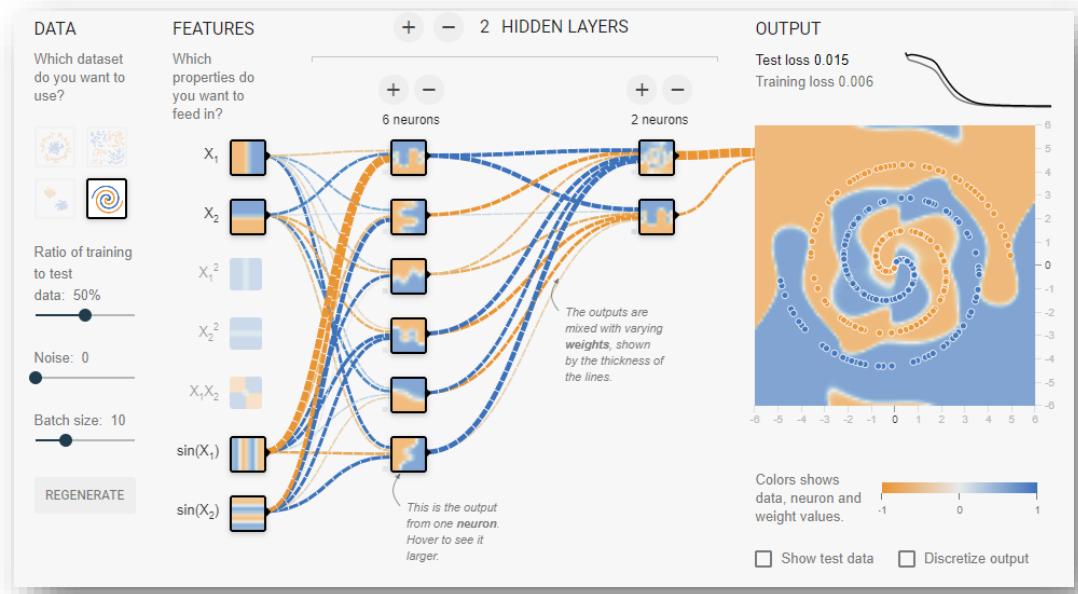
# Ejemplos de clasificación simple

- Para datasets simples (dos o cuatro cuadrantes – ver a la derecha), redes neuronales simples son suficientes:
  - Dos nodos de entrada ( $x_1, x_2$ )
  - Un nodo de salida
  - Ninguna o una capa interna, con pocos nodos



# Ejemplos de clasificación simple

- Para datasets complejos (círculos, espirales – ver a la derecha), se requieren redes neuronales más complejas:
  - Más de 2 nodos de entrada ( $x_1, x_2$  y otros)
  - Un par de nodos de salida
  - Al menos una capa interna, con más nodos ( $> 4$ , por ej)



# Ejercicio

- Probar diferentes combinaciones de:
  - Proporción entrenamiento/test
  - cantidad de capas escondidas
  - cantidad de nodos en las capas escondidas
  - nodos de entrada activados
- Determinar cuál es la combinación más simple (menos elementos) que logra un *loss* menor al 15% en el dataset de espirales.
  - La combinación se describe por #Capas, #Nodos por Capa, Nodos de Entrada activos.

