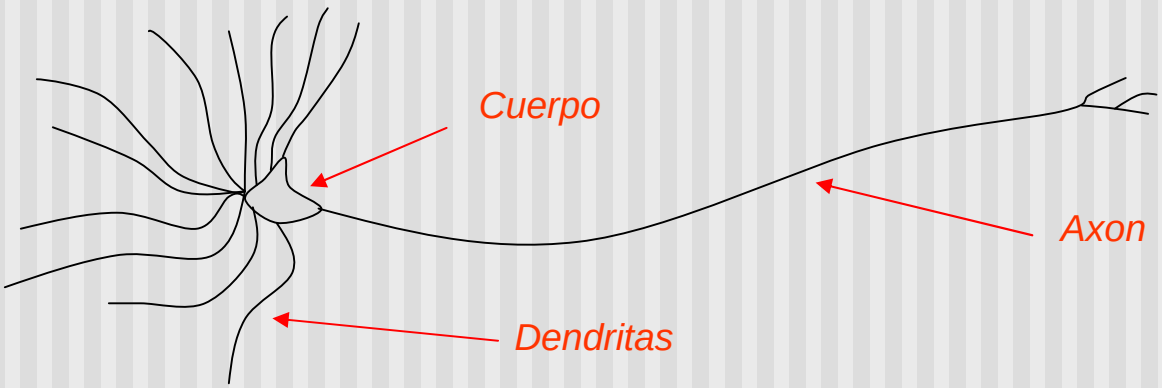
Bloque III

MÉTODOS DE APRENDIZAJE INDUCTIVO (continuación)

REDES NEURONALES

Redes neuronales (I)

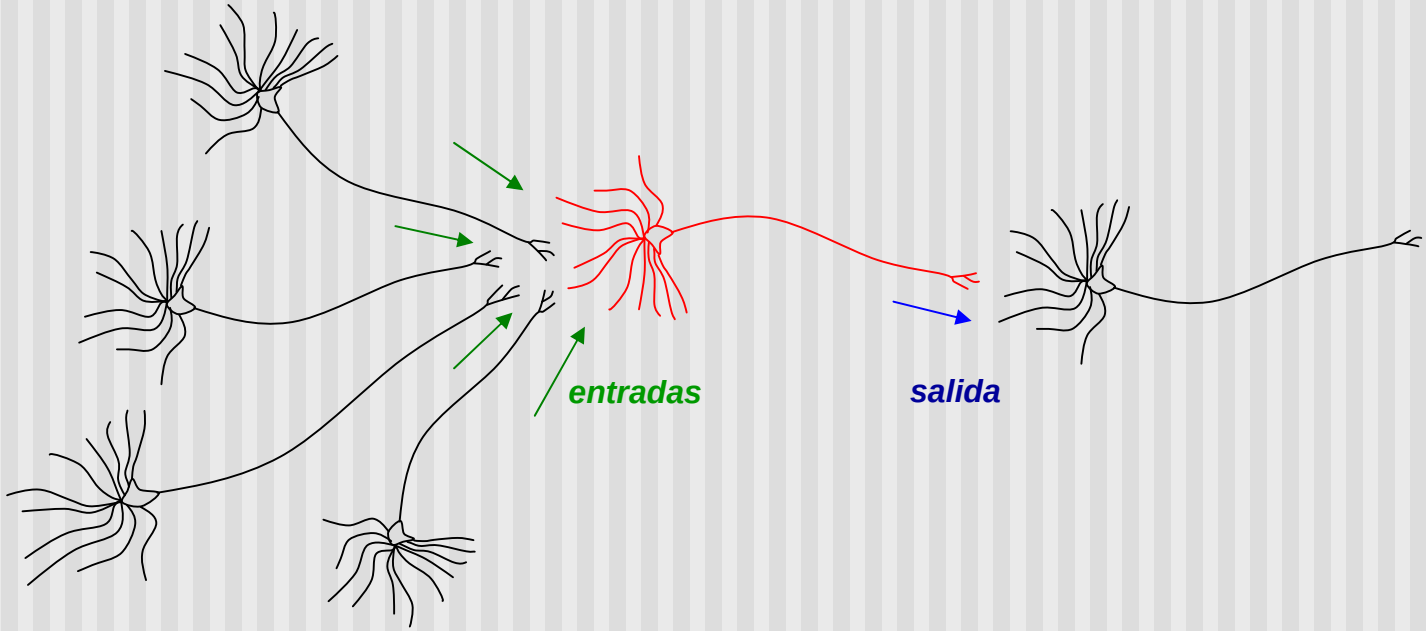
- Red neuronal: método de aprendizaje inductivo inspirado en la estructura del cerebro humano.
- Cerebro humano:
  - Millones de neuronas (10^{11} neuronas) altamente interconectadas.
  - 3 elementos fundamentales en una neurona (simplificada):
    - Cuerpo
    - Axon
    - Dendritas



  - Cada neurona se conecta a muchas otras (10^3, 10^4).
  - Las conexiones se llaman sinapsis.

Redes neuronales (II)

- ¿Cómo funcionan las conexiones entre neuronas?
  - Cada célula recibe señales (pulsos eléctricos) de las neuronas cercanas a través de sus dendritas.
  - Si el total de las señales recibidas supera un cierto umbral, la neurona se activa y envía a su vez pulsos eléctricos a través de su axon hacia otras células.

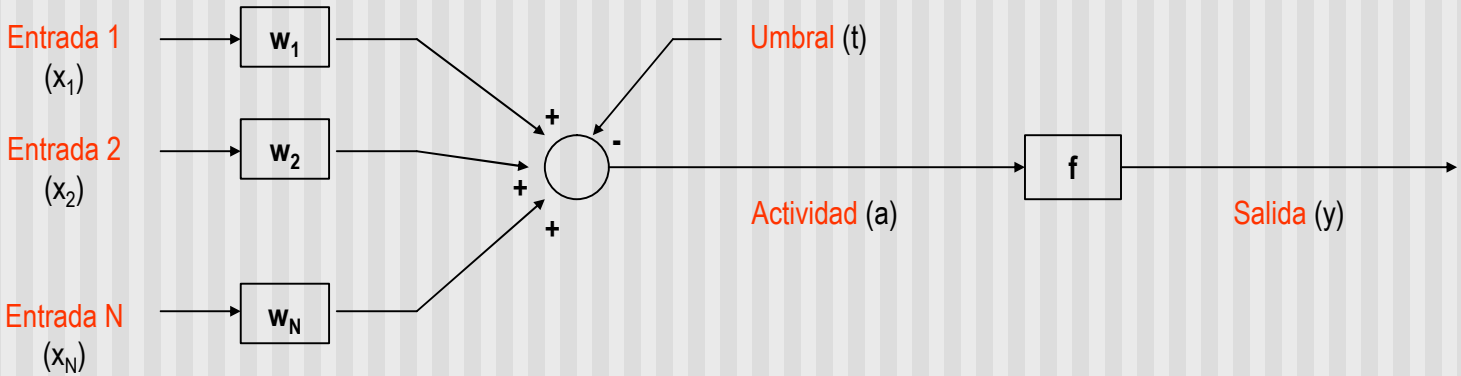


Redes neuronales (III)

- La velocidad con la que conmutan las neuronas es muy lenta (10^6 veces más lenta que una puerta lógica).
- Sin embargo, el cerebro es capaz de realizar operaciones complejas más rápidamente y más eficientemente que un ordenador (ej. Visión artificial frente a visión humana). ¿Cuál es la razón?
  - LA razón se encuentra en el alto grado de paralelismo del cerebro humano (hay millones de neuronas y conexiones).
- Las redes neuronales artificiales (ANN o NN) están inspiradas en el cerebro humano, y en las múltiples interconexiones entre neuronas.

Redes neuronales (IV)

- Modelo de una neurona artificial:



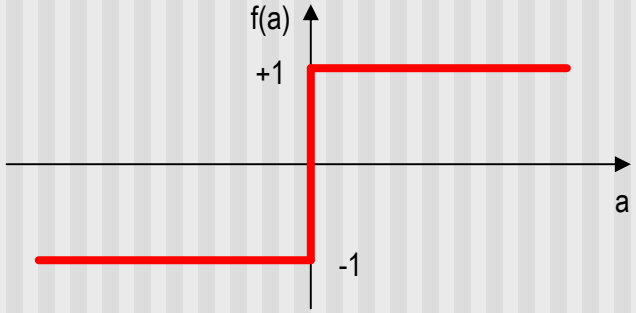
$$y = f(a)$$

$$a = \sum_{i=1}^{N} x_i \cdot w_i - t$$

$$a = \sum_{i=o}^{N} x_i \cdot w_i = \vec{x} \cdot \vec{w}; \quad x_0 = -1; \quad w_0 = t$$

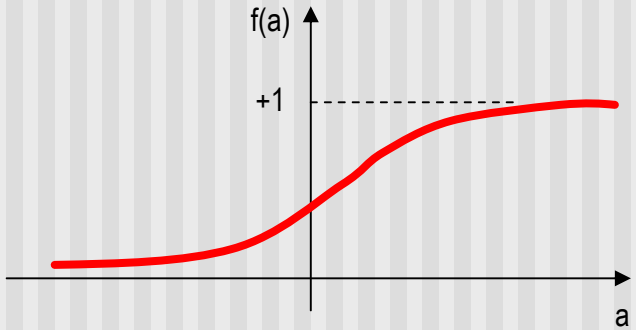
Redes neuronales (V)

- Función de activación ***f***:
  - Función más sencilla posible: signo.



$$f(a) = \begin{cases} -1 & if \;\; a < 0 \\ +1 & if \;\; a \geq 0 \end{cases}$$

  - Otra función utilizada: sigmoidea (derivable).



$$f(a) = \frac{1}{1 + e^{-k \cdot a}}$$

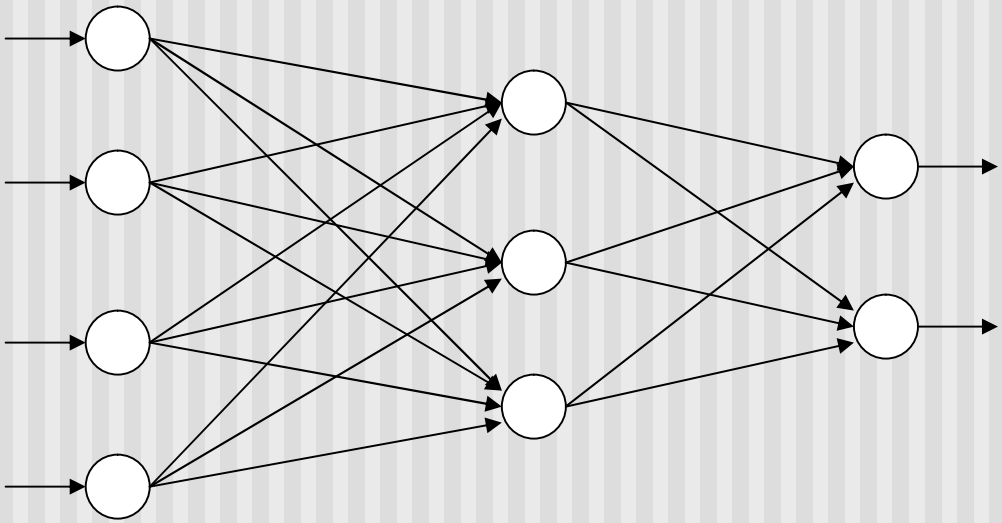
    - similar: comportamiento todo/nada para valores elevados de *k*.

Redes neuronales (VI)

- Posibles estructuras para la red:
  - Unidireccional (feedforward networks)
    - La información fluye en una única dirección.
    - Muy utilizadas, en general como clasificadores.
  - Recurrentes (feedback networks)
    - La información se realimenta mediante bucles.
    - No muy utilizadas.
  - Mapas autoorganizados (self organizing maps)
    - La estructura de la red se deforma durante el entrenamiento, intentando adaptarse a los valores de los ejemplos de entrenamiento.
    - Muy utilizadas para 'clustering' (detección de grupos de instancias similares).

Redes neuronales (VII)

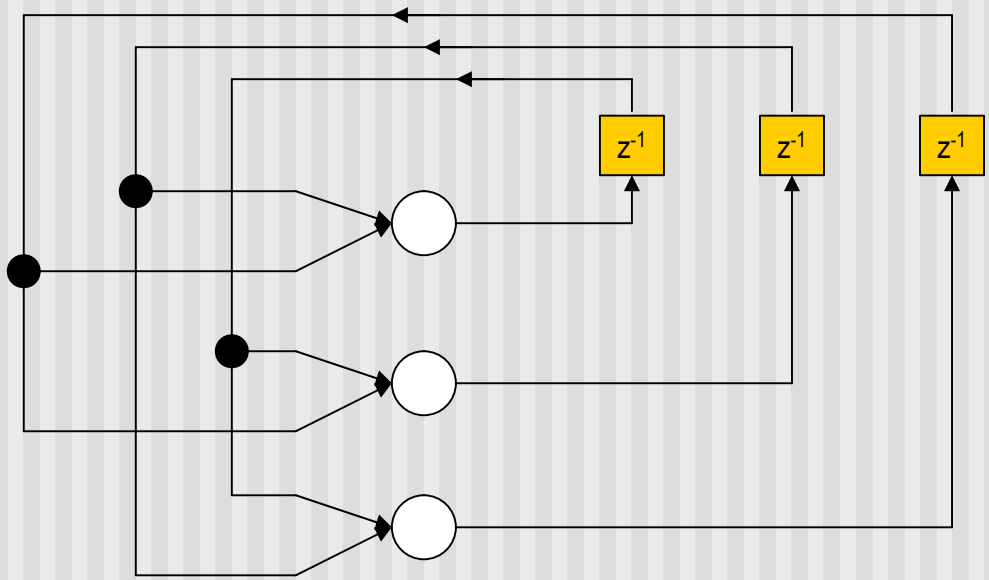
- Redes unidireccionales



  - Cada círculo representa una neurona.
  - Cada neurona recive señales desde las neuronas situadas a su izquierda y envía señales a las neuronas situadas a su derecha.

Redes neuronales (VIII)

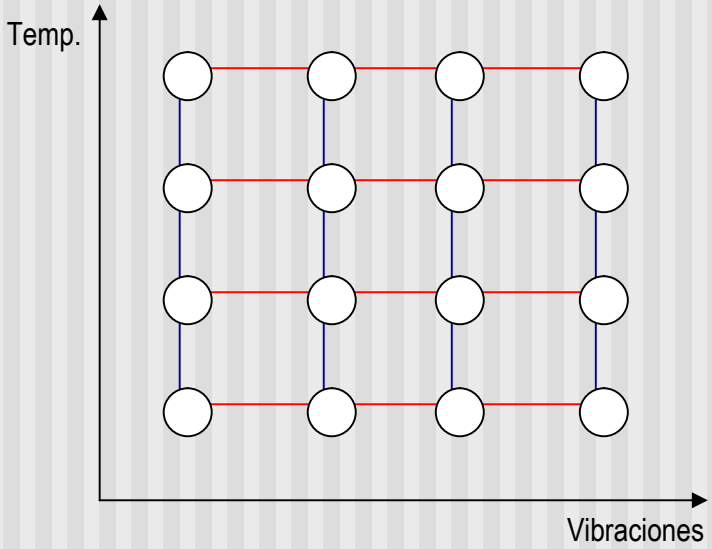
- Redes recurrentes (red de Hopfield)



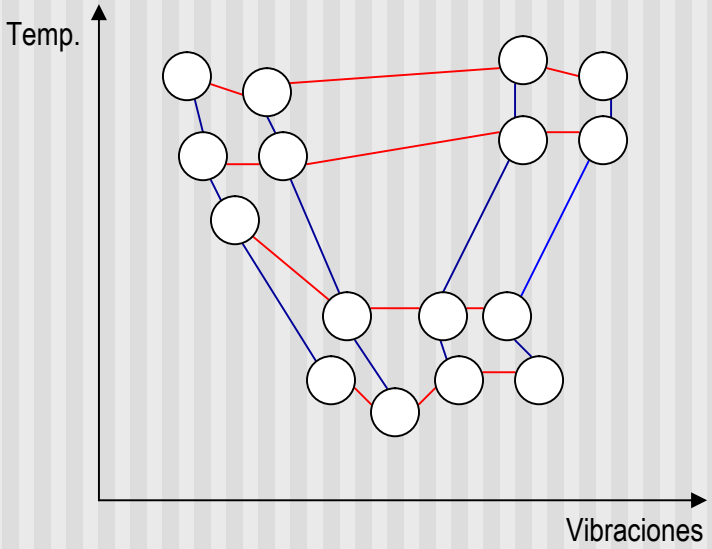
  - Circulos: neuronas; bloques z^{-1}: retrasos.
  - Las entradas de una neurona están conectadas con las salidas (retrasadas) de otras neuronas situadas al mismo nivel.

Redes neuronales (IX)

- Mapas autoorganizados (SOMs)



ANTES DEL ENTRENAMIENTO

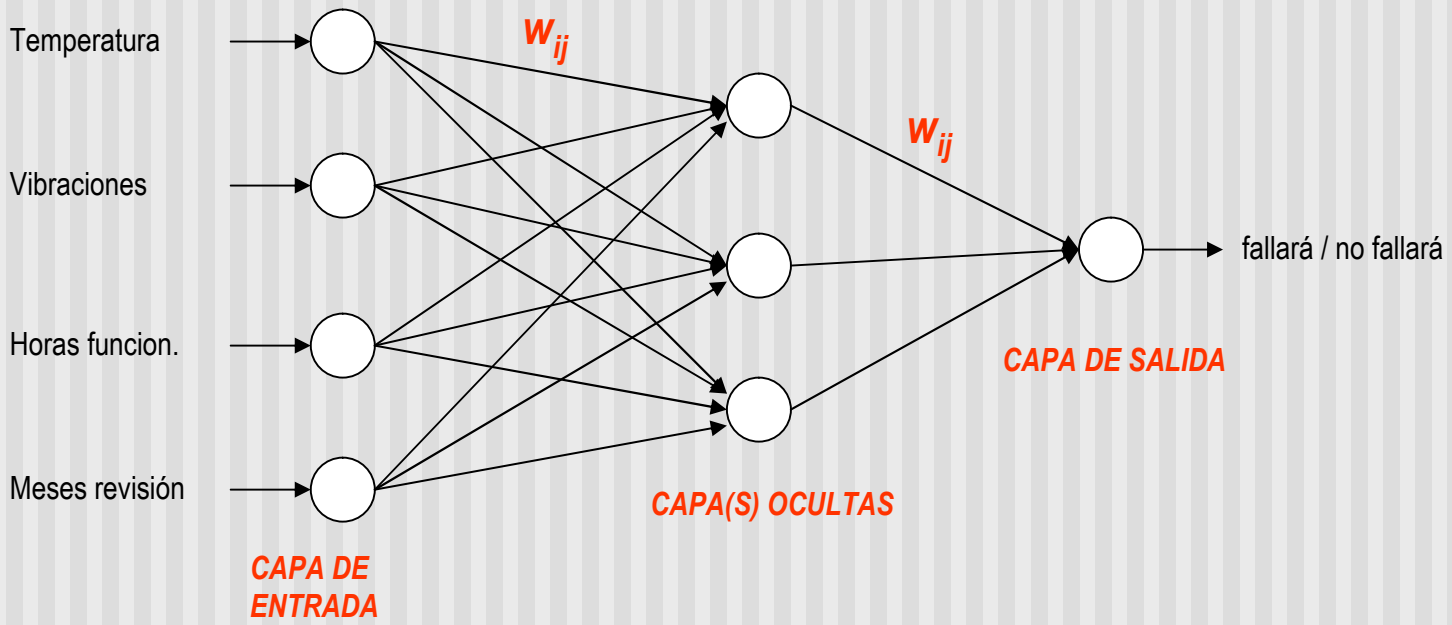


DESPUÉS DEL ENTRENAMIENTO

- Ejemplo 2D: las neuronas se adaptan a los ejemplos.
- Muy útil para representar información de múltiples dimensiones en una red 2D y visualizar clusters (grupos).

Redes neuronales (X)

- Ejemplo de red unidireccional



- Estructura en capas.
- Los pesos w_{ij} representan la influencia mutua entre neuronas.
- Durante el entrenamiento se ajustan los pesos a los ejemplos; pero la estructura permanece fija.

Redes neuronales (XI)

- Entrenamiento de redes unidireccionales
  - Los pesos se ajustan utilizando las instancias de entrenamiento.
  - Objetivo: predecir la clase correcta para cada instancia de entrenamiento.
  - Solución óptima: imposible de encontrar.
  - Se utiliza un proceso iterativo (backpropagation):
    1. Se eligen pesos aleatoriamente.
    2. Se modifican los pesos de acuerdo con los errores de clasificación obtenidos con las instancias de entrenamiento.
    3. Se repite el paso 2 mientras:
       - El error supere un cierto nivel.
       - No se sobrepase un número de iteraciones o epochs.

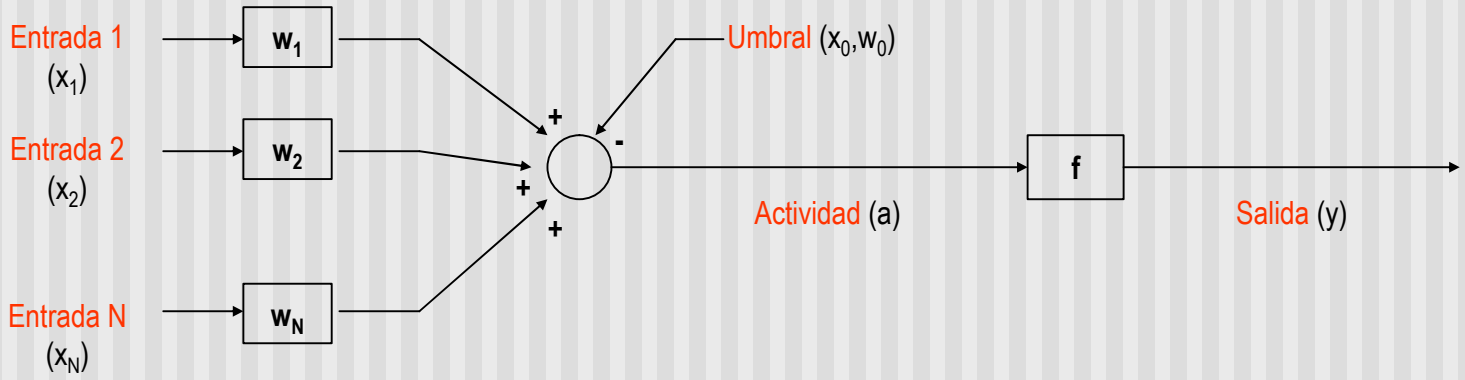
Redes neuronales (XII)

- Posibles problemas en el entrenamiento:
  - Sobreajuste cuando el número de iteraciones es demasiado elevado.
  - Mínimos locales para el error, dependiendo de los valores iniciales de los pesos.
- Parámetros a ajustar:
  - Número y tamaño de las capas ocultas.
  - Criterio de parada (normalmente 500 epochs).
  - Tasa de aprendizaje h (velocidad de convergencia).
  - Momento a (inercia para evitar mínimos locales).

Demasiados parámetros: no es fácil conseguir resultados óptimos.

Redes neuronales (XIII)

- El perceptrón simple
  - Red neuronal unidireccional más sencilla posible.
  - Una sóla neurona.



$$y = f(a)$$

$$a = \sum_{i=o}^{N} x_i \cdot w_i = \vec{x} \cdot \vec{w}$$

Redes neuronales (XIV)

- Entrenamiento del perceptrón simple
  - Se define el error para un ejemplo a partir de la diferencia entre la actividad (a) y la clase real del ejemplo

$$E(\overline{w}) = \frac{1}{2}(t-a)^2$$

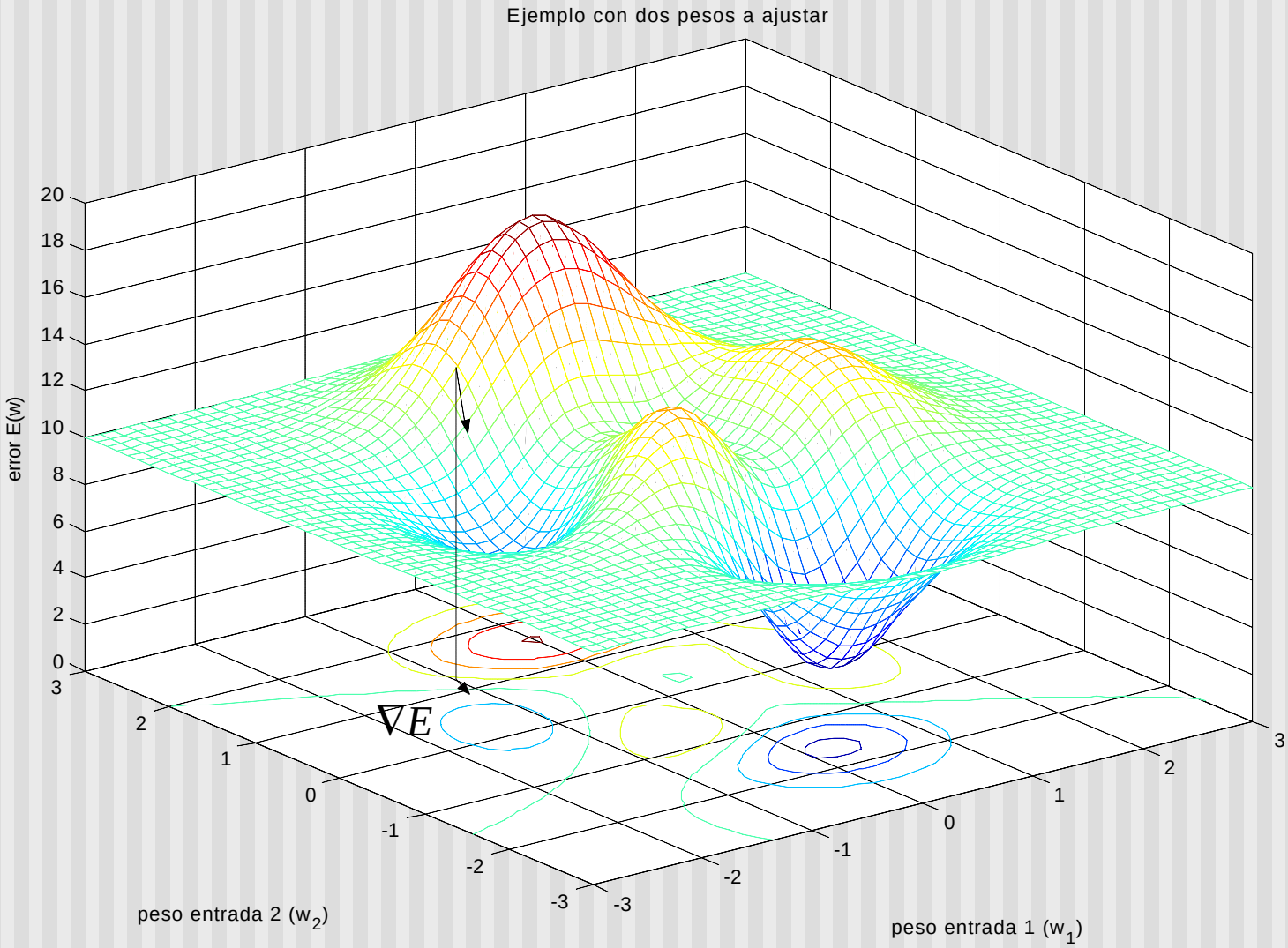
  - El error se define como una función de los pesos (vector w).

Redes neuronales (XV)

- Entrenamiento del perceptrón simple
  - Se parte de unos pesos w_i aleatorios y se modifican en la dirección que más rápidamente hace disminuir el error -> dirección del gradiente del error.

$$\nabla E(\overline{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right]$$

Redes neuronales (XVI)



Redes neuronales (XVII)

- Actualización de pesos en cada iteración:

$$\overline{w} \quad \leftarrow \quad \overline{w} - h \cdot \nabla E(\overline{w})$$

$$w_i \quad \leftarrow \quad w_i - h \cdot \frac{\partial E}{\partial w_i}$$

- Tasa de aprendizaje h :
  - *Pequeña: convergencia lenta.*
  - *Grande: puede no alcanzarse solución con precisión.*
  - *Regla general: la tasa se reduce a medida que se realizan iteraciones.*

Redes neuronales (XVIII)

- Cálculo del gradiente

$$\frac{\partial}{\partial w_i}\left[\frac{1}{2}(t-a)^2\right]=$$

$$=(t-a)\cdot\frac{\partial}{\partial w_i}(t-a)=$$

$$=(t-a)\cdot\frac{\partial}{\partial w_i}(t-\overline{w}\cdot\overline{x})=$$

$$=(t-a)\cdot(-x_i)$$

Redes neuronales (XIX)

$$w_i \leftarrow w_i - h(t-a)\cdot x_i$$

Peso i (nuevo)

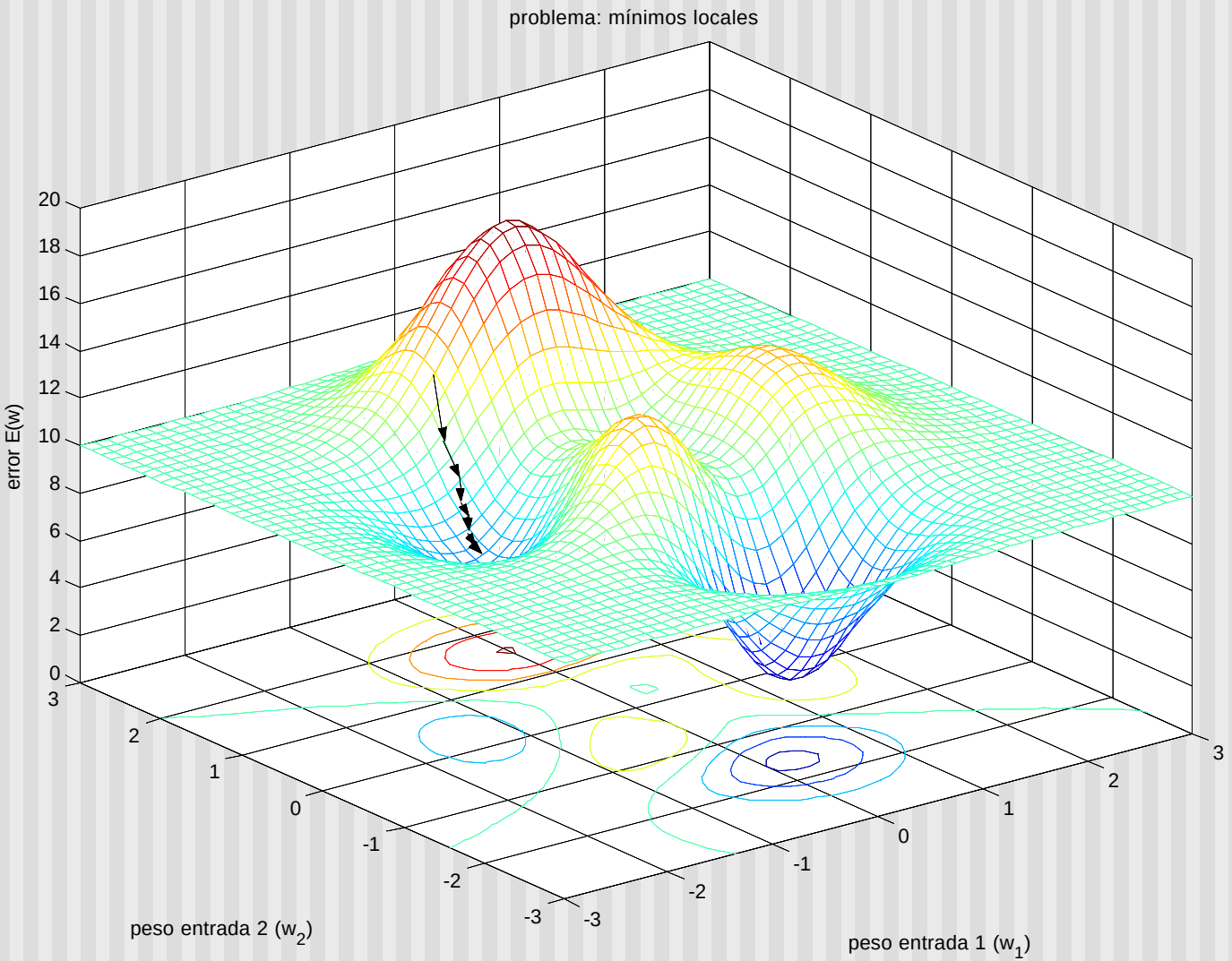
Peso i (iteración anterior)

Salida real (etiqueta)

Actividad

Entrada i

Redes neuronales (XX)



Redes neuronales (XXI)

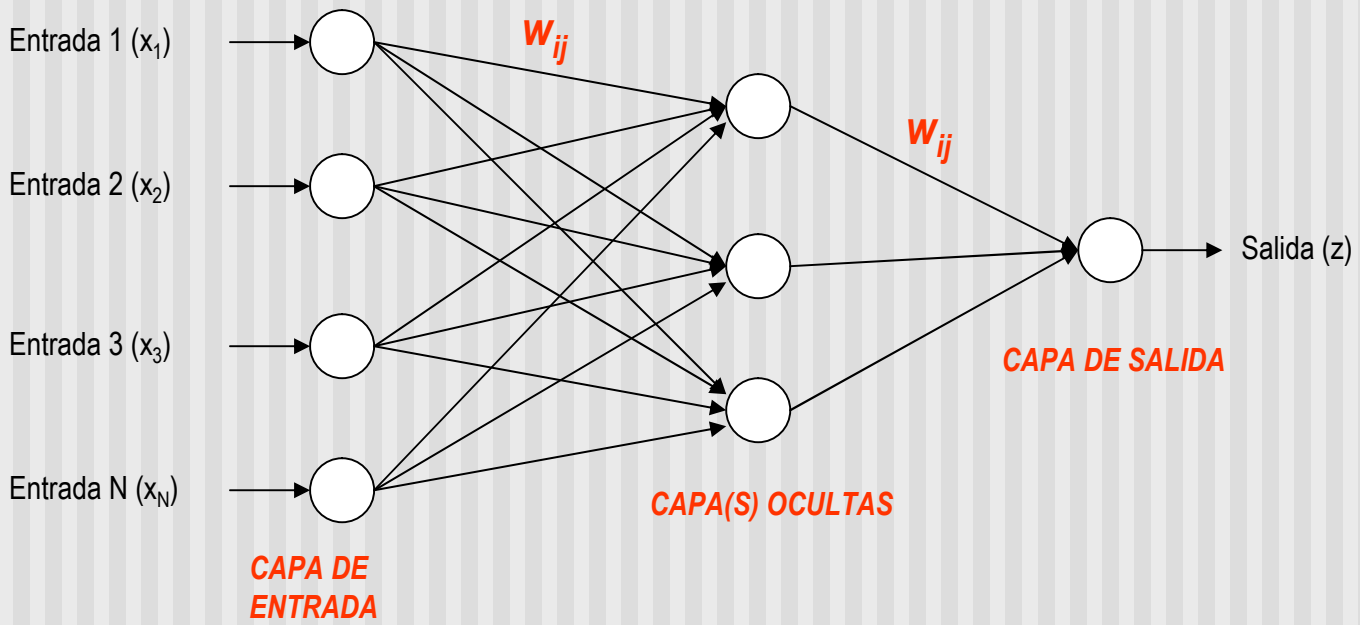
- Dos posibilidades para el entrenamiento:
  - *Incremental (la vista anteriormente):*
    - *Cada nuevo ejemplo de entrenamiento sirve para calcular el error y actualizar los pesos.*
    - *Los ejemplos se pueden utilizar varias veces para hacer tantas iteraciones como se desee.*
  - *No incremental:*
    - *En cada iteración se utilizan todos los ejemplos, y el error se define como la suma de los errores individuales:*

$$E(\overline{w}) = \frac{1}{2} \sum_{d \in ejemplos} (t_d - a_d)^2$$

$$\frac{\partial E}{\partial w_i} = \ldots = \sum_{d \in ejemplos} (t_d - a_d) \cdot (-x_{id})$$

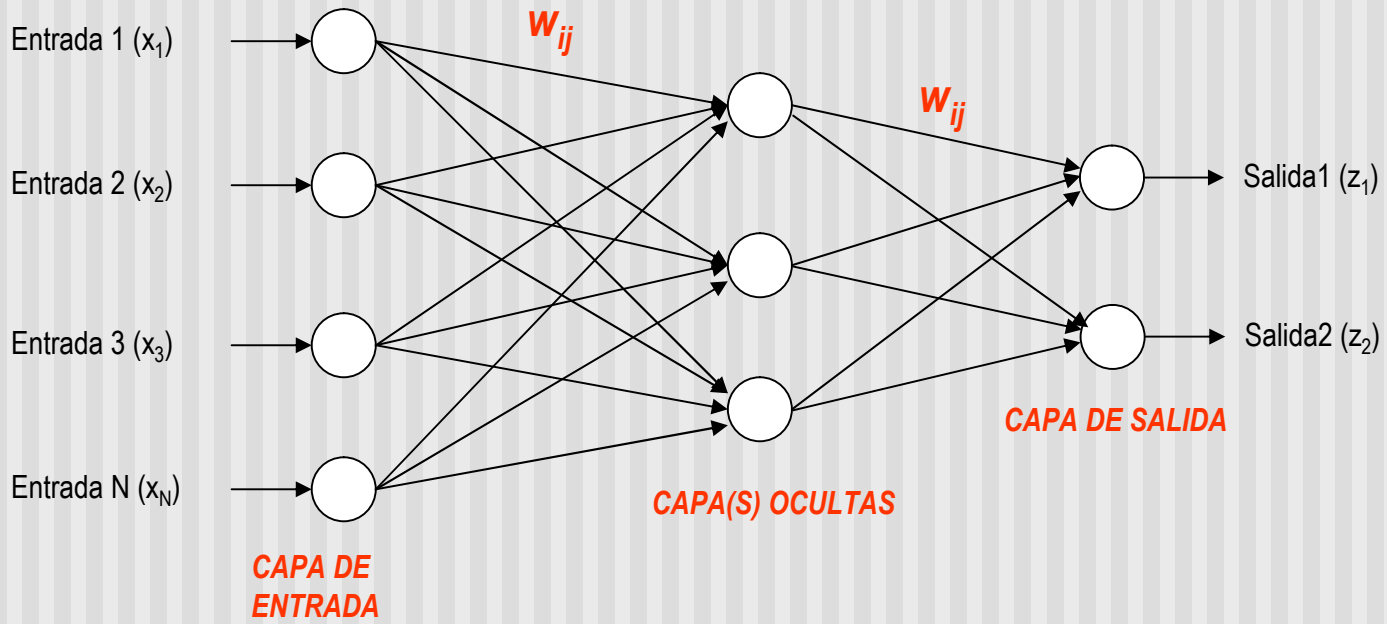
Redes neuronales (XXII)

- El perceptrón multicapa
  - Red neuronal unidireccional más común.
  - Muy utilizado como clasificador.



Redes neuronales (XXIII)

- Normalmente sólo hay una capa oculta.
- Pero pueden existir varias neuronas de salida.
  - Si el número de clases es superior a 2, hacen falta más neuronas para codificarlo.



Redes neuronales (XXIV)

- El error se define de forma distinta.
  - Debe considerar todas las salidas.
- Aprendizaje incremental (error medido para un ejemplo):

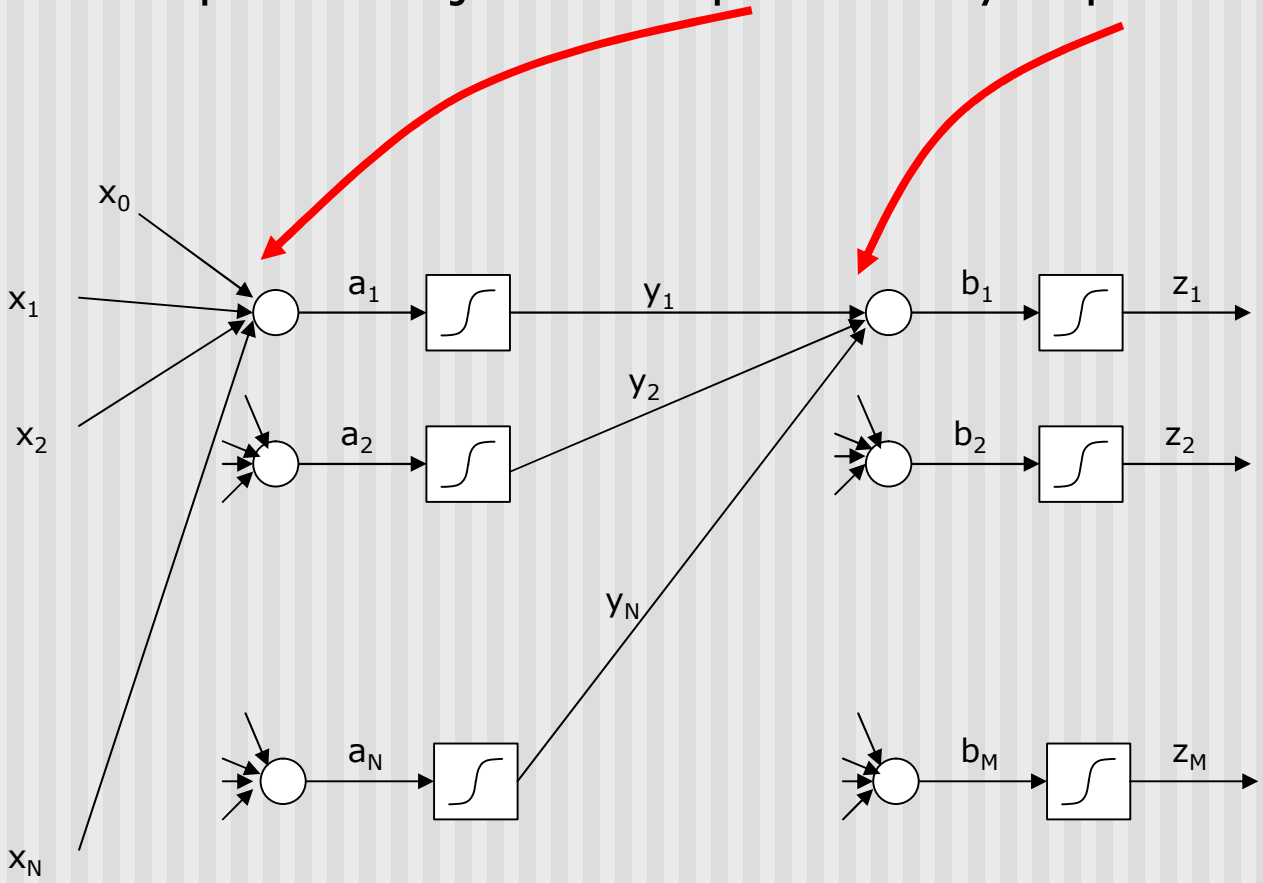
$$E = \frac{1}{2} \sum_{k \in salidas} \left(t_k - z_k \right)^2$$

- Aprendizaje no incremental (error medido para todos los ejemplos):

$$E = \frac{1}{2} \sum_{d \in ejemplos} \sum_{k \in salidas} \left(t_{kd} - z_{kd} \right)^2$$

Redes neuronales (XXV)

- Muchos pesos a ajustar: capa oculta y capa salida



Redes neuronales (XXVI)

- Entrenamiento para los pesos de la capa de salida.
  - w_{ji} = peso de la entrada i de la neurona j.

$$w_{ji} \leftarrow w_{ji} - h \cdot \frac{\partial E}{\partial w_{ji}}$$

$$E = \sum_{k \in salidas} \frac{1}{2} (t_k - z_k)^2$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial b_j} \cdot \frac{\partial b_j}{\partial w_{ji}} = \frac{\partial E}{\partial b_j} \cdot y_i$$

Redes neuronales (XXVII)

$$\frac{\partial E}{\partial b_j} = \frac{\partial E}{\partial z_j} \cdot \frac{\partial z_j}{\partial b_j}$$

$$\frac{\partial E}{\partial z_j} = \frac{\partial}{\partial z_j} \sum_k \frac{1}{2} (t_k - z_k)^2 = (t_j - z_j) \cdot \frac{\partial}{\partial z_j} (t_j - z_j) = -(t_j - z_j)$$

$$\frac{\partial z_j}{\partial b_j} = \frac{\partial}{\partial b_j} \left(\frac{1}{1+e^{-b_j}} \right) = \left(\frac{1}{1+e^{-b_j}} \right) \cdot \left(1 - \frac{1}{1+e^{-b_j}} \right) = z_j (1 - z_j)$$

$$\frac{\partial E}{\partial w_{ji}} = (t_j - z_j) \cdot z_j \cdot (1 - z_j) \cdot y_i$$

Redes neuronales (XXVIII)

$$w_{ji} \leftarrow w_{ji} + h\left(t_j - z_j\right) \cdot z_j \cdot \left(1 - z_j\right) \cdot y_i$$

Peso ji (nuevo)

Peso ji (iteración anterior)

Salida real j (etiqueta)

Salida j en iteración anterior

Salida intermedia i en iteración anterior

Redes neuronales (XXIX)

- Entrenamiento para los pesos de la capa oculta.
  - w_{ji} = peso de la entrada i de la neurona j.

$$w_{ji} \leftarrow w_{ji} - h \cdot \frac{\partial E}{\partial w_{ji}}$$

$$E = \sum_{k \in salidas} \frac{1}{2} (t_k - z_k)^2$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ji}} = \frac{\partial E}{\partial a_j} \cdot x_i$$

Redes neuronales (XXX)

$$\frac{\partial E}{\partial a_j} = \sum_{k \in salidas} \frac{\partial E}{\partial b_k} \cdot \frac{\partial b_k}{\partial a_j}$$

$$\frac{\partial E}{\partial b_k} = \quad \ldots \quad = -(t_k - z_k) \cdot z_k \cdot (1 - z_k)$$

$$\frac{\partial b_k}{\partial a_j} = \frac{\partial b_k}{\partial y_j} \cdot \frac{\partial y_j}{\partial a_j} = w_{kj} \cdot \frac{\partial y_j}{\partial a_j} = \quad \ldots \quad = w_{kj} \cdot y_j \cdot (1 - y_j)$$

$$\frac{\partial E}{\partial w_{ji}} = \left[\sum_k (t_k - z_k) \cdot z_k \cdot (1 - z_k) \cdot w_{kj}\right] \cdot y_j \cdot (1 - y_j) \cdot x_i$$

Redes neuronales (XXXI)

$$w_{ji} \leftarrow w_{ji} + h \cdot \left[\sum_k (t_k - z_k) \cdot z_k \cdot (1 - z_k) \cdot w_{kj}\right] \cdot y_j \cdot (1 - y_j) \cdot x_i$$

Peso ji (nuevo)

Peso ji (iteración anterior)

Salida real k (etiqueta)

Salida k en iteración anterior

Peso kj (iteración anterior)

Salida intermedia j en iteración anterior

Entrada i

Redes neuronales (XXXII)

- *Problema: mínimos locales.*
  - *Se introduce un elemento de inercia (momento).*

$$\Delta w_{ji}(n) = -h \cdot \frac{\partial E}{\partial w_{ji}} + a \cdot \Delta w_{ji}(n-1)$$

- *Momento* a *:*
  - *Intenta mantener la velocidad de modificación de pesos.*
  - *Puede conseguir que se salten mínimos locales.*
  - *El momento es ajustable al igual que la velocidad de convergencia: muchos parámetros.*

Redes neuronales. Resumen

1. Capacidad de representación:
   - Muy alta, fronteras de decisión no lineales.
2. Legibilidad:
   - Ninguna, los modelos son pesos para las conexiones.
3. Tiempo de cómputo on-line:
   - Rápido, las operaciones a realizar on-line son básicamente sumas y multiplicaciones.
4. Tiempo de cómputo off-line:
   - Muy lento: ajuste de pesos en un proceso iterativo.
5. Parámetros a ajustar:
   - Es complicado ajustar tanto la estructura de la red como los parámetros de entrenamiento.
6. Robustez ante ejemplos de entrenamiento ruidosos:
   - Alta.
7. Sobreajuste (overfitting):
   - Puede ocurrir si no se fijan adecuadamente los parámetros o se sobreentrena (condiciones de terminación).

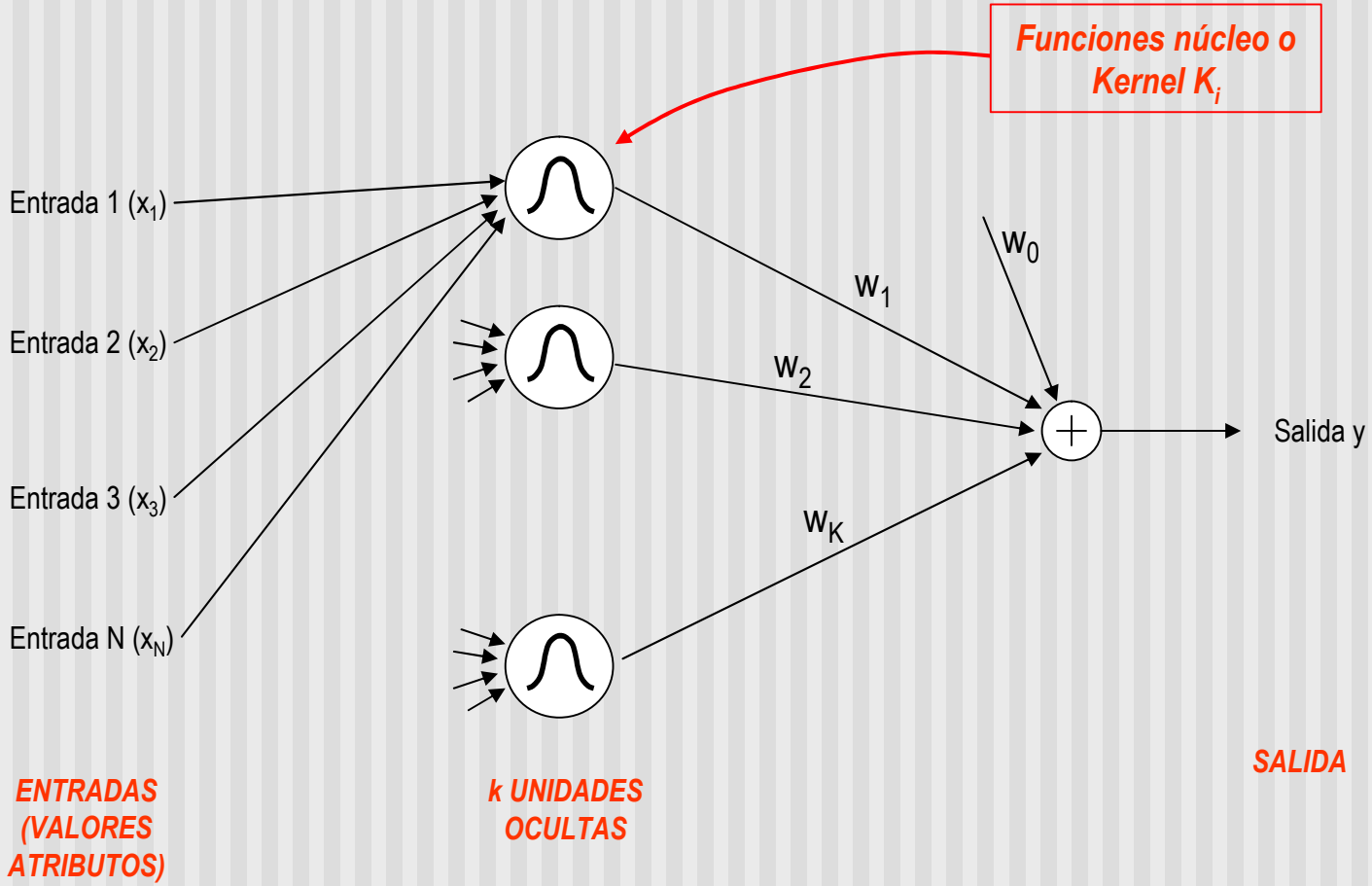
FUNCIONES DE BASE RADIAL

Funciones de base radial (I)

- Estructura muy similar a una red neuronal.
- Más fáciles de entrenar.
- Diferencias:
  - Red neuronal: aproximación de una función a partir de todos los ejemplos de entrenamiento.
  - Función de base radial (RBF): aproximación local de una función (sólo se utilizan los ejemplos de entrenamiento cercanos al ejemplo a clasificar).
- Al ser un método local tiene algunas características comunes con el vecino más cercano:
  - No es robusto ante ejemplos de entrenamiento ruidosos.
  - El tiempo de cómputo on-line es algo elevado.

Funciones de base radial (II)

Estructura básica de una RBF:



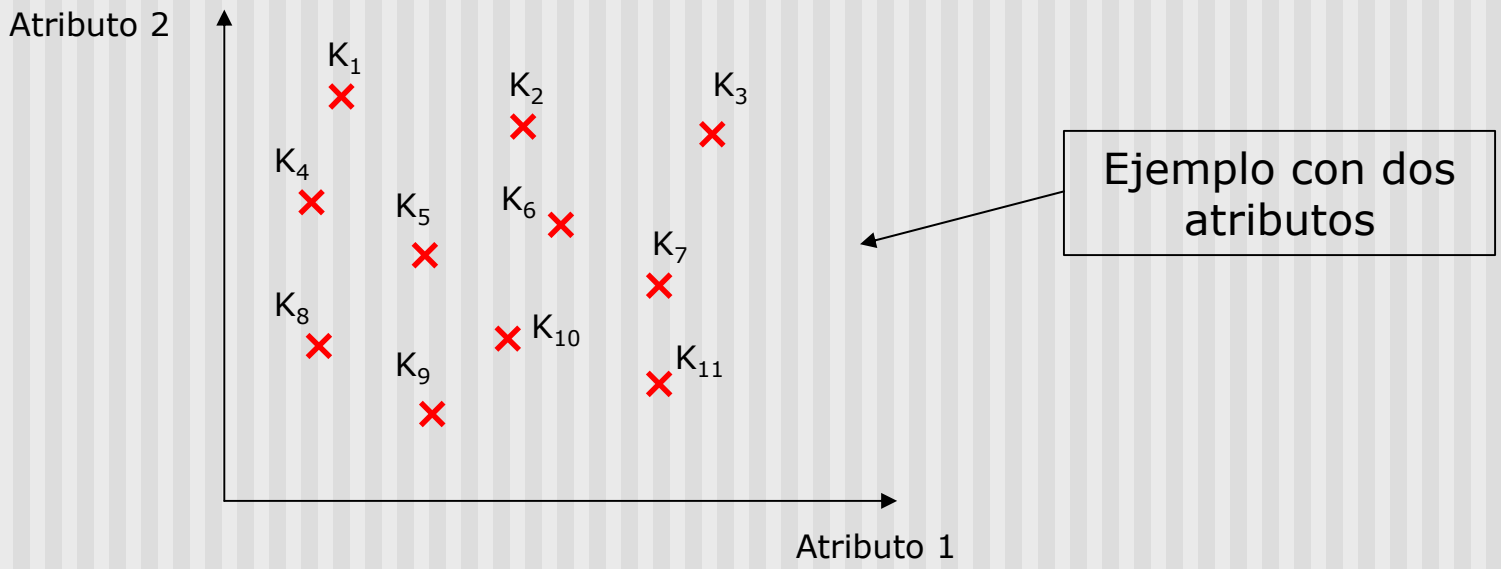
Funciones de base radial (III)

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{k} w_i \cdot K_i\left[d\left(\mathbf{x_i}, \mathbf{x}\right)\right]$$

- d = función distancia.
- $\mathbf{x_i}$ = valores de atributos del Kernel i.
- $\mathbf{x}$ = valores de atributos del ejemplo a clasificar.
- K_i= función Kernel (Gaussiana en general).
- w_i = pesos (a ajustar).
- k = número de funciones Kernel.

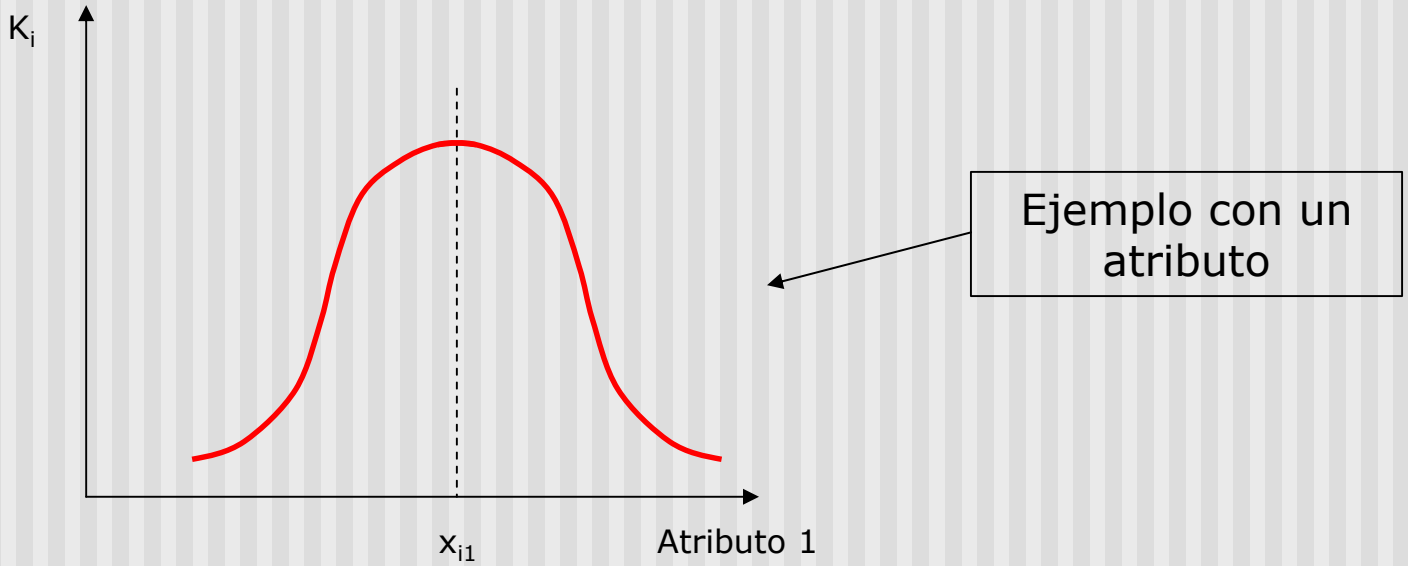
Funciones de base radial (IV)

- Cada función Kernel se asocia a un valor para los atributos (a un punto del espacio).
- En general, deben cubrir todo el espacio, o al menos la zona donde puede haber instancias o ejemplos.



Funciones de base radial (V)

- Las funciones Kernel deben ser decrecientes con la distancia.
- En general, son Gaussianas de media $\mathbf{x_i}$ y varianza fija.



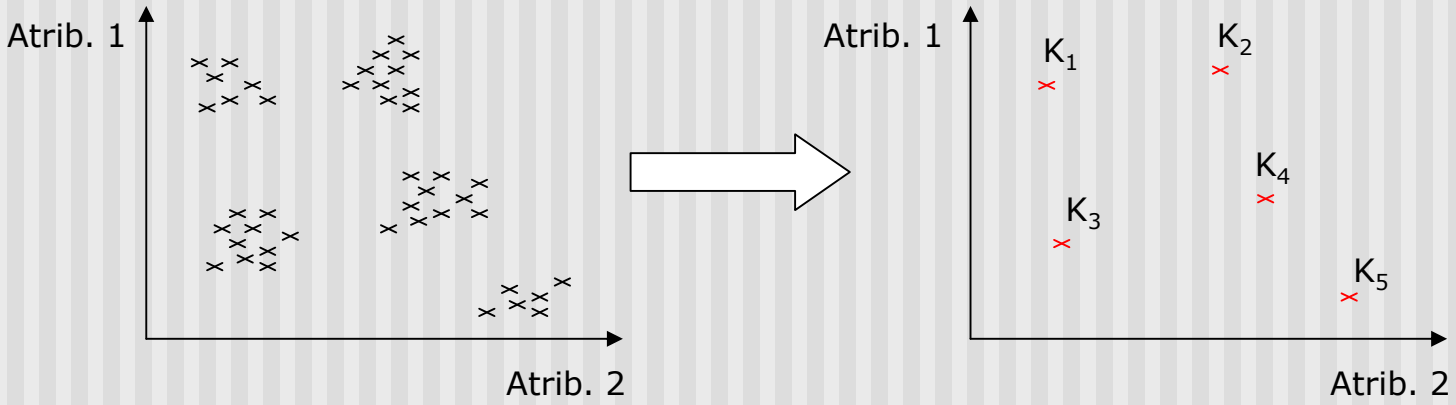
Funciones de base radial (VI)

- Clasificación mediante RBF:
  - Sólo afectan al resultado las funciones Kernel próximas al ejemplo a clasificar.
  - La proximidad se entiende como similitud de los atributos.
  - Parecido al vecino más cercano.
  - Aproximación local y no global de la función a partir de los ejemplos de entrenamiento.

- Entrenamiento de una RBF:
  - En dos etapas:
    - Se eligen manualmente media y varianza de los Kernels.
    - Se ajustan los pesos a partir de los ejemplos.
  - Más rápido (cómputo) que un perceptrón multicapa.
  - Más sencillo ajustar los parámetros del entrenamiento.

Funciones de base radial (VII)

- Selección del número de funciones Kernel:
  - Posibilidad 1:
    - Tantas funciones como ejemplos de entrenamiento.
    - Cada función centrada sobre un ejemplo (medias de las Gaussianas).
    - Varianza uniforme (desviaciones típicas de las Gaussianas).
  - Posibilidad 2:
    - Menos funciones que ejemplos de entrenamiento.
    - Se utiliza cuando el número de ejemplos es muy elevado.
    - Los Kernels se distribuyen uniformemente o no según lo estén los ejemplos:



Funciones de base radial. Resumen

1. Capacidad de representación:
   - Muy alta, similar a un perceptrón multicapa.
2. Legibilidad:
   - Ninguna, como las redes neuronales.
3. Tiempo de cómputo on-line:
   - Más lento que un perceptrón multicapa (se deben calcular distancias a todas las funciones Kernel).
4. Tiempo de cómputo off-line:
   - Más rápido que un perceptrón multicapa.
5. Parámetros a ajustar:
   - Pocos parámetros, básicamente el número de funciones Kernel.
6. Robustez ante ejemplos de entrenamiento ruidosos:
   - Baja por ser un método local.
7. Sobreajuste (overfitting):
   - Puede ocurrir si se dimensiona mal (más funciones Kernel que ejemplos de entrenamiento).

Bloque III

MÉTODOS DE APRENDIZAJE INDUCTIVO (continuación)