Introducción a las Redes Neuronales

Marzo de 2026

Contenido

1 Introducción Redes Neuronales

2 Principales Ideas

Introducción

- Las redes neuronales son aproximadores universales: una red con dos capas, capa de salida lineal, es un aproximador universal para michas funciones de activación (excepto polinomios).
- Cualquier función continua en un compacto pues ser aproximada por una red con muchas neuronas en la capa oculta.

IA en Contexto

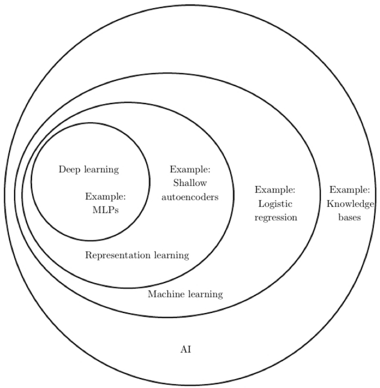


Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.

Historia IA

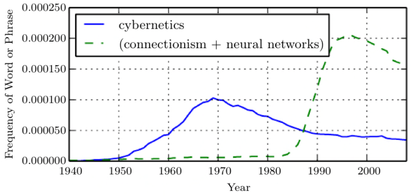


Figure 1.7: The figure shows two of the three historical waves of artificial neural nets research, as measured by the frequency of the phrases "cybernetics" and "connectionism" or "neural networks" according to Google Books (the third wave is too recent to appear). The first wave started with cybernetics in the 1940s–1960s, with the development of theories of biological learning (McCulloch and Pitts, 1943; Hebb, 1949) and implementations of the first models such as the perceptron (Rosenblatt, 1958) allowing the training of a single neuron. The second wave started with the connectionist approach of the 1980–1995 period, with back-propagation (Rumelhart *et al.*, 1986a) to train a neural network with one or two hidden layers. The current and third wave, deep learning, started around 2006 (Hinton *et al.*, 2006; Bengio *et al.*, 2007; Ranzato *et al.*, 2007a), and is just now appearing in book form as of 2016. The other two waves similarly appeared in book form much later than the corresponding scientific activity occurred.

Disponibilidad de Datos

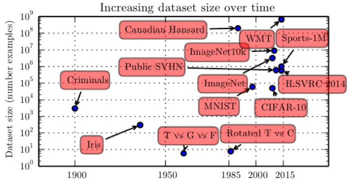


Figure 1.8: Dataset sizes have increased greatly over time. In the early 1900s, statisticians studied datasets using hundreds or thousands of manually compiled measurements (Garson, 1900; Gosset, 1908; Anderson, 1935; Fisher, 1936). In the 1950s through 1980s, the pioneers of biologically inspired machine learning often worked with small, synthetic datasets, such as low-resolution bitmaps of letters, that were designed to incur low computational cost and demonstrate that neural networks were able to learn specific kinds of functions (Widrow and Hoff, 1960; Rumelhart *et al.*, 1986b). In the 1980s and 1990s, machine learning became more statistical in nature and began to leverage larger datasets containing tens of thousands of examples such as the MNIST dataset (shown in Fig. 1.9) of scans of handwritten numbers (LeCun *et al.*, 1998b). In the first decade of the 2000s, more sophisticated datasets of this same size, such as the CIFAR-10 dataset (Krizhevsky and Hinton, 2009) continued to be produced. Toward the end of that decade and throughout the first half of the 2010s, significantly larger datasets, containing hundreds of thousands to tens of millions of examples, completely changed what was possible with deep learning. These datasets included the public Street View House Numbers dataset (Netzer *et al.*, 2011), various versions of the ImageNet dataset (Deng *et al.*, 2009, 2010a; Russakovsky *et al.*, 2014a), and the Sports-1M dataset (Karpathy *et al.*, 2014). At the top of the graph, we see that datasets of translated sentences, such as IBM's dataset constructed from the Canadian Hansard (Brown *et al.*, 1990) and the WMT 2014 English to French dataset (Schwenk, 2014) are typically far ahead of other dataset sizes.

MNIST (Nation Institute of Standard Tech)

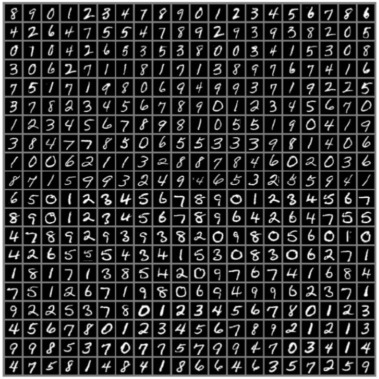


Figure 1.9: Example inputs from the MNIST dataset. The "NIST" stands for National Institute of Standards and Technology, the agency that originally collected this data. The "M" stands for "modified," since the data has been preprocessed for easier use with machine learning algorithms. The MNIST dataset consists of scans of handwritten digits and associated labels describing which digit 0-9 is contained in each image. This simple classification problem is one of the simplest and most widely used tests in deep learning research. It remains popular despite being quite easy for modern techniques to solve. Geoffrey Hinton has described it as "the *drosophila* of machine learning," meaning that it allows machine learning researchers to study their algorithms in controlled laboratory conditions, much as biologists often study fruit flies.

Conexiones

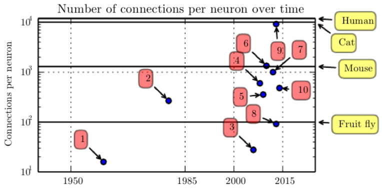


Figure 1.10: Initially, the number of connections between neurons in artificial neural networks was limited by hardware capabilities. Today, the number of connections between neurons is mostly a design consideration. Some artificial neural networks have nearly as many connections per neuron as a cat, and it is quite common for other neural networks to have as many connections per neuron as smaller mammals like mice. Even the human brain does not have an exorbitant amount of connections per neuron. Biological neural network sizes from Wikipedia (2015).

1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett *et al.*, 2009)
6. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
7. Distributed autoencoder (Le *et al.*, 2012)
8. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
9. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
10. GoogLeNet (Szegedy *et al.*, 2014a)

Tamaño Redes

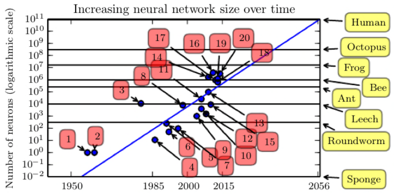


Figure 1.11: Since the introduction of hidden units, artificial neural networks have doubled in size roughly every 2.4 years. Biological neural network sizes from Wikipedia (2015).

1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart *et al.*, 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio *et al.*, 1991)
7. Mean field sigmoid belief network (Saul *et al.*, 1996)
8. LeNet-5 (LeCun *et al.*, 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton *et al.*, 2006)
11. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina *et al.*, 2009)
14. Unsupervised convolutional network (Jarrett *et al.*, 2009)
15. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le *et al.*, 2012)
18. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
19. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
20. GoogLeNet (Szegedy *et al.*, 2014a)

Arquitectura Sistema de IA

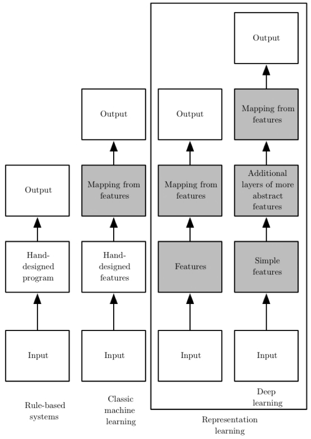


Figure 1.5: Flowcharts showing how the different parts of an AI system relate to each other within different AI disciplines. Shaded boxes indicate components that are able to learn from data.

Representaciones

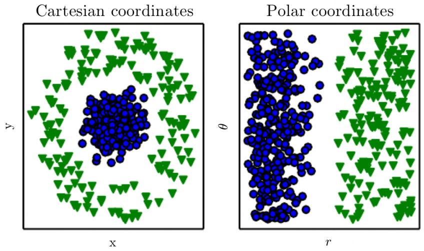


Figure 1.1: Example of different representations: suppose we want to separate two categories of data by drawing a line between them in a scatterplot. In the plot on the left, we represent some data using Cartesian coordinates, and the task is impossible. In the plot on the right, we represent the data with polar coordinates and the task becomes simple to solve with a vertical line. (Figure produced in collaboration with David Warde-Farley)

Contenido

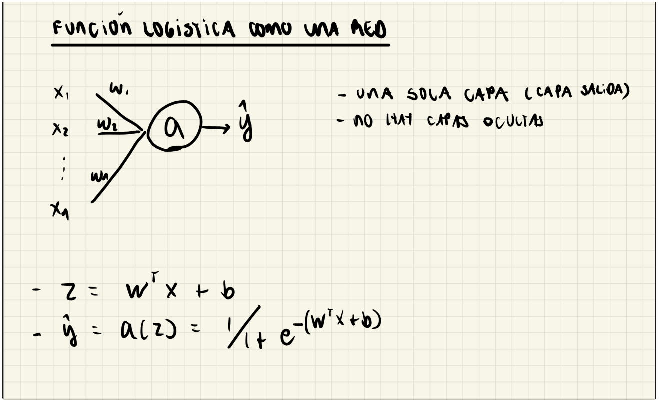
1 Introducción Redes Neuronales

2 Principales Ideas

Introducción

- Redes neuronales (estructura recursiva).
- Funciones de activación.
- Función de perdida.
- Optimización: *back propagation* del gradiente descendiente.

La red más básica: Función logística



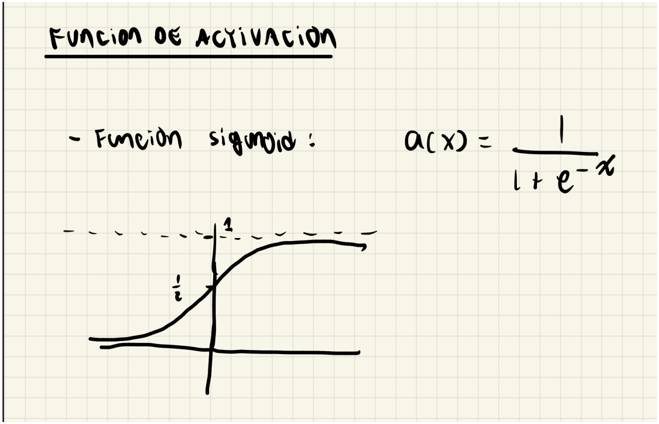
FUNCIÓN LOGÍSTICA COMO UNA RED

- UNA SOLA CAPA (CAPA SALIDA)
- NO HAY CAPAS OCULTAS

- $z = w^T x + b$
- $\hat{y} = a(z) = 1/1 + e^{-(w^T x + b)}$

- El vector de pesos y sesgo de la única capa se denotan por W_1, b_1, respectivamente.

Función de Activación



FUNCION DE ACTIVACION

- Funcion sigmoid:

$$a(x) = \frac{1}{1+e^{-x}}$$

Función de pérdida

- Datos de entrenamiento $\tau = \{(x^1, y^1), ..., (x^n, y^n)\}$

Pérdida con un ejemplo

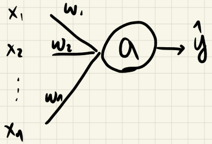
$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -y^{(i)} \ln(\hat{y}^{(i)}) - (1 - y^{(i)}) \ln(1 - \hat{y}^{(i)})$ donde $\hat{y}^{(i)}$ es el pronóstico.

Función de costo

$$
\begin{aligned}
J(w, b) &= \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})) \\
&= \frac{1}{m} \sum_{i=1}^{m} -y^{(i)} \ln(\hat{y}^{(i)}) - (1 - y^{(i)}) \ln(1 - \hat{y}^{(i)})
\end{aligned}
$$

La red más básica: Función logística

FUNCIÓN LOGÍSTICA COMO UNA RED



- UNA SOLA CAPA (CAPA SALIDA)
- NO HAY CAPAS OCULTAS

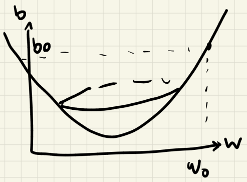
- $z = w^T x + b$
- $\hat{y} = a(z) = 1/_{1+} e^{-(w^T x + b)}$

Optimizacion: Gradiente descendiente

OPTIMIZACIÓN

- GRADIENTE DESCENDENTE

$$\min_{w,b} \; J(w,b) \;\; \longrightarrow$$



$$w \leftarrow w - \alpha \underbrace{\nabla J(w_0, b_0)}_{\equiv dw}$$

$$b \leftarrow b - \alpha \underbrace{\nabla J(w_0, b_0)}_{\equiv db}$$

- Obsérvese que J depende de los n ejemplos de entrenamiento. Si para el cálculo de J y ∇J se usan todos los ejemplo se llama *batch learning*.

Optimizacion: Gradiente descendiente

- Ahora, en general el problema de calcular:

$$\frac{\partial J(w,b)}{\partial w}, \frac{\partial J(w,b)}{\partial b} \tag{1}$$

es dificil, excepto si se explota la estructura recursiva de una red neuronal. Esto es lo que se conoce como *back propagation*, técnica que se apalanca fuertemente en la regla de la cadena.

Optimizacion: Gradiente descendiente

Ejemplo función logística: un ejemplo

- Supongamos que hay dos variables predictoras y tenemos un problema de clasificación.

$$z = w_1 x_1 + w_2 x_2 + b,$$

$$a = \sigma(z),$$

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -y^{(i)} \ln(\hat{y}^{(i)}) - (1 - y^{(i)}) \ln(1 - \hat{y}^{(i)}),$$

$$J(w, b) = \mathcal{L}(\hat{y}^{(i)}, y^{(i)}))$$

$$J(w, b) = \mathcal{L}(a(z), y), \text{ donde } \quad z = w_1 x_1 + w_2 x_2 + b$$

$$\Rightarrow \frac{\partial J(w, b)}{\partial w_1} = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_1} \equiv dW_1$$

- Con función de activación sigmoid:

$$\frac{\partial J}{\partial a} = \frac{a-y}{a(1-a)}, \quad \frac{\partial a}{\partial z} = a^2 e^{-z} = a(1-a), \quad \frac{\partial z}{\partial w_1} = x_1$$

- La derivada para b:

$$\Rightarrow \frac{\partial J}{\partial b} = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial b} \equiv db \qquad \Rightarrow db = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \equiv dz$$

En resumen

Con función de activación sigmoid:

$$dw_j = (a(z^{(i)}) - y^{(i)})x_j^{(i)}$$

$$db = a(z^{(i)}) - y^{(i)}$$

Optimizacion: Gradiente descendiente

Ejemplo función logística: Varios ejemplos

- Ahora calculamos el gradiente cuando hay varios ejemplos:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}))$$

$$\begin{aligned}\frac{\partial J(w,b)}{\partial w_j} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial \mathcal{L}(\hat{y}^{(i)}, y^{(i)})}{\partial w_j} \\ &= \frac{1}{m}\sum_{i=1}^{m}(a(z^{(i)}) - y^{(i)})x_j^{(i)} \\ \frac{\partial J(w,b)}{\partial b} &= \frac{1}{m}\sum_{i=1}^{m}(a(z^{(i)}) - y^{(i)})\end{aligned}$$