Redes Neuronales y Redes Profundas

Enero 2023

① Introducción

- Las redes neuronales son aprox. universales: una red con dos capas, con la capa de salida lineal, para muchas funciones de activación aprox. cualquier función continua en un compacto siempre y cuando se tengan varias neuronas (pag. 230 Bishop).
  Sin embargo, las activaciones no pueden ser polinomios

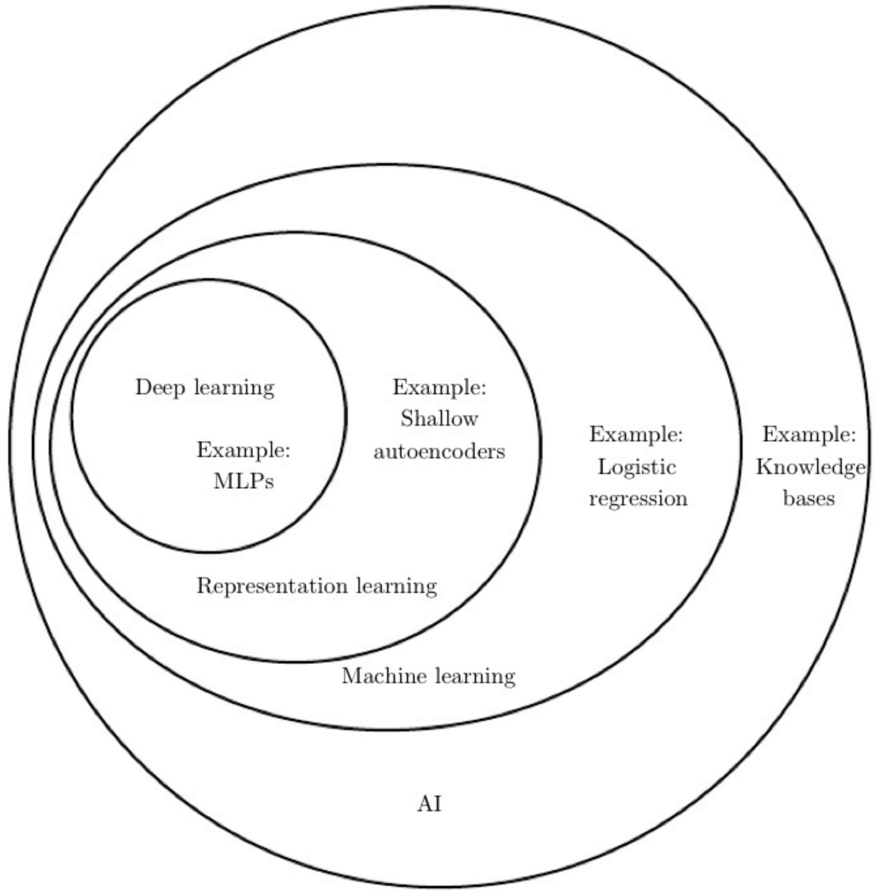


Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.

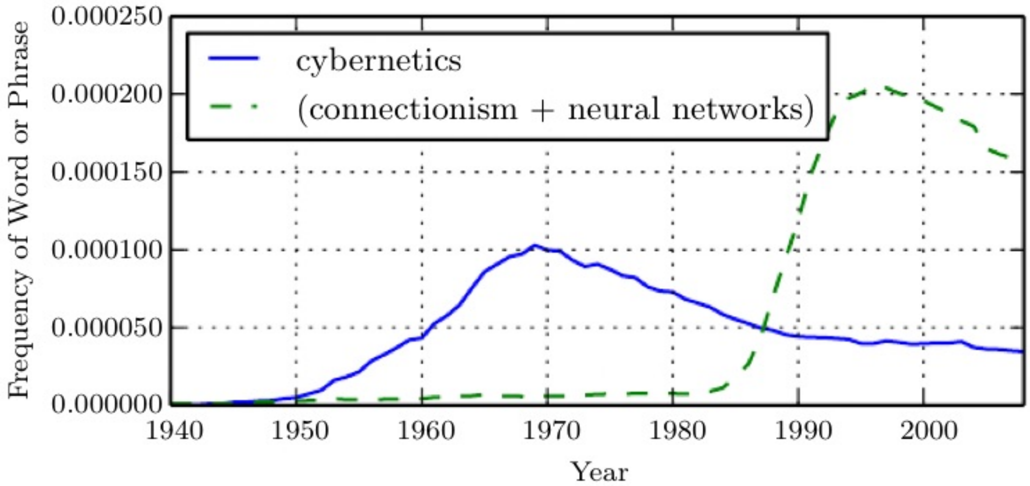


Figure 1.7: The figure shows two of the three historical waves of artificial neural nets research, as measured by the frequency of the phrases "cybernetics" and "connectionism" or "neural networks" according to Google Books (the third wave is too recent to appear). The first wave started with cybernetics 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.

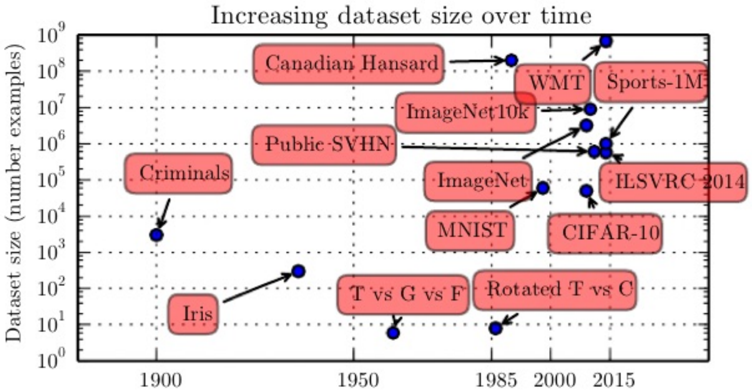


Figure 1.8: Dataset sizes have increased greatly over time. In the early 1900s, statisticians studied datasets using hundreds or thousands of manually compiled measurements (Garson, 1900; Gosset, 1908; Anderson, 1935; Fisher, 1936). In the 1950s through 1980s, the pioneers of biologically inspired machine 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.

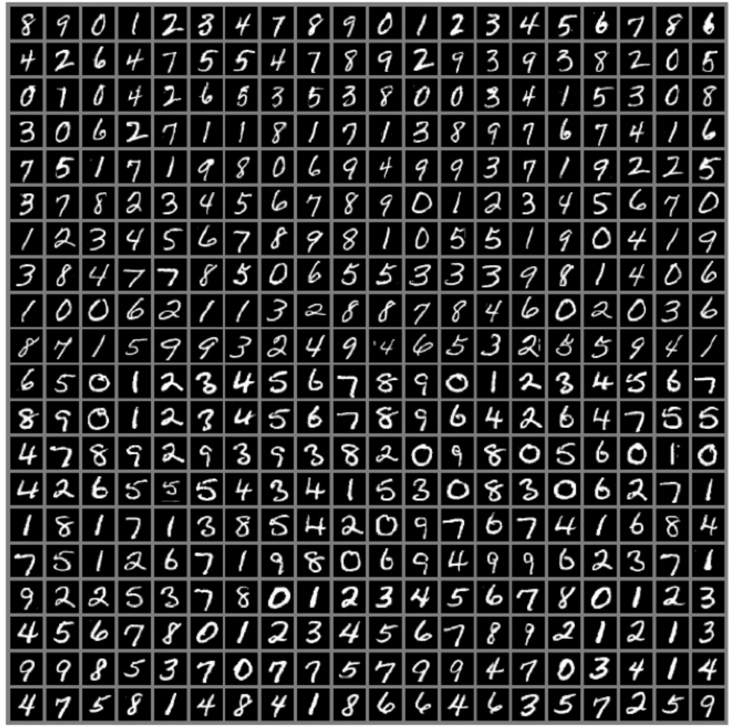


Figure 1.9: Example inputs from the MNIST dataset. The "NIST" stands for National Institute of Standards and Technology, the agency that originally collected this data. The "M" stands for "modified," since the data has been preprocessed for easier use with machine learning algorithms. The MNIST dataset 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.

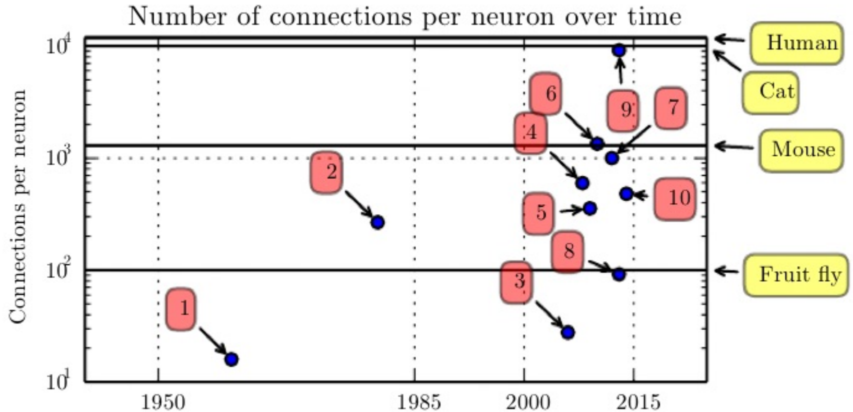


Figure 1.10: Initially, the number of connections between neurons in artificial neural networks was limited by hardware capabilities. Today, the number of connections between neurons is mostly a design consideration. Some artificial neural networks have nearly as many connections per neuron as a cat, 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)

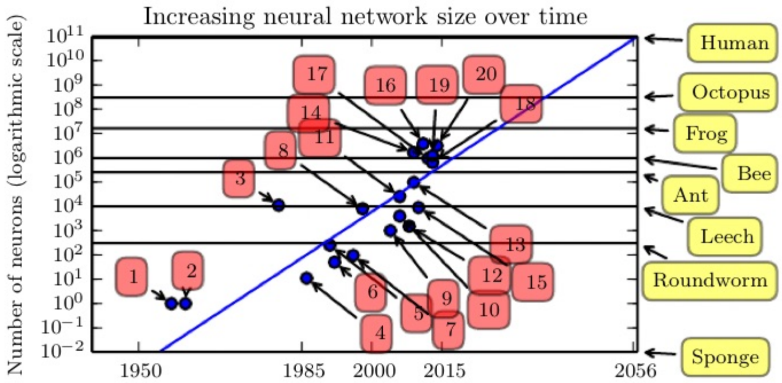


Figure 1.11: Since the introduction of hidden units, artificial neural networks have doubled in size roughly every 2.4 years. Biological neural network sizes from Wikipedia (2015).

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)

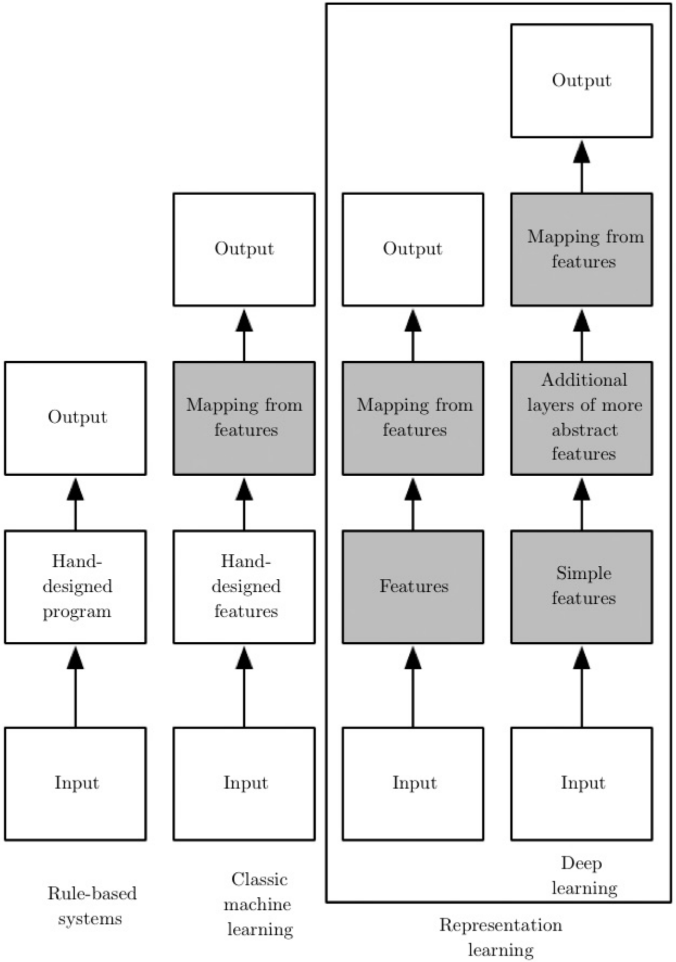


Figure 1.5: Flowcharts showing how the different parts of an AI system relate to each other within different AI disciplines. Shaded boxes indicate components that are able to learn from data.

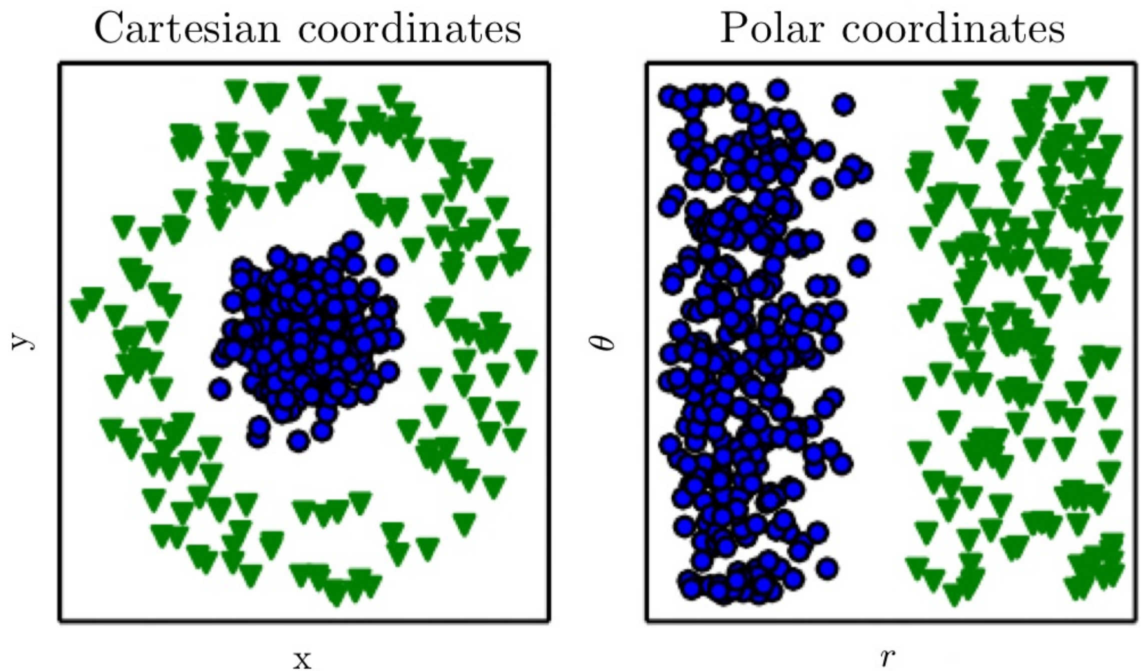


Figure 1.1: Example of different representations: suppose we want to separate two categories of data by drawing a line between them in a scatterplot. In the plot on the left, we represent some data using Cartesian coordinates, and the task is impossible. In the plot on the right, we represent the data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

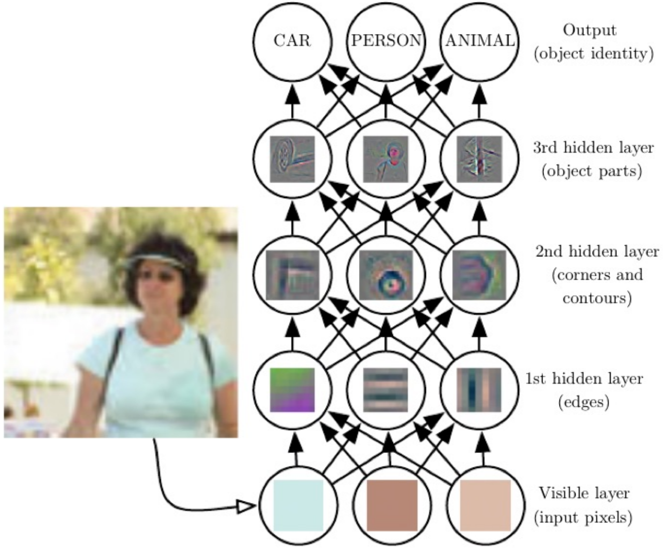


Figure 1.2: Illustration of a deep learning model. It is difficult for a computer to understand the meaning of raw sensory input data, such as this image represented as a collection of pixel values. The function mapping from a set of pixels to an object identity is very complicated. Learning or evaluating this mapping seems insurmountable if tackled directly. Deep learning resolves this difficulty by breaking the desired complicated mapping into a series of nested simple mappings, each described by a different layer of the model. The input is presented at the *visible layer*, so named because it contains the variables that we are able to observe. Then a series of *hidden layers* extracts increasingly abstract features from the image. These layers are called "hidden" because their values are not given in the data; instead the model must determine which concepts are useful for explaining the relationships in the observed data. The images here are visualizations of the kind of feature represented by each hidden unit. Given the pixels, the first layer can easily identify edges, by comparing the brightness of neighboring pixels. Given the first hidden layer's description of the edges, the second hidden layer can easily search for corners and extended contours, which are recognizable as collections of edges. Given the second hidden layer's description of the image in terms of corners and contours, the third hidden layer can detect entire parts of specific objects, by finding specific collections of contours and corners. Finally, this description of the image in terms of the object parts it contains can be used to recognize the objects present in the image. Images reproduced with permission from Zeiler and Fergus (2014).

2 LAS IDEAS PRINCIPALES

A. REDES NEURONALES

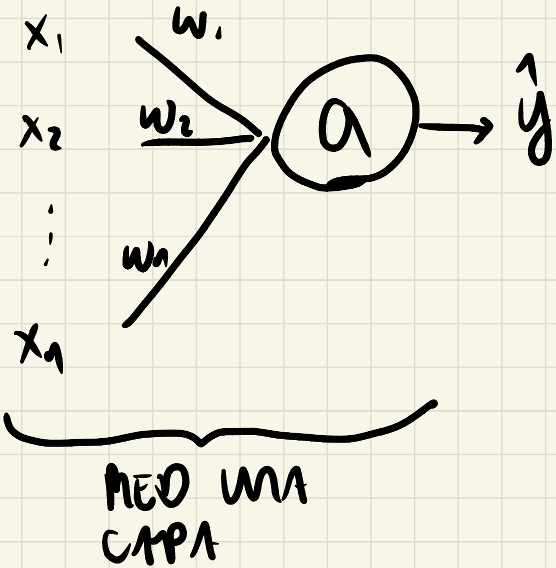
B. FUNCIONES DE ACTIVACION

C. FUNCION DE PERDIDA

D. OPTIMIZACION
   - → GRADIENTE DESCENDIENTE
   - → "BACK PROPAGATION"

A Funcion logistica como una red

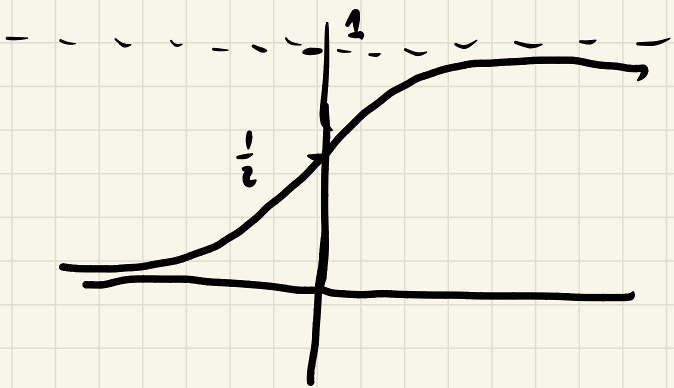
- Una sola capa (capa salida)
- No hay capas ocultas



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

B FUNCIÓN DE ACTIVACIÓN

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



c FUNCIÓN DE PÉRDIDA

- ENTROPÍA CRUZADA
- PÉRDIDA CON UN EJEMPLO:

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

- VARIOS EJEMPLOS:

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

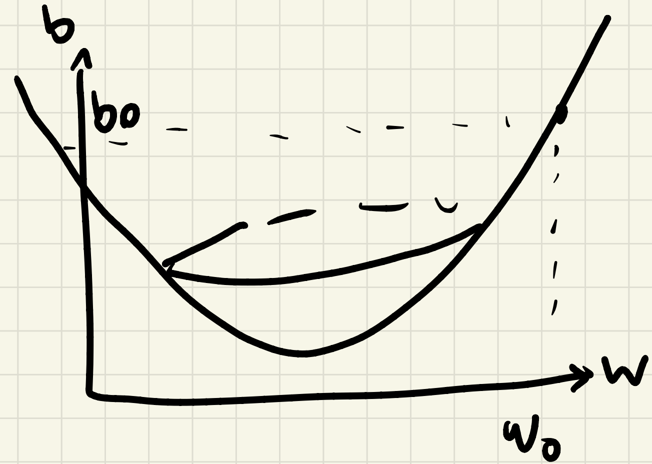
COSTO

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

D OPTIMIZACIÓN

- GRADIENTE DESCENDENTE

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



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

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

- BATCH LEARNING / ONLINE LEARNING (Bishop 5.2.4)

- **BACKPROPAGATION**

ES EL PROBLEMA DE CALCULAR:

$\frac{\partial J}{\partial W}$, $\frac{\partial J}{\partial b}$ usando la regla de la cadena

② EJEMPLOS

EJEMPLO FUNCIÓN LOGÍSTICA COMO UNA RED NEURONAL

- DOS VARIABLES
- UN SOLO EJEMPLO:

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

$$a = \sigma(z)$$

$$\mathcal{L}(y, a) = -y\ln(a) - (1-y)\ln(1-a)$$

$$J(w, b) = \mathcal{L}(y, a)$$

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

$$\Rightarrow \frac{\partial J(w,b)}{\partial w_1} = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_1} \equiv dw_1$$

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

$$\frac{\partial z}{\partial w_1} = x_1$$

$$\Rightarrow$$

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \cancelto{1}{\frac{\partial z}{\partial b}} \equiv db = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z}$$

En resumen

- con funcion de activacion sigmoid: g

$$dz = \frac{\partial J}{\partial z} = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} = \frac{\partial y}{\partial a} \frac{\partial a}{\partial z} = a - y$$

$$dw_1 = \frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_1} = (a-y)x_1$$

$$db = \frac{\partial J}{\partial b}$$

- varios ejemplos

$$\frac{\partial J}{\partial w_1}(w,b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{\partial y}{\partial a^{(i)}} \cdot \frac{\partial a^{(i)}}{\partial z^{(i)}} \cdot \frac{\partial z^{(i)}}{\partial w_1}}_{= dw_1^{(i)}}$$

$$\frac{\partial y}{\partial a^{(i)}} = \frac{a^{(i)} - y^{(i)}}{a^{(i)}(1 - a^{(i)})}$$

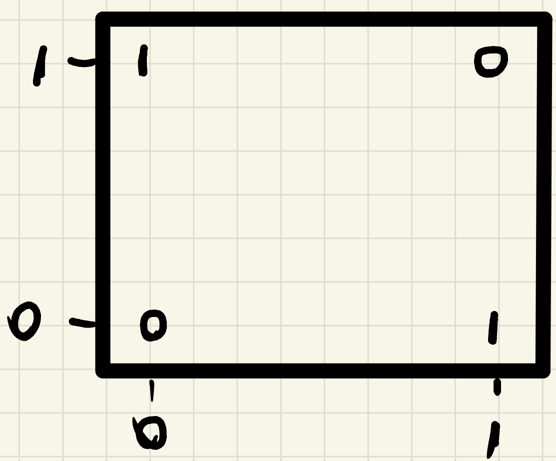
$$\frac{\partial a^{(i)}}{\partial z^{(i)}} = (a^{(i)})^2 e^{-z^{(i)}} = a^{(i)}(1 - a^{(i)})$$

$$\frac{\partial z^{(i)}}{\partial w_1} = x^{(i)} \qquad \frac{\partial y}{\partial b} = \frac{\partial y}{\partial a^{(i)}} \cdot \frac{\partial a^{(i)}}{\partial z^{(i)}}$$

EJEMPLO XOR: APRENDIENDO REPRESENTACIONES

$$x \in \{0,1\} \times \{0,1\}$$

$$f_{XOR}(x) = \begin{cases} 1 & \text{si } x_1 = 1 \ \veebar\ x_2 = 1 \\ 0 & \text{CASO CONTRARIO} \end{cases}$$



- Si tratamos de aproximar XOR por una función lineal:

$$f_{xor}(x) \cong w^T x + b$$

- Si minimizamos el error cuadrático se obtiene $w = 0$, $b = 1/2$:

$$(f_{xor}(0,0) - b)^2 + (f_{xor}(0,1) - w_2 - b)^2 + \ldots$$

output layer
identity

hidden layer
Relu

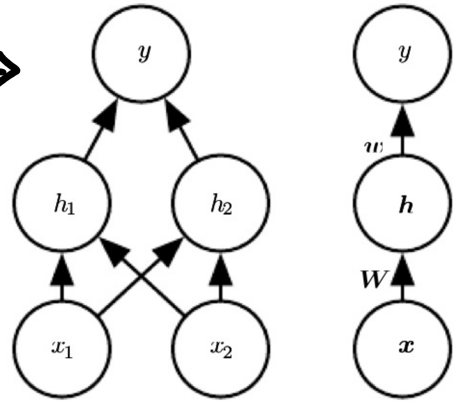


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. *(Left)*In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. *(Right)*In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact. Sometimes we annotate the edges in this graph with the name of the parameters that describe the relationship between two layers. Here, we indicate that a matrix$\boldsymbol{W}$ describes the mapping from $\boldsymbol{x}$ to $\boldsymbol{h}$, and a vector $\boldsymbol{w}$ describes the mapping from $\boldsymbol{h}$ to y. We typically omit the intercept parameters associated with each layer when labeling this kind of drawing.

- $a^{[1]}(x) = \max\left\{0, W^{[1]T}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}b^{[1]}\\b^{[1]}\end{bmatrix}\right\}$ donde

$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix}$$

- $a^{[2]}(x) = W^{[2]T} a^{[1]}(x) + b^{[2]}$

- Al minimizar el error cuadrático:

$$W^{[1]} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad b^{[1]} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \qquad x \in \left\{ \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix} \right\}$$

$$W^{[2]} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \quad b^{[2]} = [0] \qquad y \in \{0, 1, 1, 0\}$$

- Esta red neuronal reproduce perfectamente la función XOR:

suponganmos que $X = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow z^{[1]} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow a^{[1]} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$\Rightarrow z^{[2]} = 1 \Rightarrow a^{[2]} = 1$

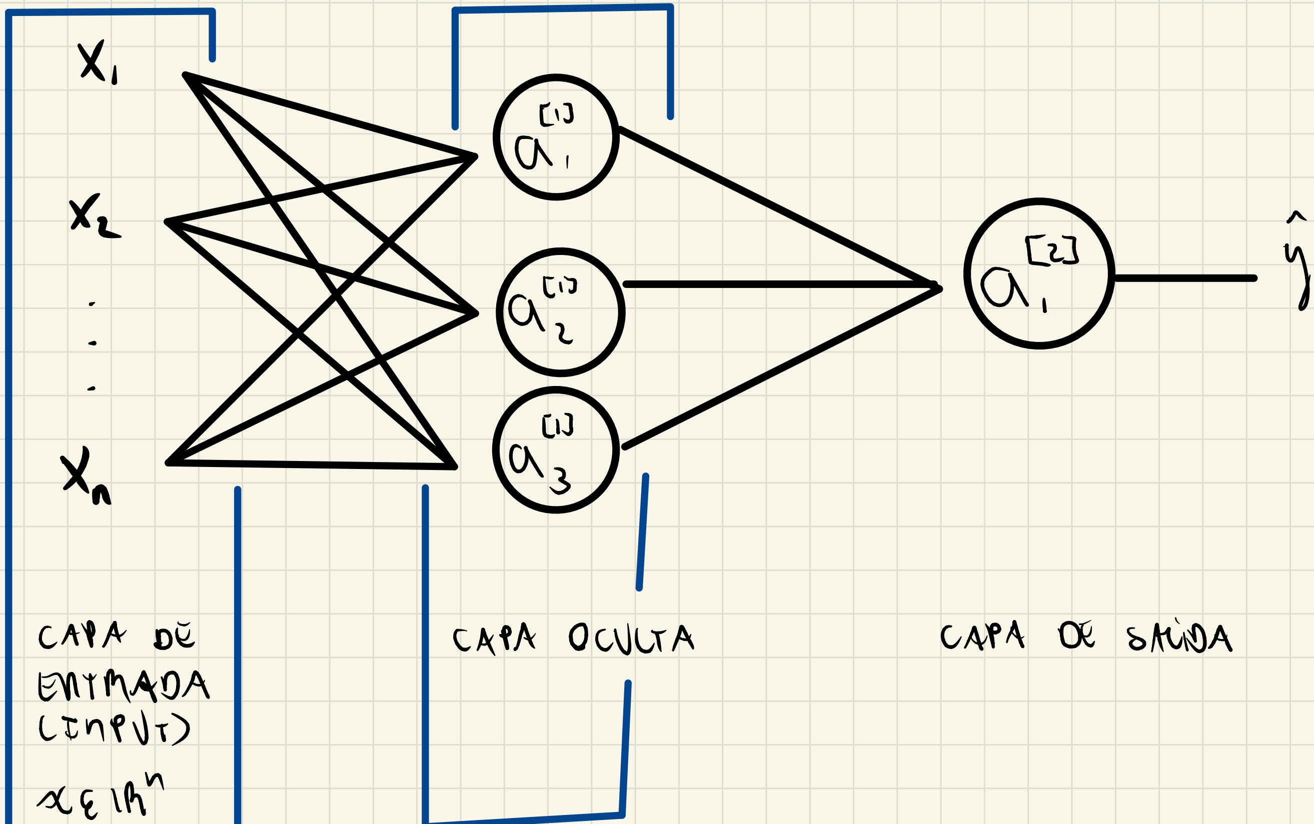
③ RED DE DOS CAPAS

A ARQUITECTURA

B MAS FUNCIONES DE ACTIVACION

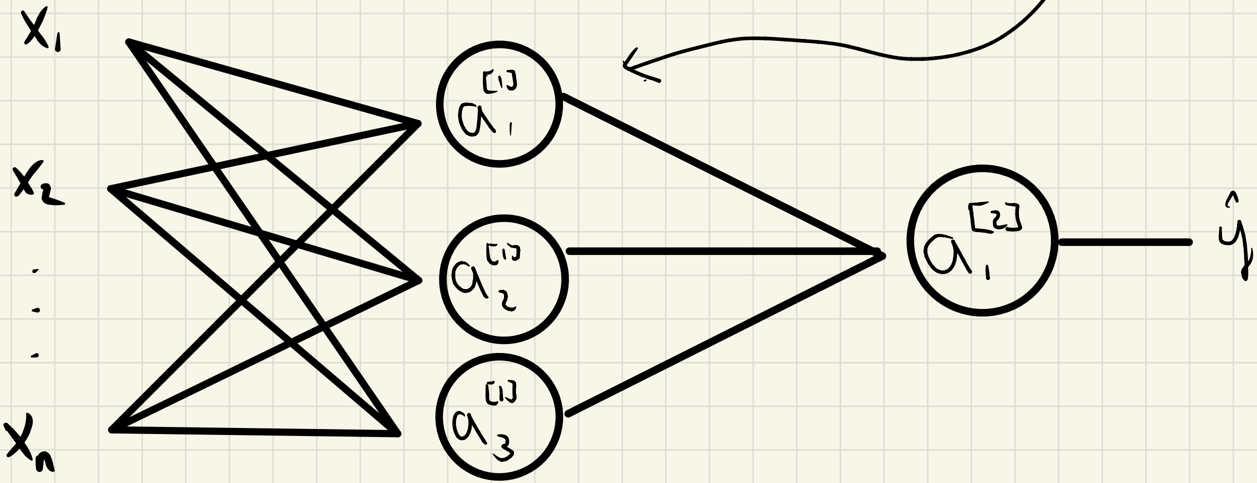
C EJEMPLOS DE APROX. FUNCIONES Y CLASIFICACION

A ARQUITECTURA DOS CAPAS
- ENTRADA
- UNA CAPA OCULTA



$$z^{(i)[1]} = W^{[1]T} x^{(i)} + b^{(i)}$$

$$a^{(i)[1]} = \sigma\left(z^{(i)[1]}\right)$$



CAPA DE ENTRADA (INPUT)

$x \in \mathbb{R}^n$

CAPA OCULTA

CAPA DE SALIDA

FORWARD PROPAGATION

$$\boxed{x} \longrightarrow \boxed{a^{[1]} = g^{[1]}(z^{[1]})} \longrightarrow \boxed{a^{[2]} = g^{[2]}(z^{[2]})} \rightarrow \hat{y}$$

$$z^{[1]} = W^{[1]}X + b^{[1]} \qquad\qquad z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

- FUNCION DE COSTOS UN EJEMPLO:

$$\mathcal{L}(W_1, b_1, W_2, b_2) = L(y_i, \hat{y}_i(W_1, b_1, W_2, b_2))$$

- DERIVADAS DE $\mathcal{L}$ CON RESPECTO A W_1, W_2 ES IGUAL QUE EN EL CASO DE LA FUNCIÓN LOGÍSTICA SOLO QUE AHORA $a^{[1]}$ JUEGA EL PAPEL DE x.

CASO: OUTPUT BINARIO, PÉRDIDA ENTROPIA CRUZADA, ACTIVACION SALIDA SIGMOID

CAPA L

$$dz^{[L]} = a^{[L]} - y$$

$$dW^{[L]} = dz^{[L]} \underbrace{a^{[L-1]}}$$

en vez de x.

$$\boxed{z^{[L]} = W^{[L]} a^{[L-1]} + b^{[L]}}$$

$$db^{[L]} = dz^{[L]}$$

CODIGO: EJEMPLO $L=2$

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

- Observese que una vez se hace el forward propagation se puede calcular $dW^{[L]}$, $db^{[L]}$

<u>CAPA L-1</u> : EJEMPLO CON $L=2$, CAPA OCULTA ACTIVACIÓN $a^{[1]} = g^{[1]}(z^{[1]})$

$$\mathcal{L}(y,\hat{y}(W_1,b_1,W_2,b_2))$$

$$= \mathcal{L}(y, g^{[2]}(W^{[2]} g^{[1]}(W^{[1]}x + b^{[1]}) + b^{[2]}))$$

$$\Rightarrow dW^{[1]} = \underbrace{\underbrace{\frac{\partial \mathcal{L}}{\partial \hat{y}}\frac{\partial g^{[2]}}{\partial z^{[2]}}}_{dz^{[2]} = a^{[2]} - y} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}}\frac{\partial g^{[1]}}{\partial z^{[1]}}}_{dz^{[1]}}\frac{\partial z^{[1]}}{\partial W^{[1]}}$$

$$\frac{\partial g^{[1]}}{\partial z^{[1]}} \rightarrow g_1'(z^{[1]})$$

$$\frac{\partial z^{[1]}}{\partial W^{[1]}} \rightarrow x$$

$$\frac{\partial z^{[2]}}{\partial a^{[1]}} \rightarrow \frac{\partial z_2}{\partial g_1} = W_2$$

<u>CODIGO</u>

$$\boxed{dz^{[1]} = dz^{[2]}\, W_2\, g_1'(z_1^{[1]})}$$

$$\boxed{dW^{[1]} = dz^{[1]}\, x}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$$

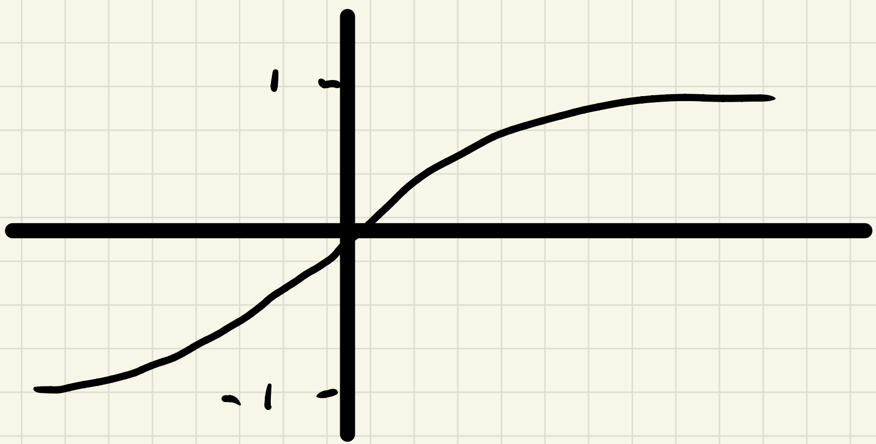
$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

B Funciones de activación

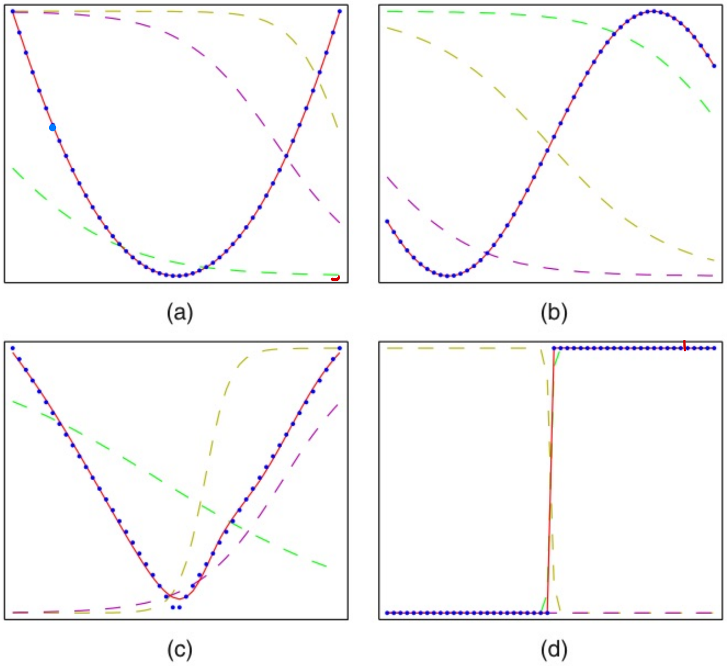
- sigmoid (logística)
- ReLU
- Tanh

$\left.\begin{matrix}\text{ReLU}\\ \text{Tanh}\end{matrix}\right\}$ común en capas ocultas



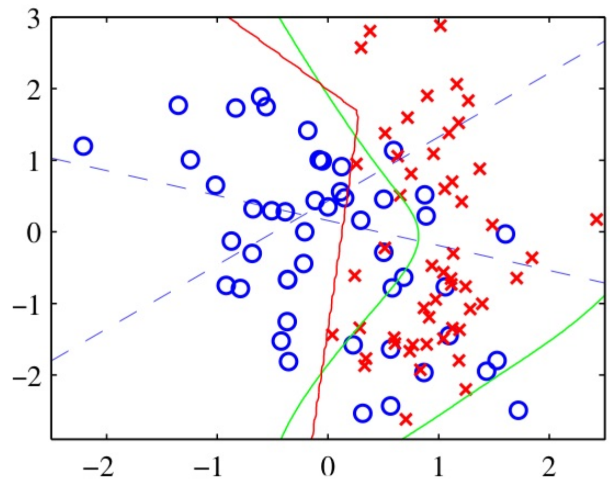
$$\frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Ejemplos aproximación



- Capacidad de aproximación de una red (e.g., cuadrática, seno, valor absoluto y Heavside). Los datos son los 50 puntos azules. Se entrena una red con dos capas, tres neuronas, función de activación tanh, salidas lineales. Las salidas de las tres neuronas ocultas se muestran con líneas punteadas.

EJEMPLO
CLASIFICACION



- Capacidad de aproximación de una red: las lineas punteadas son las salidas de cada una de las dos neuronas (hipersuperficies). Funciones de activación tanh y salida logistica sigmoid.
- La línea verde es el clasificador Bayesiano. La roja el clasificador de la red.

④ INTERPRETACIÓN PROBABILÍSTICA DE LAS SALIDAS DE UNA RED

- nos permite elegir mejor las funciones de activación
- da otra interpretación de la función de pérdida y costo

① REGRESIÓN

en un problema de regresión supongamos que $P(y \mid x, w) = N(y \mid \hat{y}(x, w), \beta^{-1})$ donde β es la varianza inversa (precisión) de la distribución

Gaussiana:

$$P(y \mid x, w) = \frac{1}{(2\pi\sigma)^{1/2}} e^{-\frac{1}{2\sigma}(y - \hat{y}(x,w))^2}$$

si $\beta = 1/\sigma$

$$\Rightarrow P(y \mid x, w) = \left(\frac{\beta}{2\pi}\right)^{1/2} e^{-\frac{\beta}{2}(y - \hat{y}(x,w))^2}$$

suponiendo independencia de y condicional a x, la log verosimilitud es:

$$\Rightarrow \mathcal{L}(w, \beta) = \frac{1}{2} m \ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2} \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)}(x^{(i)}, w))^2$$

- MAXIMIZAR VEROSIMILITUD ES EQUIVALENTE A MINIMIZAR PÉRDIDA CUADRÁTICA

2 CLASIFICACIÓN BINARIA

- si la salida es binaria cero o uno y la red genera es una probabilidad con una función sigmoid:

$$P(y \mid x, w) = \hat{y}(x,w)^{y} (1 - \hat{y}(x,w))^{1-y}$$

- suponiendo independencia de y condicional a x, la log verosimilitud es:

$$J(w) = \frac{1}{m}\sum_{i=1}^{m}\Big[y^{(i)}\ln(\hat{y}(x^{(i)}, w)) + (1-y^{(i)})\ln(1-\hat{y}(x^{(i)}, w))\Big]$$

③ CLASIFICACIÓN MULTICLASE

$y \in \{1, \dots k\}$

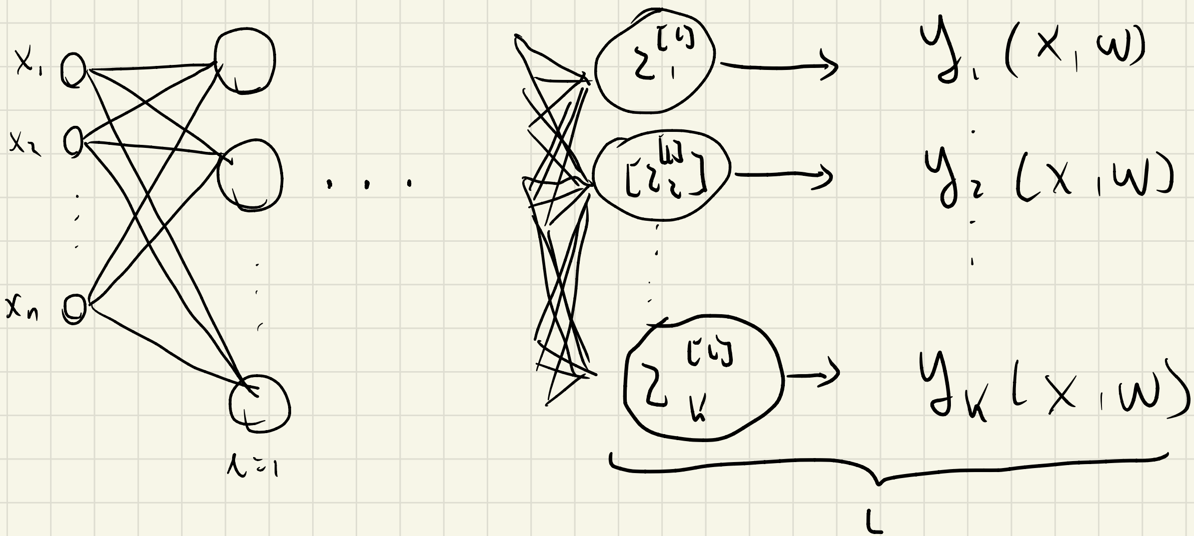
suponiendo la salida de la red es

$p(y_k = 1 \mid w)$ por ejemplo cuando

la función activación es:

SOFTMAX OUTPUT ACTIVATION :

$$y_k(x,w) = p(y_k = 1 \mid x) = \frac{e^{z_k^{[L]}}}{\sum_{i=1}^{K} e^{z_i^{[L]}}}$$



Observese que la ultima capa tiene una forma especial.
No es la misma función de activación en cada neurona.

se puede interpretar como una neurona final que recibe un vector de input $z^{[L]} = (z_1^{[L]}, \dots, z_K^{[L]})$
y la salida es un vector de probabilidades:
$y = (y_1, \dots, y_K)$

- en este caso con las mismas hipótesis anteriores el log de la función de verosimilitud es:

$$\ell(w) = \sum_{i=1}^{m} \sum_{k=1}^{K} y_{kn} \ln(\hat{y}_k(x^{(i)}, w))$$

(4) TEORIA DE LA INFORMACION

- Likely events should have low information content, and in the extreme case, events that are guaranteed to happen should have no information content whatsoever.
- Less likely events should have higher information content.
- Independent events should have additive information. For example, finding out that a tossed coin has come up as heads twice should convey twice as much information as finding out that a tossed coin has come up as heads once.

- MOTIVACION PARA DISTRIBUCIONES DISCRETAS: LA información de observar x es $I(x) = -\ln P(x)$. en unidades de nats. un nat es la información que tiene un evento con prob. $1/e$. Si usamos $\log_2$ es unidades de bits. un bit es la información de un evento que sucede con prob. $1/2$. ESTA DEFINICION no se extiende a varb. continuas.

- <u>ENTROPIA DE SHANNON</u> (discretas o continuas)

  LA entropia de P es : $H(p) = E_p[-\ln(p(x))]$

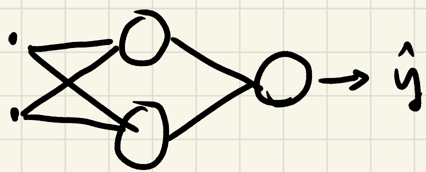
- LA DIVERGENCIA KL permite interpretar la maximizacion de verosimilitud como la minimizacion de la divergencia de KL entre la verdadera distribucion y la distrib empirica.

⑤ INICIALIZACIÓN PARÁMETROS

- En esta parte demostramos porque inicializar con ceros no es una buena idea para redes con $L \geq 2$.

caso logístico funciona bien pero no en general

- Considere esta red:



$$w^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad W^{[2]} = [0 \ 0]$$

$$b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad b^{[2]} = [0]$$

INICIALIZACIÓN

$$\Rightarrow z^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad a^{[1]} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

$$z^{[2]} = [0], \quad a^{[2]} = [1/2]$$

FORWARD PROP.

BACKWARD

$db^{[2]} = dz^{[2]} = 1/2 - y$, $dW^2 = [\,1/2(1/2-y) \quad 1/2(1/2-y)\,]^T$

$db^{[1]} = dz^{[1]} = [0 \quad 0]^T$, $dW^1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

Obsérvese que:

$dz_1^{[1]} = dz_2^{[1]}$

$dW^1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \text{———} \\ \text{———} \end{bmatrix} = \begin{bmatrix} v \rightarrow \\ v \rightarrow \end{bmatrix}$ filas iguales

$W^1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} v \\ v \end{bmatrix} \Rightarrow$ $\boxed{\text{Filas iguales}}$

BACKWARD:

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

- **PRIMERA ACTUALIZACIÓN DE PARÁMETROS**

$$W^{[1]} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} - \alpha \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}, \quad b^{[1]} = \begin{bmatrix}0\\ 0\end{bmatrix}$$

$$W^{[2]} = [0 \; 0] - \alpha \frac{(1/2 - y)}{2}[1 \; 1], \quad b^{[2]} = \left[-\frac{\alpha}{2}(1/2 - y)\right]$$

- Obsérvese que:

$$W^{[1]} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}, \quad b^{[1]} = \begin{bmatrix}0\\ 0\end{bmatrix}$$

$$W^{[2]} = [v' \; v], \quad b^{[2]} = [v']$$

- **Forward Prop (segunda iteración)**

$$z^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad a^{[1]} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

$$z^{[2]} = \frac{1}{2} w' + \frac{1}{2} w' = w', \quad a^{[2]} = \frac{1}{1+e^{-w'}}$$

- **BACKWARD PROP (segunda iteración)**

$$dz_1^{[1]} = dz_2^{[1]}$$

$$dW^{[1]} = \begin{bmatrix} dz_1^{[1]} \\ dz_2^{[1]} \end{bmatrix} [x_1 \quad x_2] = \begin{bmatrix} dz_1^{[1]} x_1 & dz_1^{[1]} x_2 \\ dz_2^{[1]} x_1 & dz_2^{[1]} x_2 \end{bmatrix} = \begin{bmatrix} w'' \\ w'' \end{bmatrix}$$ filas iguales

- **SEGUNDA ACTUALIZACIÓN DE PARÁMETROS**

$$W^{[1]} = \begin{bmatrix} — \\ — \end{bmatrix}$$ $\boxed{\text{filas iguales}}$

- Si continuamos iterando (más generalmente usando inducción) obtenemos que:

$$W^{[1]} = \begin{bmatrix} r \longrightarrow \\ r \longrightarrow \end{bmatrix} \rightsquigarrow \boxed{\text{Filas iguales}}$$

Luego:

$$z^{[1]} = \begin{bmatrix} r \rightarrow \\ r \rightarrow \end{bmatrix} = \begin{bmatrix} r \cdot [x_1, x_2] \\ r \cdot [x_1, x_2] \end{bmatrix} \Rightarrow z_1^{[1]} = z_2^{[1]}$$

$\Rightarrow a^{[1]}(z_1^{[1]}) = a^{[1]}(z_2^{[1]})$ en todas las iteraciones: las dos unidades están haciendo lo mismo (simétricas). Más unidades no hacen nada

⑥ RED NEURONAL GENERAL

- FORWARD PROPAGATION

$$z^{[k](i)} = W^{[k]} a^{[k-1](i)} + b^{[k]}$$

$$a^{[k](i)} = g^{[k]}\left(z^{[k](i)}\right)$$

donde:

$$a^{[0](i)} = x^{i}$$
$$a^{[L](i)} = \hat{y}^{(i)}$$

- **BACKWARD PROPAGATION**: implementación recursiva de la regla de la cadena

- Cálculos corresponden a capa de salida y activación sigmoid

→ REG LOGÍSTICA

$$\frac{\partial J}{\partial w_1}(w,b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial y}{\partial a^{(i)}} \cdot \frac{\partial a^{(i)}}{\partial z^{(i)}} \cdot \frac{\partial z^{(i)}}{\partial w_1}$$

$$\frac{\partial y}{\partial a^{(i)}} = \frac{a^{(i)} - y^{(i)}}{a^{(i)}(1-a^{(i)})}$$

$$\frac{\partial a^{(i)}}{\partial z^{(i)}} = (a^{(i)})^2 e^{-z^{(i)}} = a^{(i)}(1-a^{(i)})$$

$$\frac{\partial z^{(i)}}{\partial w_1} = x^{(i)}$$

$$\frac{\partial y}{\partial b} = \frac{\partial y}{\partial a^{(i)}} \cdot \frac{\partial a^{(i)}}{\partial z^{(i)}}$$

CAPA L

$$dz^{[L](i)} = \frac{1}{m}\sum_{i=1}^{m}\left(a^{[L](i)} - y^{(i)}\right)$$

$$dW^{[L]} = \frac{1}{m}\sum_{i=1}^{m} dz^{[L](i)}\, a^{[L-1](i)}$$

$$\boxed{z^{[L](i)} = W^{[L]} a^{[L-1](i)} + b^{[L]}}$$

$$db^{[L]} = \frac{1}{m}\sum_{i=1}^{m} dz^{[L](i)}$$

- CAPA $1 \leq l \leq L-1$, $l = k$

$$z^{[k](i)} = W^{[k]} a^{[k-1](i)} + b^{[k]}$$

$$a^{[k](i)} = g^{[k]}(z^{[k](i)})$$

$$\Rightarrow \frac{\partial a^{[k]}}{\partial w_j^{[k]}} = \frac{\partial g^{[k]}}{\partial z}(z^{[k]})\, a_j^{[k-1]} = dw_j^{[k]}$$

$$\Rightarrow \frac{\partial a^{[k]}}{\partial b_j} = \underbrace{\frac{\partial g^{[k]}}{\partial z}(z^{[k]})}_{dz^{[k]}}$$