Clasificación Supervisada

Redes Neuronales

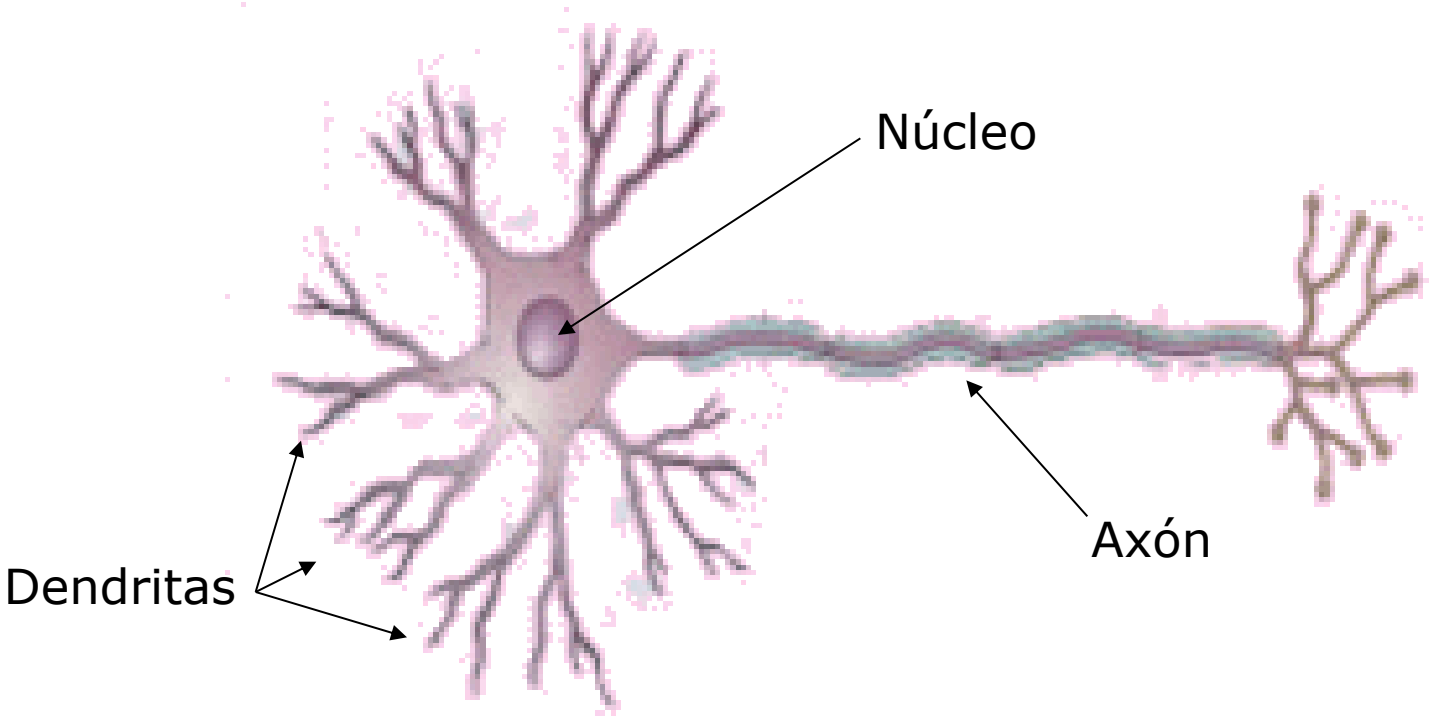
Jesús Ariel Carrasco Ochoa
Instituto Nacional de Astrofísica, Óptica y Electrónica

Redes Neuronales

- Idea central: simular el funcionamiento del cerebro, el cual es un conjunto grande de neuronas interconectadas
- La unidad básica es la neurona
- La idea inicial es tener una neurona artificial
- Posteriormente tener un cierto número de neuronas artificiales conectadas formando una red neuronal artificial

Redes Neuronales

- Neurona real



Redes Neuronales

- El núcleo está en el cuerpo celular y controla el funcionamiento de la neurona
- Las dendritas son conexiones de entrada
- El axón es una conexión de salida
- Las conexiones entre dendritas y axones se llamas sinapsis

Redes Neuronales

- Cuando las dendritas reciben un impulso de un axón, pueden estimular o inhibir la producción e una salida
- Una neurona no produce una salida a menos que los estímulos que recibe a través de sus dendritas alcance cierto umbral
- En este sentido las neuronas son dispositivos de todo o nada

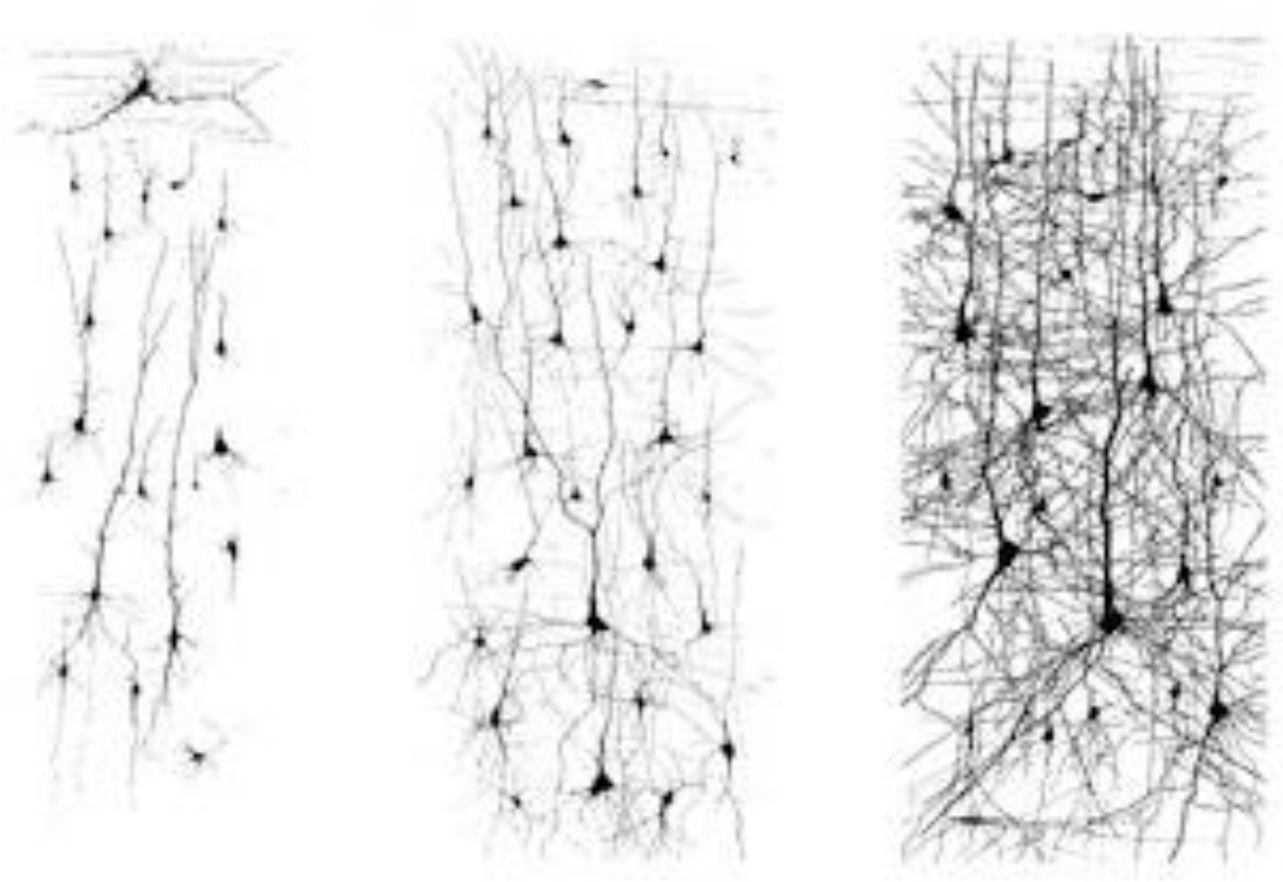
Redes Neuronales

- El cerebro humano tiene aproximadamente 10^{11} neuronas
- Cada neurona tiene hasta 10^5 sinapsis
- En el cerebro humano hay aproximadamente 10^{16} sinapsis
- $10^{16} \approx 2^{53} = 2^{13}2^{40} = 2^{23}2^{30}$

  $= 8192$ Teras $= 8388608$ Gigas

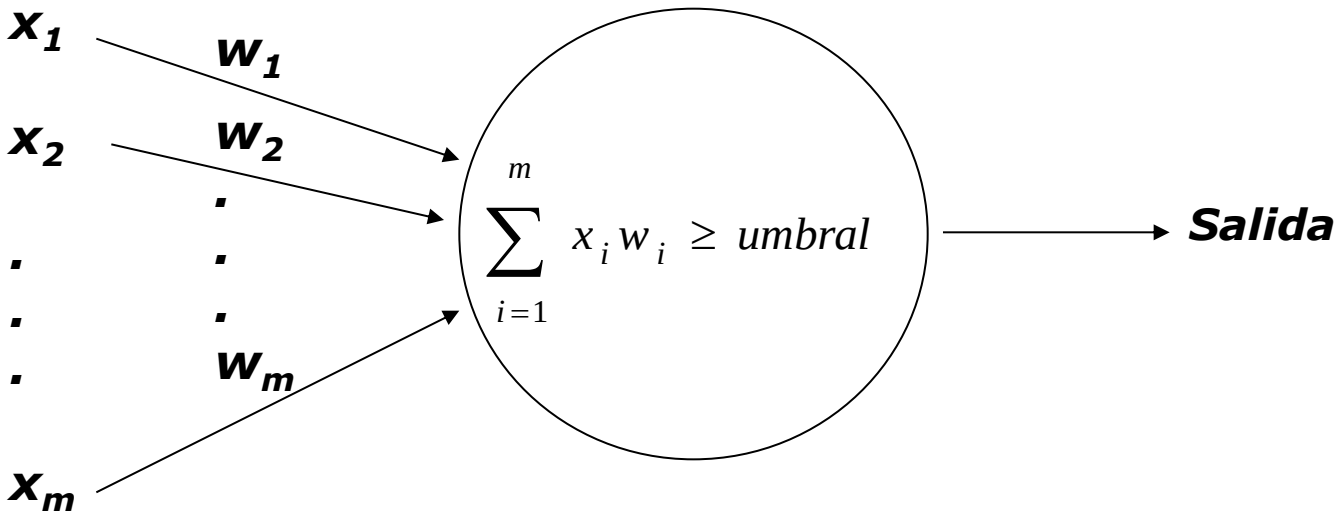
Redes Neuronales

- Una red neuronal real se vería como



Redes Neuronales

- Neurona Artificial



Redes Neuronales

- Si

$$\sum_{i=1}^{m} x_i w_i \geq umbral$$

  - entonces se produce una ***salida***=1
  - En caso contrario se produce una ***salida***=0

- como

$$\sum_{i=1}^{m} x_i w_i \geq umbral \quad \text{es equivalente a} \quad \sum_{i=1}^{m} x_i w_i - umbral \geq 0$$

Redes Neuronales

- Para simplificar se agrega una entrada con valor fijo 1 y se agrega un peso w_0=-***umbral***, con lo que

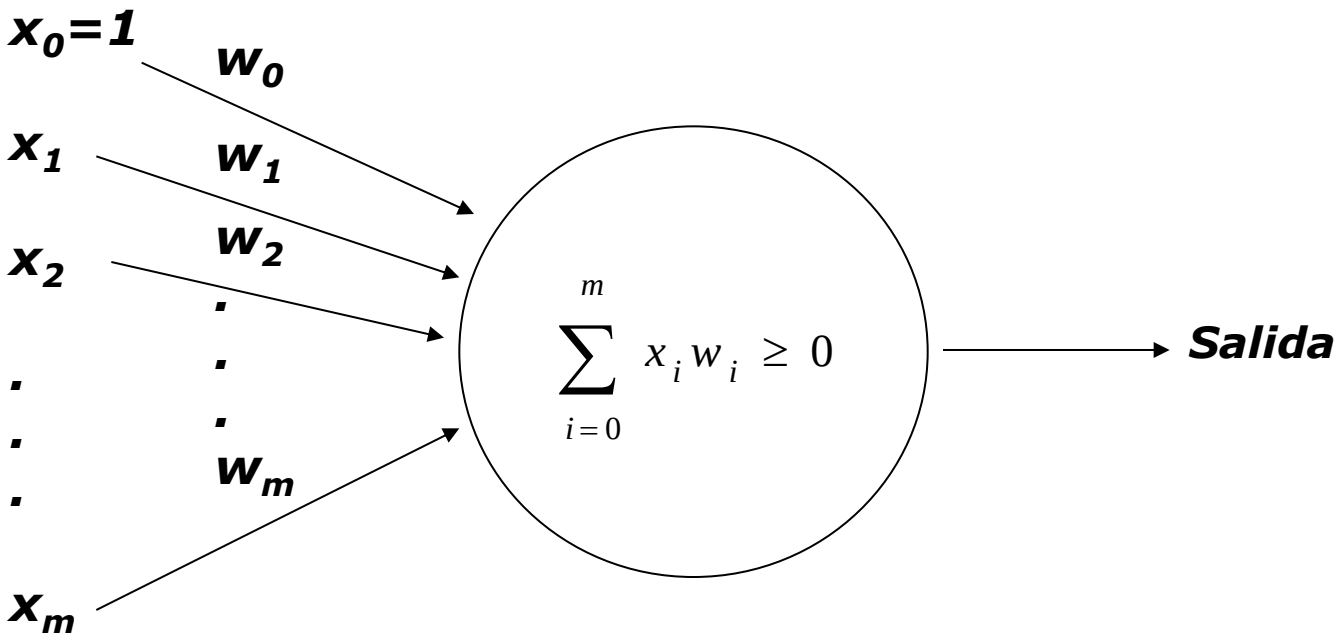
$$\sum_{i=1}^{m} x_i w_i - umbral \geq 0$$

$$\left(\sum_{i=1}^{m} x_i w_i \right) + w_0 \geq 0$$

$$\sum_{i=0}^{m} x_i w_i \geq 0 \qquad \text{con} \quad x_0 = 1$$

Redes Neuronales

- Neurona Artificial



Redes Neuronales

- Una Red Neuronal Artificial es un conjunto de neuronas artificiales interconectadas

- Entrenar la red consiste en encontrar los valores adecuados para los pesos $\boldsymbol{w_i}$ de cada neurona, tal que la salida final sea la deseada

Redes Neuronales

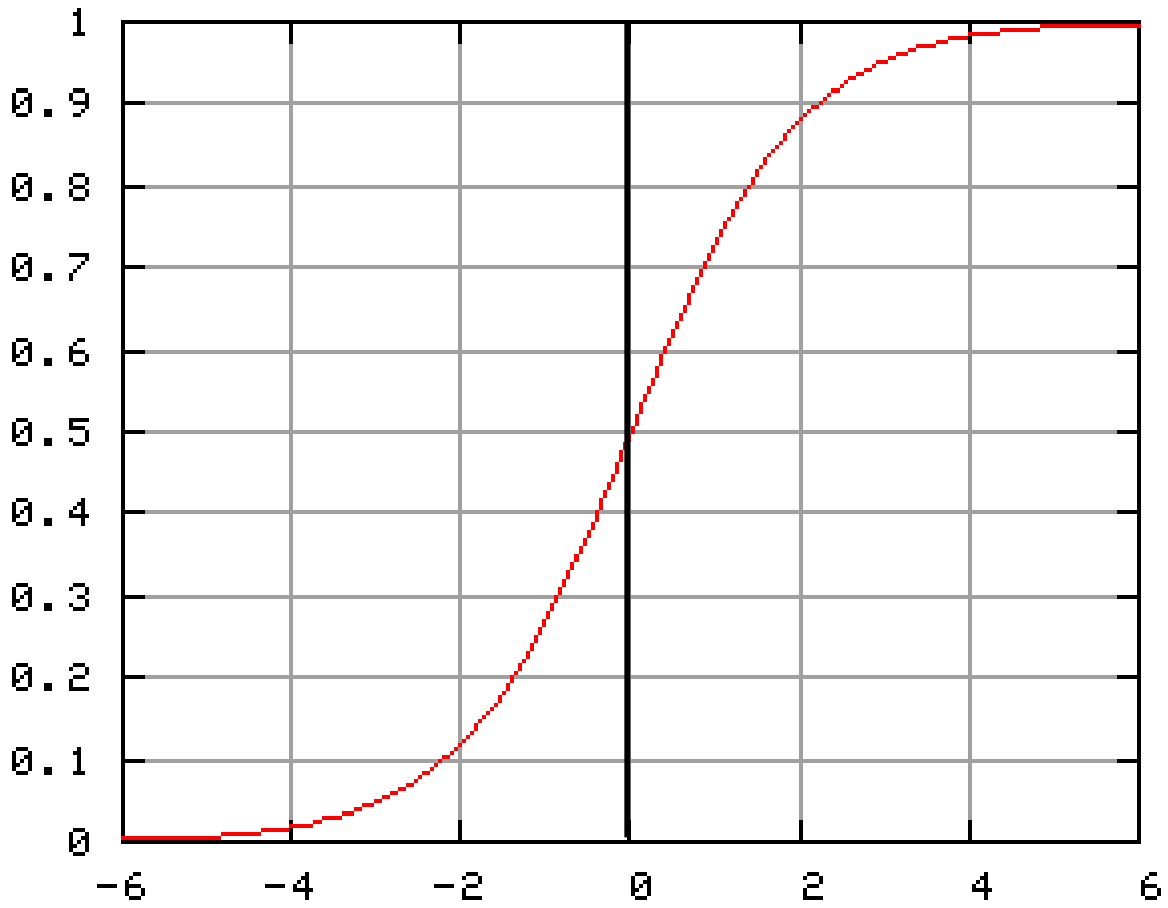
- Para generalizar al caso de salidas reales se puede tomar

$$salida = f\left(\sum_{i=1}^{m} x_i w_i\right)$$

- Donde $f: \Re \rightarrow \Re$

- Comúnmente se usa una sigmoidal

Redes Neuronales



Redes Neuronales

- Comúnmente

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

- Esta función cumple que

$$\text{cuando} \quad x \to \infty, \quad e^{-x} \to 0, \quad f(x) \to 1$$

$$x \to -\infty, \quad e^{-x} \to \infty, \quad f(x) \to 0$$

Redes Neuronales

- Además

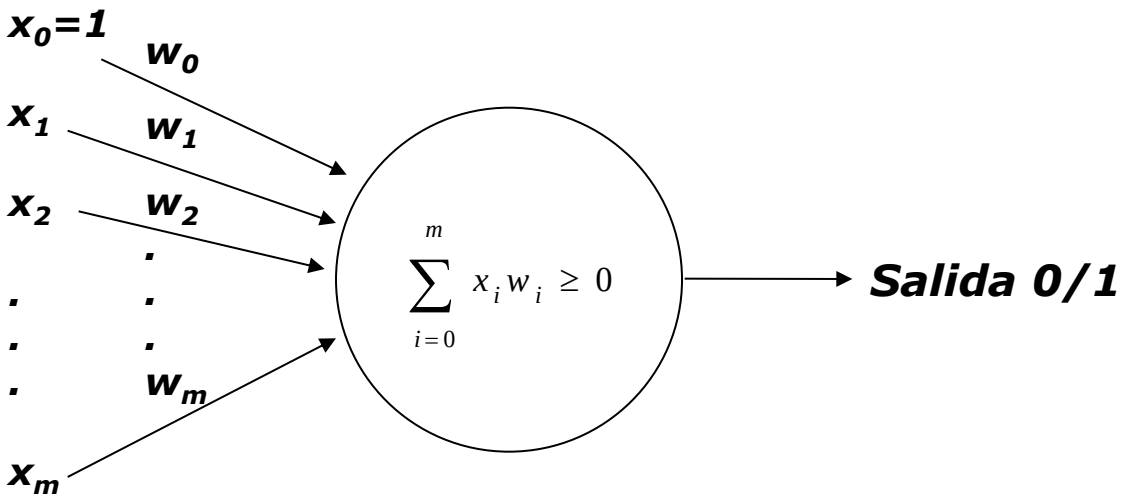
$$f'(x) = \left(\frac{1}{1+e^{-x}} \right)' = -\frac{1}{\left(1+e^{-x}\right)^2}\left(1+e^{-x}\right)' = -\frac{1}{\left(1+e^{-x}\right)^2}\left(-e^{-x}\right)$$

$$= \frac{e^{-x}}{\left(1+e^{-x}\right)^2} = \frac{\left(1+e^{-x}\right)-1}{\left(1+e^{-x}\right)^2} = \frac{1}{\left(1+e^{-x}\right)} - \frac{1}{\left(1+e^{-x}\right)^2}$$

$$= \frac{1}{\left(1+e^{-x}\right)}\left(1-\frac{1}{\left(1+e^{-x}\right)}\right) = \underline{\underline{f(x)\left(1-f(x)\right)}}$$

Perceptrón (Frank Rosenblatt, 1958)

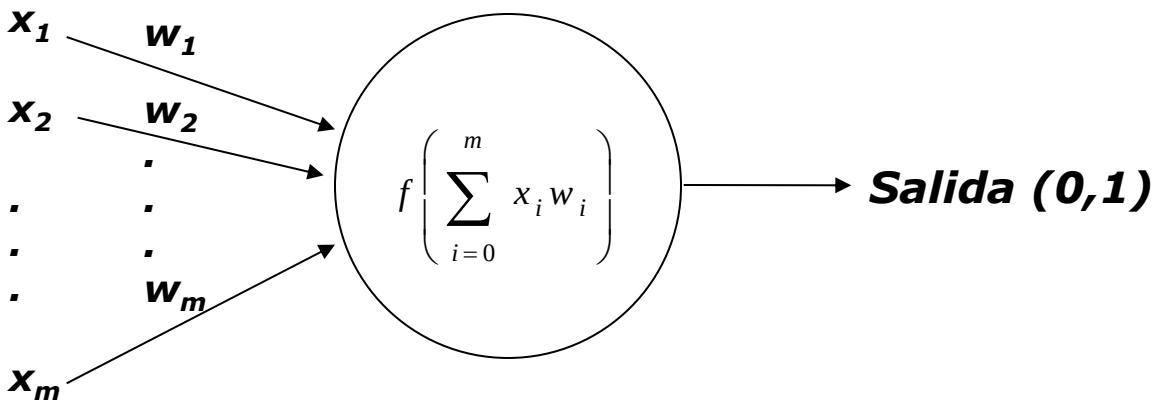
- Perceptrón (Booleano)



- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

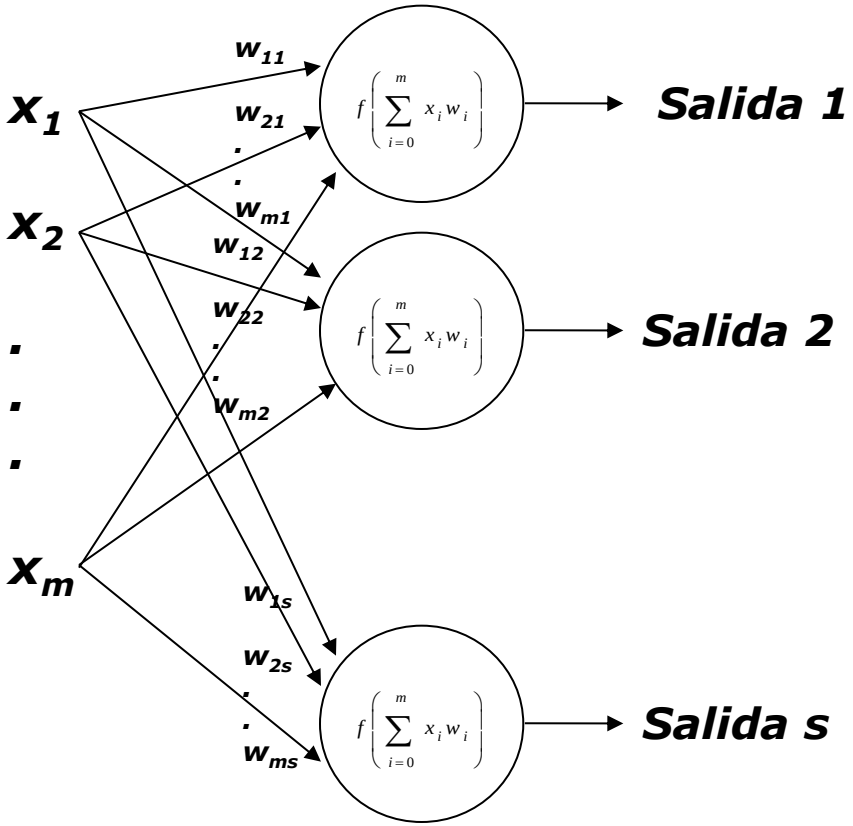
Perceptrón

- Perceptrón (continuo)



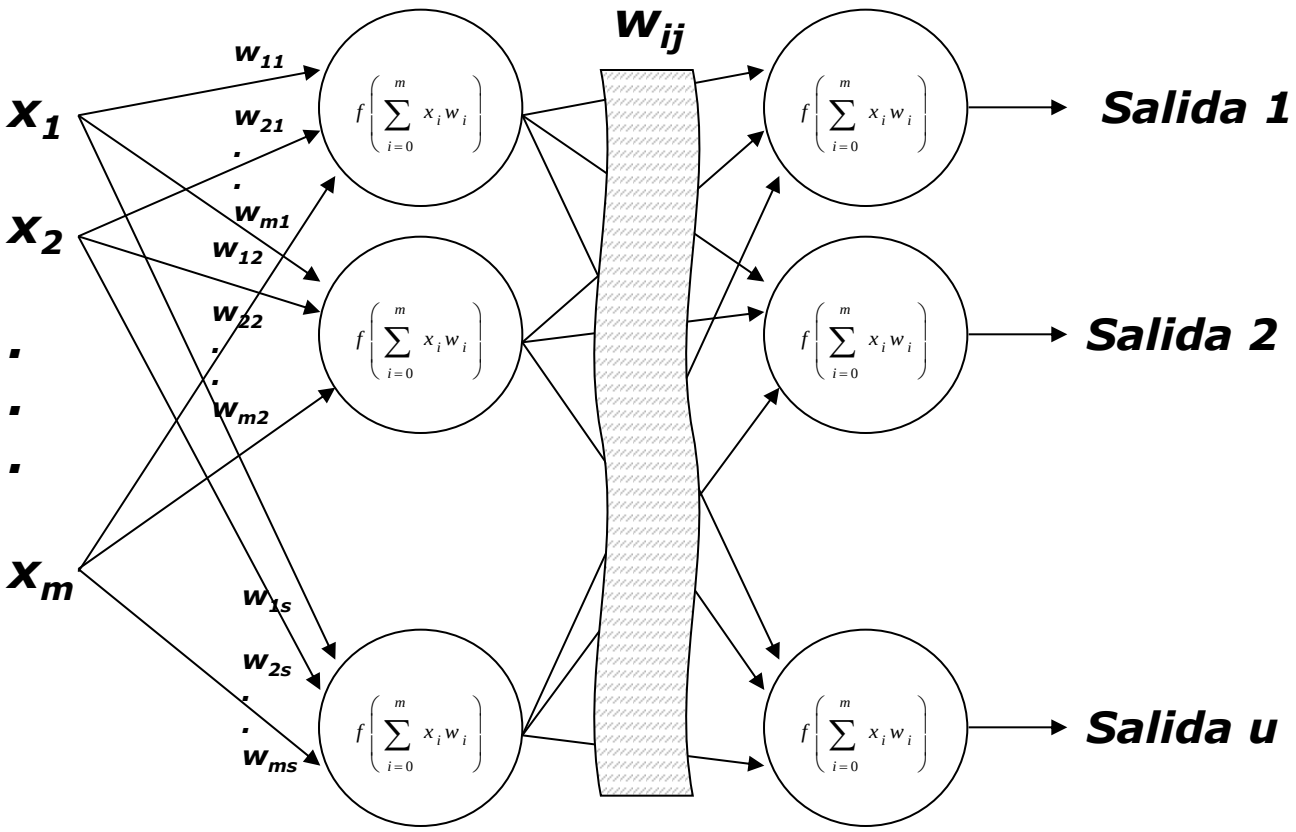
Redes Neuronales

- Red de una sola capa de Perceptrones



Redes Neuronales

- Perceptrón multicapa

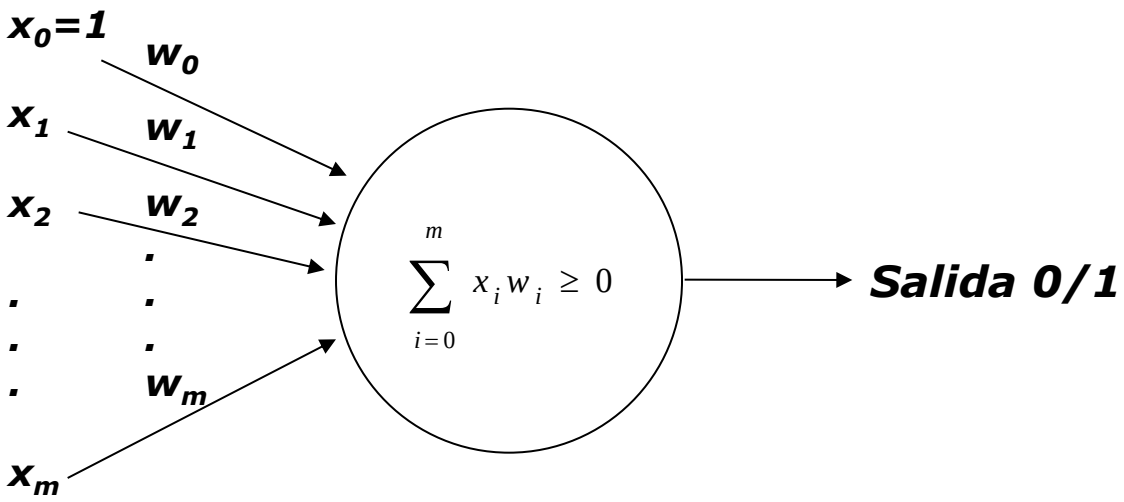


Como entrenar un Perceptrón

- Si se tienen ***n*** objetos $O_1, O_2, \ldots, O_n$
- Separados en dos clases C_0 y C_1
- Descritos por ***m*** atributos Booleanos $x_1, x_2, \ldots, x_m$

Como entrenar un Perceptrón

- El perceptrón tiene la forma



Como entrenar un Perceptrón

- Si una entrada $x_1, x_2, \ldots, x_m$ produce 0 y se esperaba 1 hay que aumentar los valores de los pesos que multiplican a las entradas con valor 1

- Si una entrada $x_1, x_2, \ldots, x_m$ produce 1 y se esperaba 0 hay que reducir los valores de los pesos que multiplican a las entradas con valor 1

Como entrenar un Perceptrón

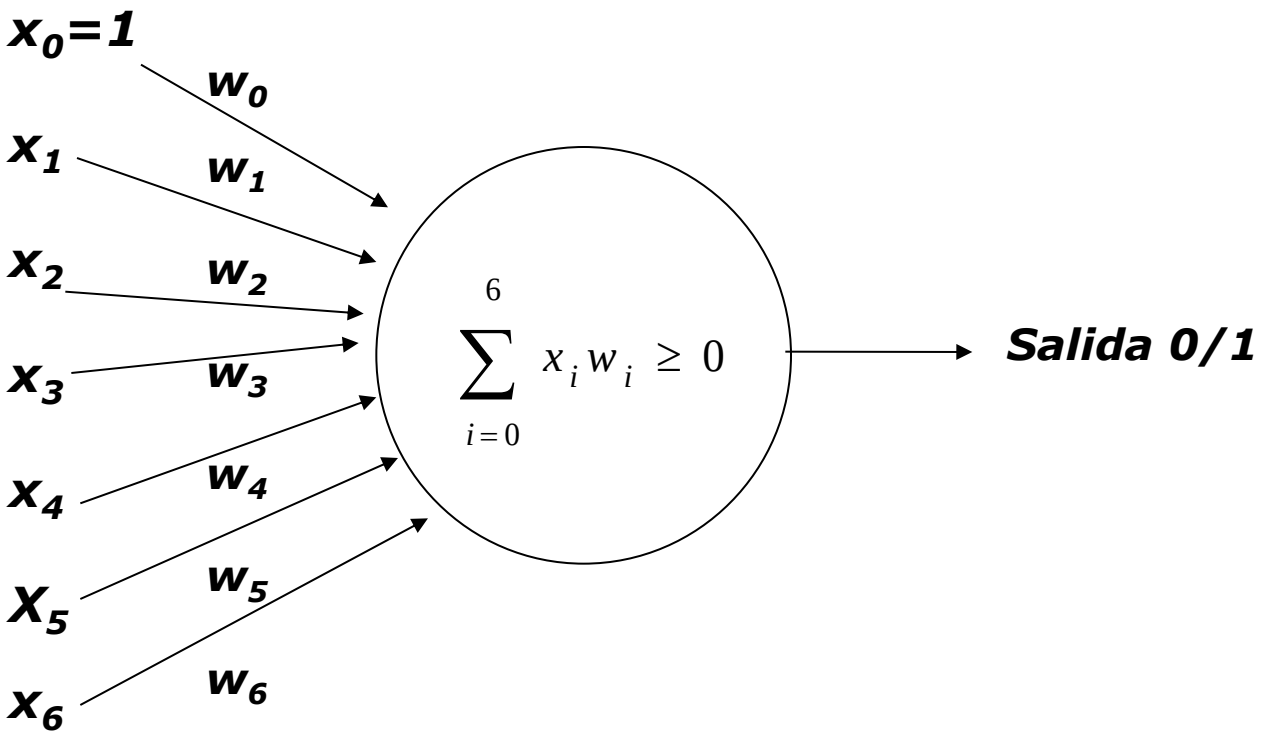
1. Inicializar $w_i=0$, $i=0,\ldots,m$
2. Si los pesos w_i producen una salida correcta para toda la muestra → FIN
3. Seleccionar una entrada para la que se produzca una salida incorrecta
4. Si la salida producida es 0 tomar
   $w_i=w_i+x_i$ para $i=0,\ldots,m$
5. Si la salida producida es 1 tomar
   $w_i=w_i-x_i$ para $i=0,\ldots,m$
6. Ir a 2

Como entrenar un Perceptrón

- Suponemos que se tiene una muestra de entrenamiento como sigue

T	x_1	x_2	x_3	x_4	x_5	x_6	**Salida**
O_1	1	0	0	0	1	1	0
O_2	0	1	0	0	0	1	0
O_3	0	0	1	0	1	0	1
O_4	0	1	1	1	0	0	1

Como entrenar un Perceptrón



Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S	S_0	S_1	S_2
O_1	1	1	0	0	0	1	1	0			
O_2	1	0	1	0	0	0	1	0			
O_3	1	0	0	1	0	1	0	1			
O_4	1	0	1	1	1	0	0	1			

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2								
3								
4								

Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S	S_0	S_1	S_2
O_1	1	1	0	0	0	1	1	0	1		
O_2	1	0	1	0	0	0	1	0	1		
O_3	1	0	0	1	0	1	0	1	1		
O_4	1	0	1	1	1	0	0	1	1		

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2								
3								
4								

Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S	S_0	S_1	S_2
O_1	1	1	0	0	0	1	1	0	1		
O_2	1	0	1	0	0	0	1	0	1		
O_3	1	0	0	1	0	1	0	1	1		
O_4	1	0	1	1	1	0	0	1	1		

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2	-1	-1	0	0	0	-1	-1	$-x_1$
3								
4								

Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S	S_0	S_1	S_2
O_1	1	1	0	0	0	1	1	0	1	0	
O_2	1	0	1	0	0	0	1	0	1	0	
O_3	1	0	0	1	0	1	0	1	1	0	
O_4	1	0	1	1	1	0	0	1	1	0	

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2	-1	-1	0	0	0	-1	-1	$-x_1$
3								
4								

Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S	S_0	S_1	S_2
O_1	1	1	0	0	0	1	1	0	1	0	
O_2	1	0	1	0	0	0	1	0	1	0	
O_3	1	0	0	1	0	1	0	1	1	0	
O_4	1	0	1	1	1	0	0	1	1	0	

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2	-1	-1	0	0	0	-1	-1	$-x_1$
3	0	-1	0	1	0	0	-1	$+x_3$
4								

Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S	S_0	S_1	S_2
O_1	1	1	0	0	0	1	1	0	1	0	0
O_2	1	0	1	0	0	0	1	0	1	0	0
O_3	1	0	0	1	0	1	0	1	1	0	1
O_4	1	0	1	1	1	0	0	1	1	0	1

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2	-1	-1	0	0	0	-1	-1	$-x_1$
3	0	-1	0	1	0	0	-1	$+x_3$
4								

Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S	S_0	S_1	S_2
O_1	1	1	0	0	0	1	1	0	1	0	0
O_2	1	0	1	0	0	0	1	0	1	0	0
O_3	1	0	0	1	0	1	0	1	1	0	1
O_4	1	0	1	1	1	0	0	1	1	0	1

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2	-1	-1	0	0	0	-1	-1	$-x_1$
3	0	-1	0	1	0	0	-1	$+x_3$
4								

Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S
O_a	1	1	1	0	0	1	1	0
O_b	1	0	1	1	1	1	0	1
O_c	1	1	0	1	0	1	1	0

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2	-1	-1	0	0	0	-1	-1	$-x_1$
3	0	-1	0	1	0	0	-1	$+x_3$
4								

Como entrenar un Perceptrón

T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S
O_a	1	1	1	0	0	1	1	0
O_b	1	0	1	1	1	1	0	1
O_c	1	1	0	1	0	1	1	0

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2	-1	-1	0	0	0	-1	-1	$-x_1$
3	0	-1	0	1	0	0	-1	$+x_3$
4								

Como entrenar un Perceptrón

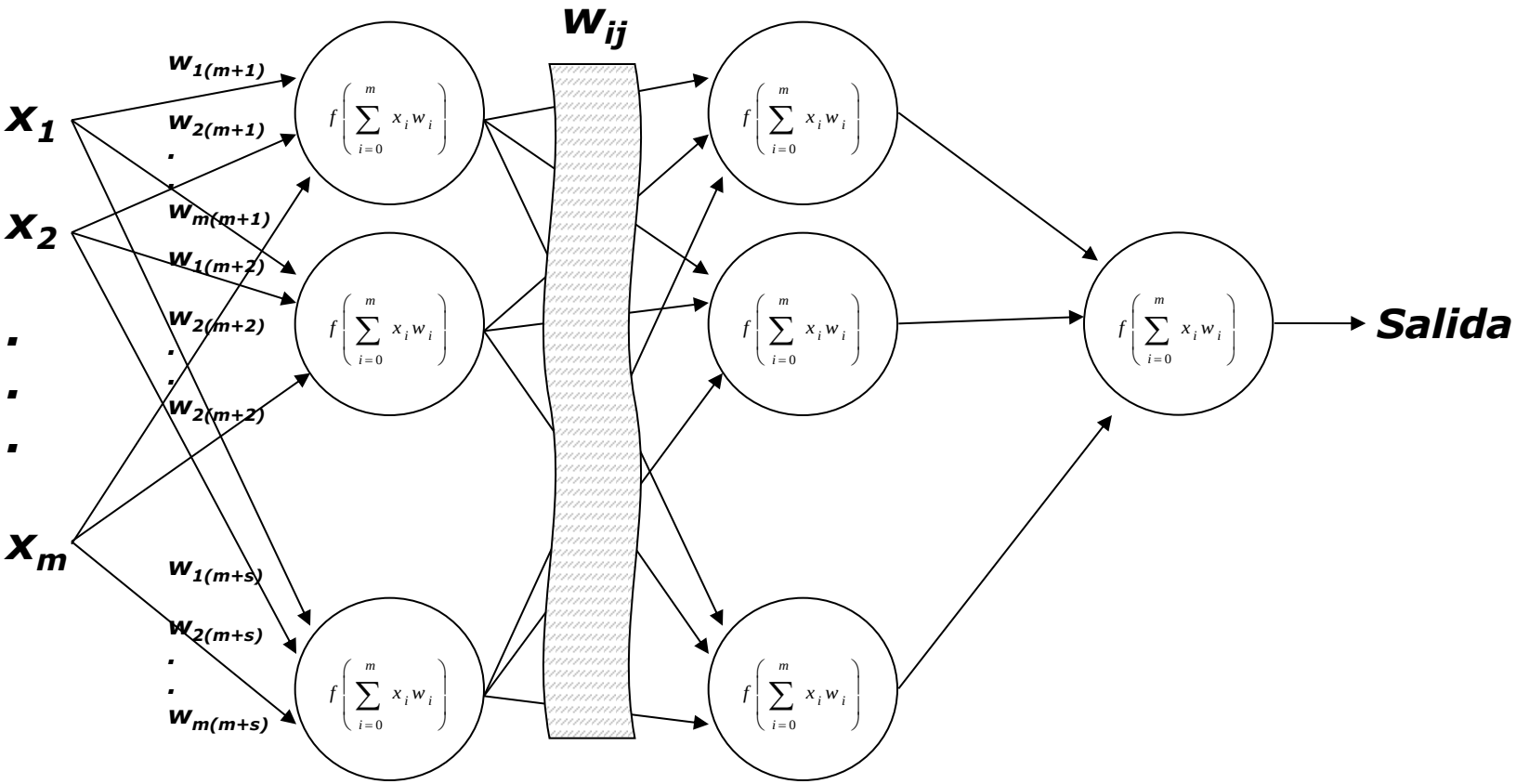
T	x_0	x_1	x_2	x_3	x_4	x_5	x_6	S
O_a	1	1	1	0	0	1	1	0
O_b	1	0	1	1	1	1	0	1
O_c	1	1	0	1	0	1	1	0

i	w_0	w_1	w_2	w_3	w_4	w_5	w_6	
1	0	0	0	0	0	0	0	
2	-1	-1	0	0	0	-1	-1	$-x_1$
3	0	-1	0	1	0	0	-1	$+x_3$
4								

Como entrenar un Perceptrón Multicapa

- Si se tiene una muestra T con n objetos $O_1, O_2, \ldots, O_n$
- Descritos por m atributos reales $x_1, x_2, \ldots, x_m$
- Cada objeto O_q tiene asociada la salida deseada D_q y produce la salida S_q, $q=1,\ldots,n$
- $D_q, S_q \in [0,1]$, $q=1,\ldots,n$

Como entrenar un Perceptrón Multicapa



Como entrenar un Perceptrón Multicapa

- Como evaluarlo?
- Se evalúa por capas
- Con las entrada se evalúa la capa 1, con las salidas de la capa 1 se evalúa la capa 2, y así sucesivamente hasta evaluar la última neurona que produce la salida de la red

Como entrenar un Perceptrón Multicapa

- Como entrenarlo?

- Retropropagación (backpropagation)

  - Werbos, P. (1974). Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

  - Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

Retropropagación (backpropagation)

- Se considera el siguiente funcional

$$P = -\sum_{O_q \in T} \left(D_q - S_q \right)^2$$

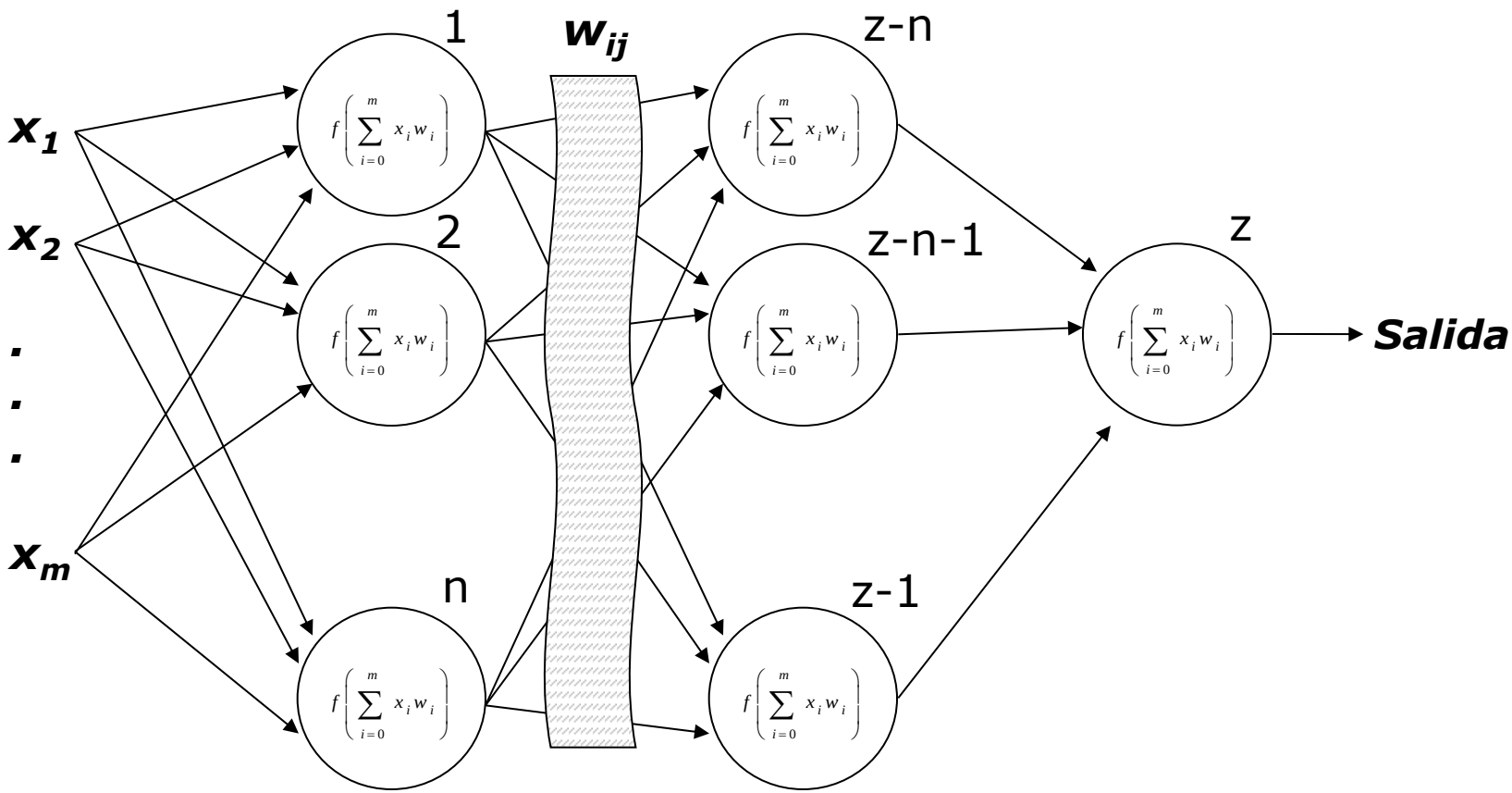
Salida deseada
Para $\boldsymbol{O_q}$

Salida observada
Para $\boldsymbol{O_q}$

Retropropagación (backpropagation)

- El valor máximo de P es 0
- El objetivo es maximizar P
- Se busca obtener los pesos w_{ij} que maximicen P
- Para esto se deriva P respecto a los w_{ij}
- Supondremos que se tienen z neuronas $N_1,\ldots,N_z$
- w_{ij} es el peso de la conexión de la N_i con N_j
- Sea o_j la salida de la neurona N_j
- Sea K_j el conjunto de los índices de las neuronas a las que se conecta la salida de la neurona N_j

Retropropagación (backpropagation)



Retropropagación (backpropagation)

1. Inicializar aleatoriamente los pesos w_{ij}
2. Determinar el parámetro de rapidez r
3. Si el desempeño es satisfactorio → FIN
4. Tomar para todo i,j: $\Delta_{ij} = 0$
5. Para cada $O_p \in T$
    5. Calcular la salida de la red S_p
    6. Tomar $\beta_z = D_p - S_p$
    7. Para $j = z-1,\ldots,1$
        8. Tomar $\beta_j = \sum_{k \in K_j} w_{jk} o_k (1 - o_k) \beta_k$
        9. Para toda N_i que se conecte a una entrada de N_j
            10. Tomar $\Delta_{ij} = \Delta_{ij} + r o_i o_j (1 - o_j) \beta_j$
11. Tomar para todo i,j: $w_{ij} = w_{ij} + \Delta_{ij}$
12. Ir a 3

Perceptrón Multicapa

- Intuitivamente si una red da buenos resultados, una red más grande debería dar mejores resultados.
- Esto no es cierto en todos los casos.
- Una red más grande tendrá más pesos para entrenar.
- Entre más variables libres se tengan más oscilación puede haber entre los puntos a entrenar.
- Heurística: El número de pesos por entrenar para cada salida debe ser menor que el número de muestras de entrada

Redes Neuronales

- Redes hacia adelante (feedforward)
- Redes recurrentes (feedback)
- Redes de Base Radial (RBFN)
- Redes de Hopfield (memorias auto-asociativas)
- Mapas Auto-organizables (SOM)
- Redes Morfológicas
- Redes Neuronales Celulares
- Redes Neuronales Convolucionales (CNN)
- Autoencoders
- Redes profundas
- etc.

Redes de Función de Base Radial

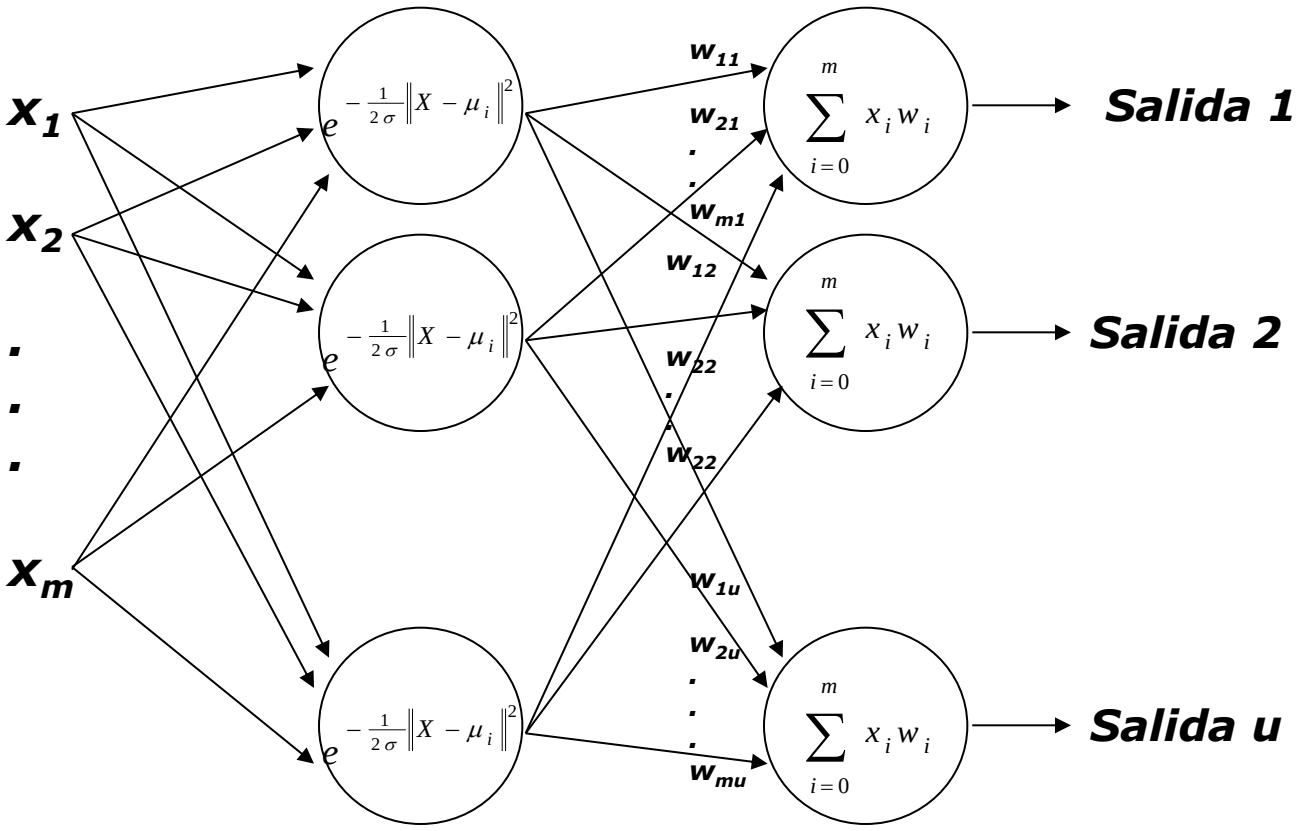
- Usualmente tienen solo 2 capas
- Las sinapsis de las entadas de la primera capa no tienen pesos asociados.
- La función de salida de las neuronas de la primera capa es una función de base radial, comúnmente una función tipo Gaussiana, usualmente

$$f_i(X) = e^{-\frac{1}{2\sigma}\left\|X-\mu_i\right\|^2}$$

- La función de salida de las neuronas de la última capa es solamente la suma de los pesos del producto de las entradas por los pesos asociados.
- De este modo la salida está dada por

$$Salida_j = \sum w_{ij}\, e^{-\frac{1}{2\sigma}\left\|X-\mu_i\right\|^2}$$

Redes de Función de Base Radial



Redes de Interpolación

- Es una red como la descrita
- La primera capa tiene una neurona por cada objeto en la muestra
- La media para la función de la neurona ***i*** de la capa de entrada es igual al objeto ***i*** de la muestra
- La varianza es un parámetro dado y es igual para todas las neuronas de la primera capa

Como entrenar una Red de Interpolación

- Si cada objeto O_j de la muestra está descrito por el vector $\boldsymbol{X_j}$ y se espera que produzca la salida $\boldsymbol{D_j}=(\boldsymbol{d_{j1}},\ldots,\boldsymbol{d_{ju}})$

- Para estimar los pesos de la salida $\boldsymbol{i}$ se forma el siguiente sistema de ecuaciones

$$f_1(X_1)w_{1i} + f_2(X_1)w_{2i} + \ldots + f_m(X_1)w_{mi} = d_{1i}$$

$$f_1(X_2)w_{1i} + f_2(X_2)w_{2i} + \ldots + f_m(X_2)w_{mi} = d_{2i}$$

$$\vdots$$

$$f_1(X_m)w_{1i} + f_2(X_m)w_{2i} + \ldots + f_m(X_m)w_{mi} = d_{mi}$$

Como entrenar una Red de Interpolación

- Se resuelve el sistema de ecuaciones para cada salida.
- Si son muchos objetos en la muestra esto puede ser muy costoso
- Para resolver esto se usan las redes de aproximación

Redes de Aproximación

- Se seleccionan solamente algunos objetos de la muestra
- Con estos objetos se construye una red de interpolación con los objetos seleccionados
- Se ajustan los pesos para los demás objetos de la muestra

Como entrenar una Red de Aproximación

- Se considera el siguiente funcional

$$P = -\sum_{O_q \in T} \left(D_q - S_q \right)^2$$

Salida deseada Para O_q

Salida observada Para O_q

Como entrenar una Red de Aproximación

1. Seleccionar algunos objetos de la muestra y crear con ellos una red de interpolación
2. Determinar el parámetro de rapidez ***r***
3. Si el desempeño es satisfactorio → FIN
4. Para cada neurona de salida ***j***=1,…,***u***
   5. Para cada peso w_{ij}
      6. Tomar $$w_{ij} = w_{ij} + r \sum_{p=1}^{n} (d_{pj} - S_{pj}) e^{-\frac{1}{2\sigma}\left\| X_p - \mu_i \right\|^2}$$
7. Ir a 3

Redes de Aproximación

- Para evitar la construcción de la red de interpolación
  - Inicializar los pesos aleatoriamente
  - Seguir el proceso de ajuste de los pesos
  - Ajustar al mismo tiempo las medias de las Gaussianas

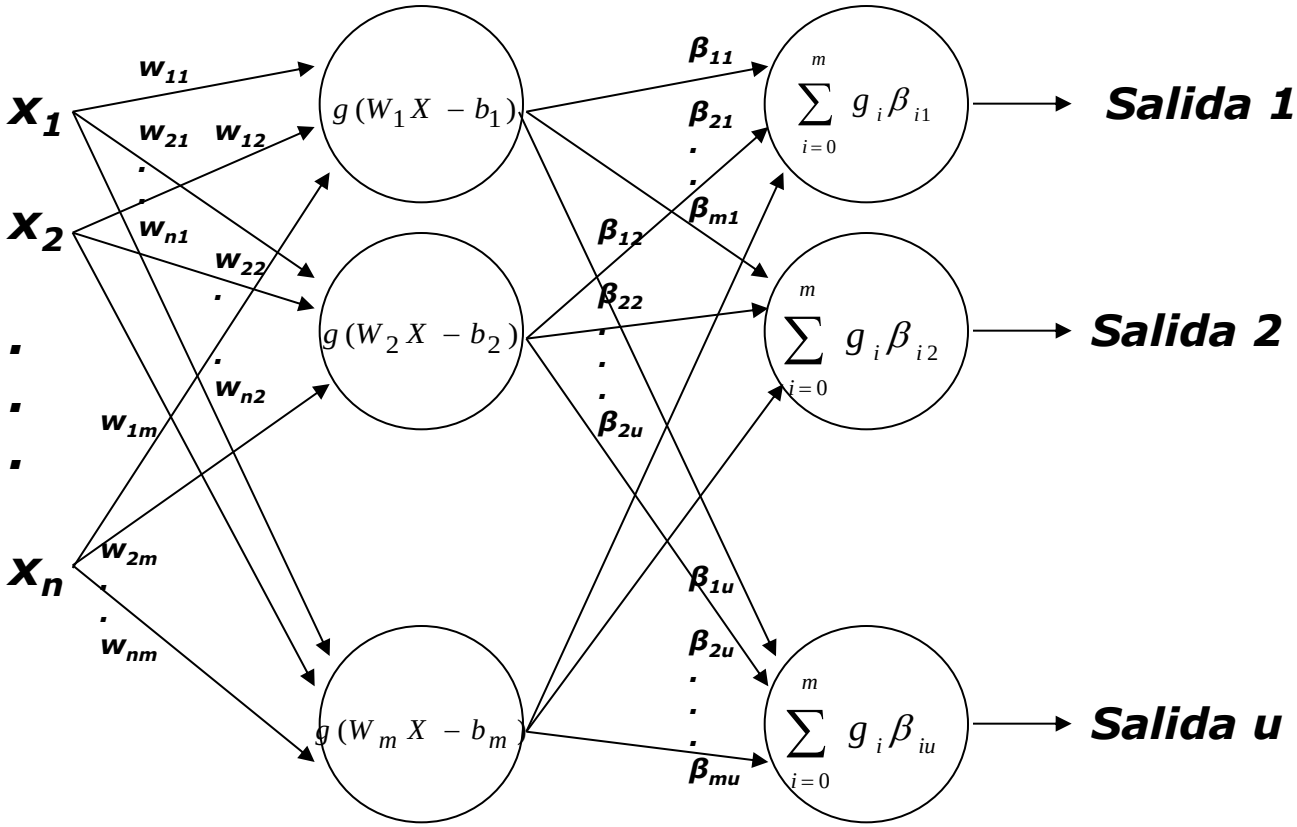
Como entrenar una Red de Aproximación

1. Inicializar aleatoriamente los pesos w_{ij} y las μ_i
2. Determinar el parámetro de rapidez r
3. Si el desempeño es satisfactorio → FIN
4. Para cada neurona de salida $j=1,\ldots,u$
   5. Para cada peso w_{ij}
      6. Tomar $\Delta w_{ij} = r \sum_{p=1}^{n} (d_{pj} - S_{pj}) e^{-\frac{1}{2\sigma}\left\| X_p - \mu_i \right\|^2}$
      7. Tomar $\Delta \mu_i = r \sum_{p=1}^{n} w_{ij} (d_{pj} - S_{pj}) e^{-\frac{1}{2\sigma}\left\| X_p - \mu_i \right\|^2} \frac{1}{\sigma} (X_p - \mu_i)$
      8. Tomar $w_{ij} = w_{ij} + \Delta w_{ij}$

         $\mu_i = \mu_i + \Delta \mu_i$
9. Ir a 3

Extreme Learning Machines (ELM)

ELM en esencia es un método para entrenar redes neuronales artificiales de la forma:



Extreme Learning Machines (ELM)

1. Inicializar aleatoriamente los pesos $\boldsymbol{w_{ik}}$ y los sesgos b_i (i=1,..,m;k=1,..,n)
2. Si cada objeto O_j (j=1,…,N) de la muestra está descrito por el vector $\boldsymbol{X_j}$ y se espera que produzca la salida $\boldsymbol{D_{j*}}$=($\boldsymbol{d_{j1}}$,…,$\boldsymbol{d_{ju}}$). Para cada salida i generar el sistema $A\,\beta_{*i}=D_{*i}$ como:

$$g(W_1 X_1 + b_1)\beta_{1i} + \ldots + g(W_m X_1 + b_m)\beta_{mi} = d_{1i}$$

$$g(W_1 X_2 + b_1)\beta_{1i} + \ldots + g(W_m X_2 + b_m)\beta_{mi} = d_{2i}$$

$$\vdots$$

$$g(W_1 X_N + b_1)\beta_{1i} + \ldots + g(W_m X_N + b_m)\beta_{mi} = d_{Ni}$$

3. Tomar $\beta_{*i}=A^{-}D_{*i}$
   A^{-} es la inversa generalizada de Moore-Penrose de A

Inversa generalizada de Moore-Penrose

- Sea A una matriz de nxm, la inversa generalizada de Moore-Penrose, denotada como A^- es una matriz de mxn, que cumple:
  - $AA^-A = A$
  - $A^-AA^- = A^-$
  - $A^-A = (A^-A)^T$
  - $AA^- = (AA^-)^T$

Inversa generalizada de Moore-Penrose

- La inversa generalizada de Moore-Penrose de una matriz *A* siempre existe y es única.
- Si *A* es cuadrada e invertible entonces $A^{-}=A^{-1}$

Inversa generalizada de Moore-Penrose

- En nuestro caso:

Si $A\beta_{*j} = D_{*j}$

entonces $A^{-}A\beta_{*j} = A^{-}D_{*j}$

por propiedades de la inversa generalizada de Moore-Penrose nos queda $A^{-}A\beta_{*j} = A^{-}AA^{-}D_{*j}$

por lo tanto eliminando $A^{-}A$ nos queda:

$\beta_{*j} = A^{-}D_{*j}$

Inversa generalizada de Moore-Penrose

- Cómo calcularla ?

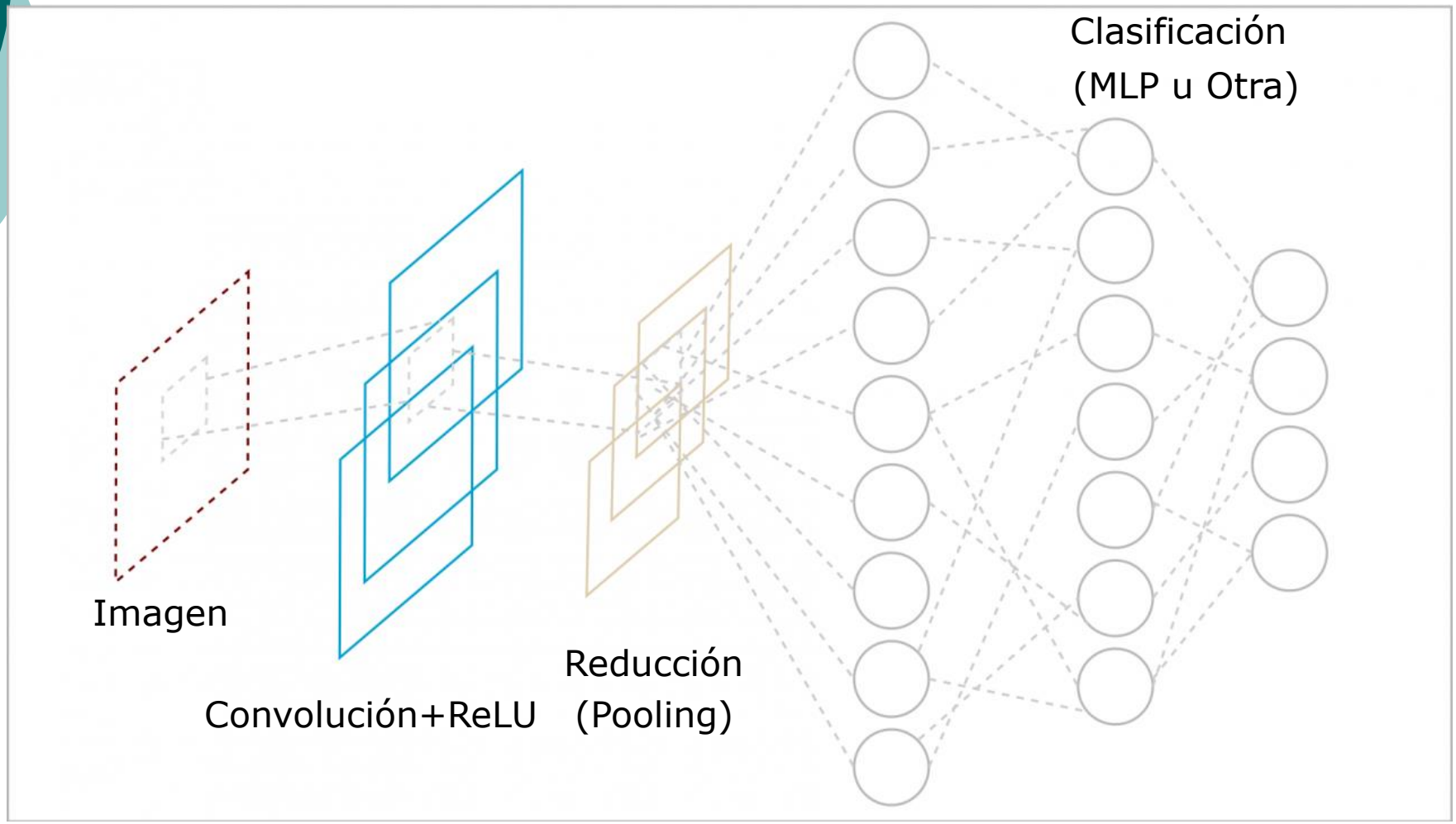
$$A^{-} = (A^{T} A)^{-1} A^{T}$$

$$A^{-} = (A^{T} A - \lambda I)^{-1} A^{T}$$

Redes Neuronales Convolucionales (CNN)

- Se usan principalmente para procesamiento de imágenes
  - Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets* (pp. 267-285). Springer, Berlin, Heidelberg.
  - LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541-551.
  - LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.
- Basadas en operaciones de convolución
- Se han usado en procesamiento de texto

Redes Neuronales Convolucionales (CNN)



Convolución

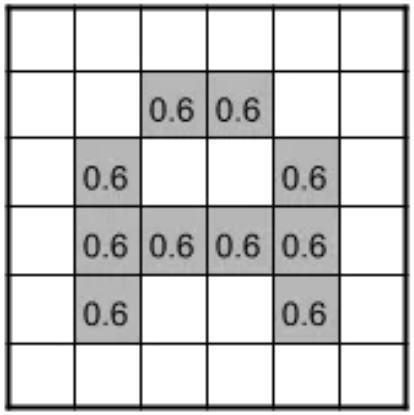
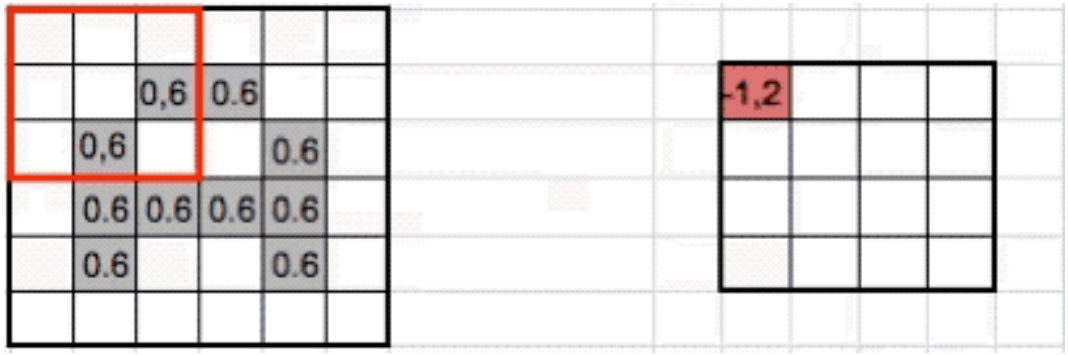


Imagen de entrada

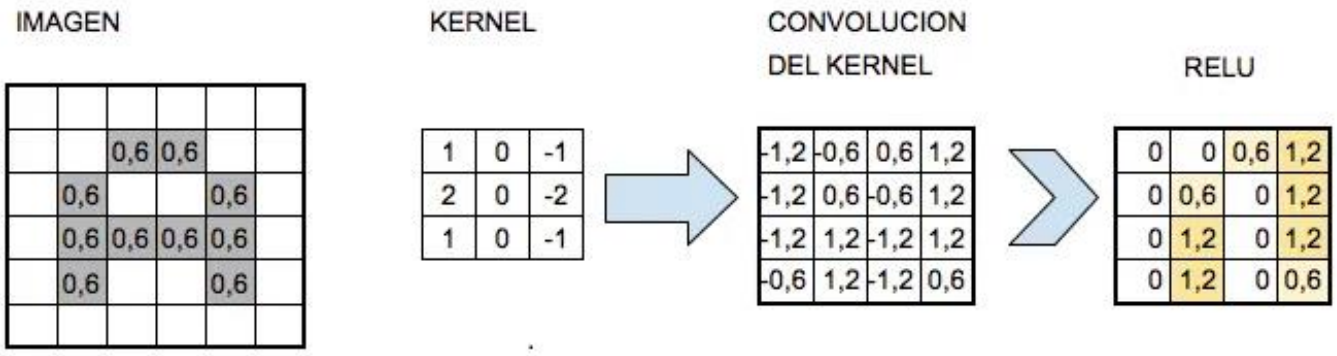
1	0	-1
2	0	-2
1	0	-1

kernel



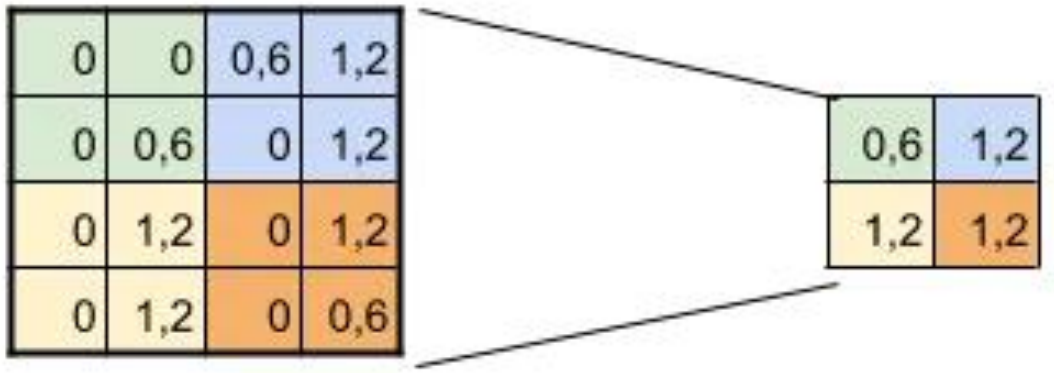
Convolución + ReLU

- Rectifier Linear Unit : f(x)=max(0,x)

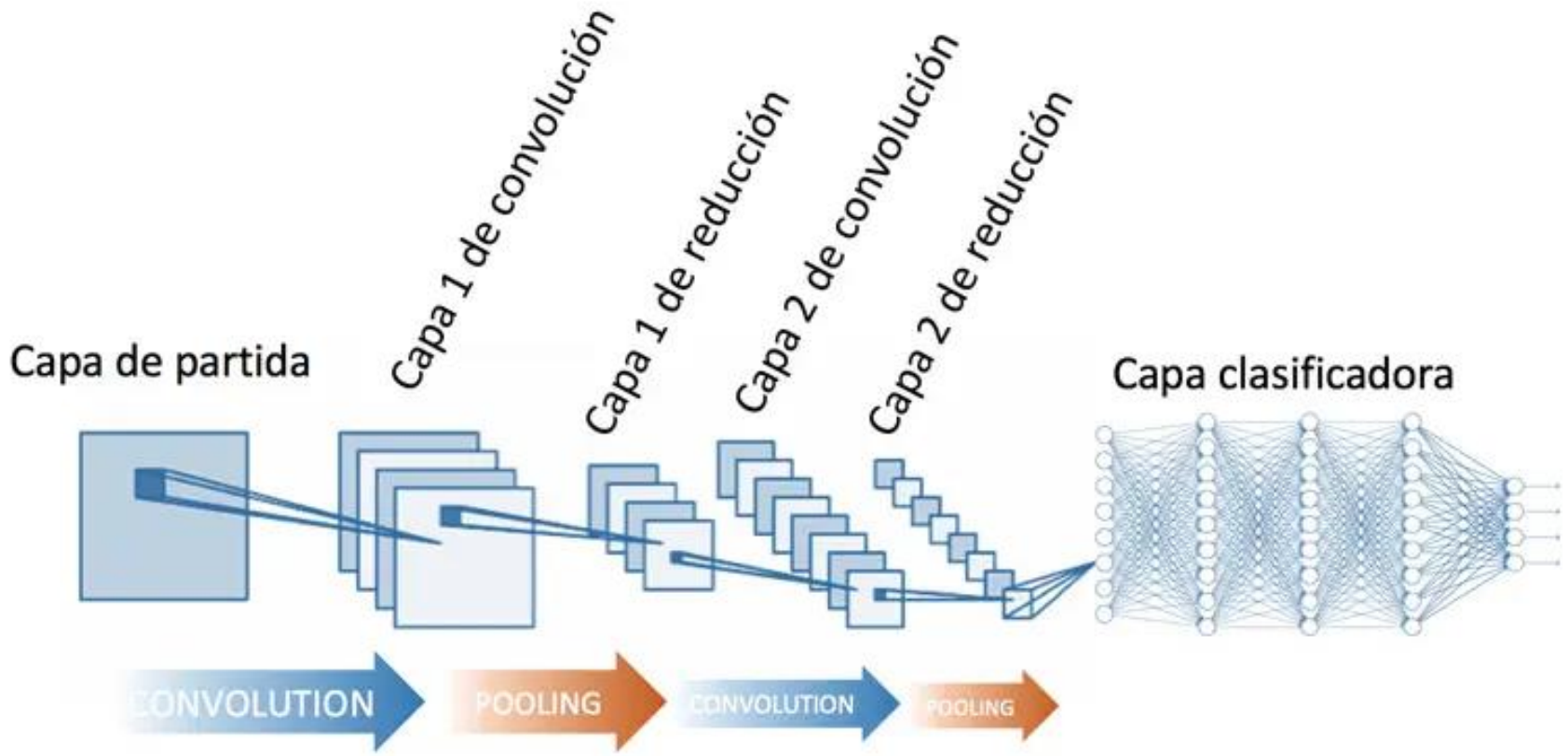


Reducción (Pooling)

- El más común es el Max-Pooling



Redes Neuronales Convolucionales (CNN)



Redes Neuronales Convolucionales (CNN)

Cómo se entrenan ?

- Se entrenan usando retropropagación.
- El número de parámetros a optimizar en la fase de convolución no es tan grande, solamente los valores de los kernels (filtros).

Redes Profundas

Redes con muchas capas.

Usualmente CNN

Cómo se entrenan ?

- Utilizando muchos recursos computacionales

Clasificación Supervisada

Redes Neuronales

Jesús Ariel Carrasco Ochoa
Instituto Nacional de Astrofísica, Óptica y Electrónica