Redes de Neuronas Artificiales

Aprendizaje Automático

Ingeniería Informática

Fernando Fernández Rebollo y Daniel Borrajo Millán

Grupo de Planificación y Aprendizaje (PLG)
Departamento de Informática
Escuela Politécnica Superior
Universidad Carlos III de Madrid

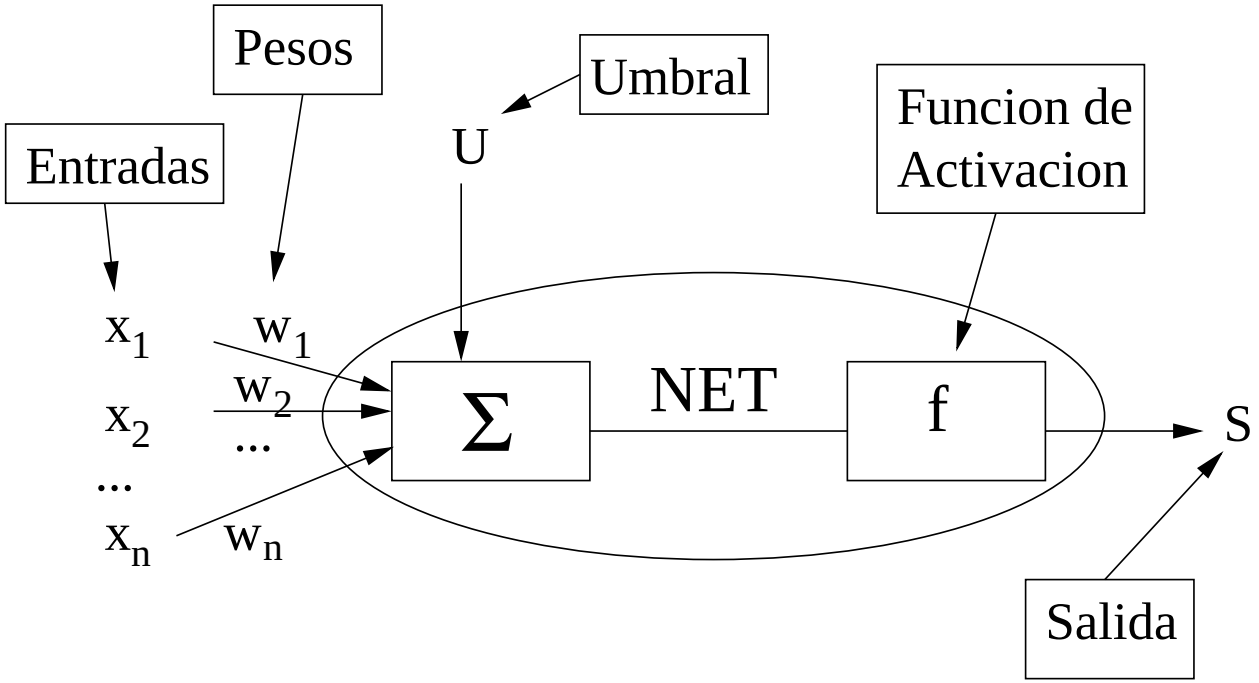
27 de febrero de 2009

En Esta Sección:

Definición de Neurona Artificial

- Búsqueda de algoritmos capaces de procesar información al igual que el cerebro humano
- Neurona artificial: unidad elemental de una red de neuronas artificiales
- Características de una neurona:
  1. Recibe un conjunto de **señales de entrada** procedentes del mundo exterior o de otras neuronas
  2. Las señales de entrada se reciben a través de unas conexiones, las cuales tienen un número real asociado llamado **peso**
  3. **Procesa la información** recibida, mediante una serie de operaciones simples
  4. Emite una **señal de salida** como respuesta a las señales de entrada

Estructura de una Neurona



- Salida de la neurona:
  $S = f(NET) = f(x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + U) = f(\sum_{i=1}^{n} x_i w_i + U)$

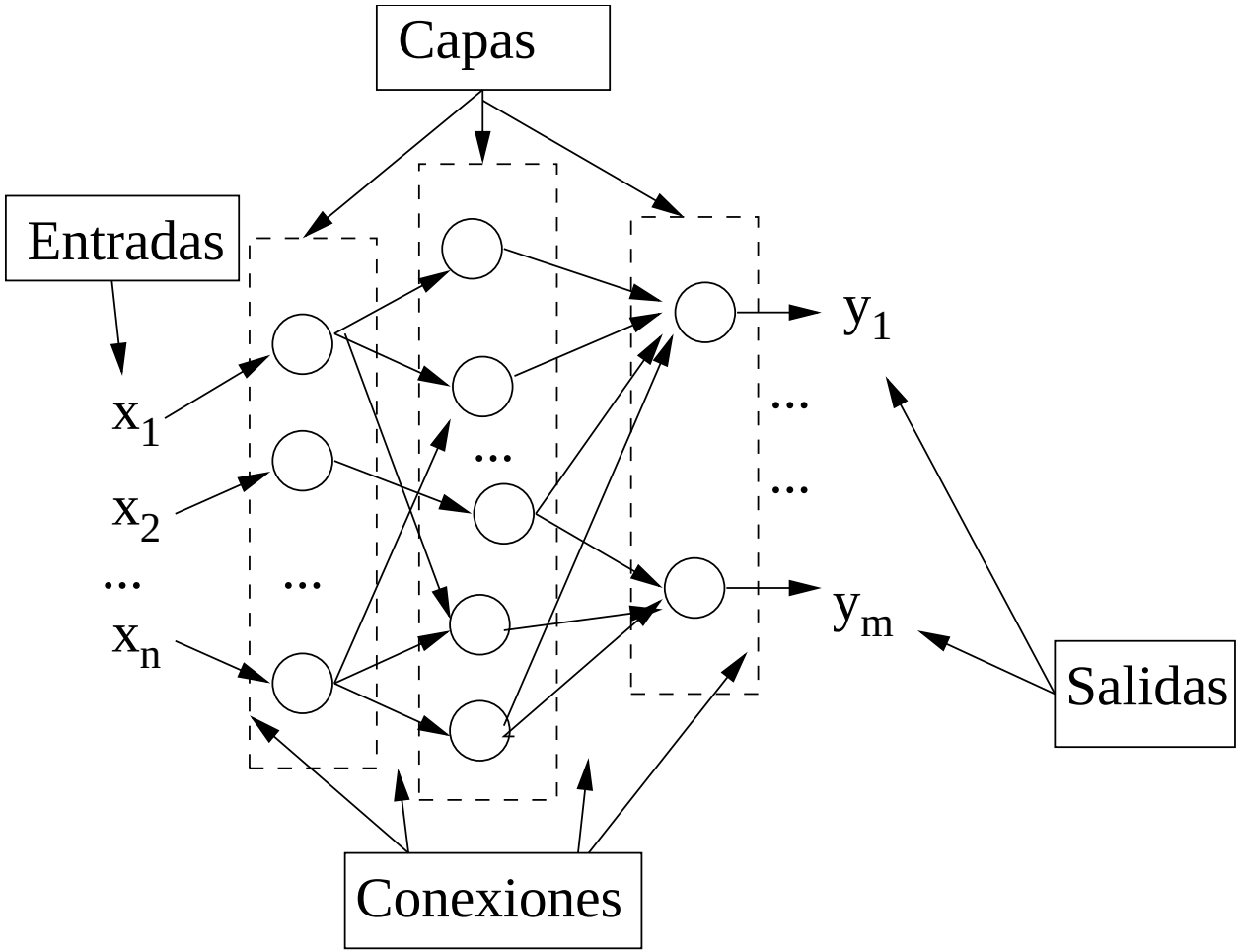
Funciones de activación

- Función lineal:
  $f(x) = x$
- Función umbral:
  $$f(x) = \begin{cases} f1 & x > 0 \\ -f1 & x \leq 0 \end{cases}$$
- Función gausiana:
  $f(x) = e^{\frac{-x^2}{2}}$
- Funciones sigmoidales
  - Función en $(0, 1)$: $f(x) = \frac{1}{1+e^{-x}}$
  - Función en $(-1, 1)$:
    $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Definición de Red de Neuronas

- Conjunto de neuronas artificiales conectadas entre sí mediante una serie de arcos llamados **conexiones**.
- Estas conexiones tienen números reales asociados, llamados **pesos** de la conexión
- Las neuronas generalmente se distribuyen en **capas** de distintos niveles, con conexiones que unen las neuonas de las distintas capas y/o neuronas de una misma capa.

Ejemplo de Red de Neuronas

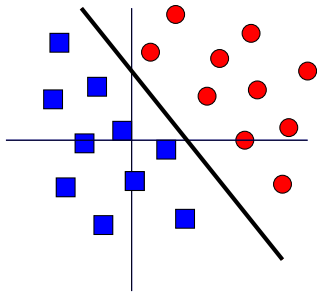


Definición de Aprendizaje de la Red de Neuronas

- Aprendizaje de la Red: proceso mediante el cual la red modifica los pesos de las conexiones para que las salidas de la red se vayan adaptando de manera paulatina al funcionamiento que se considera correcto.
- Esta modificación de los pesos se realiza en base a un criterio establecido:
  - Aprendizaje supervisado: para cada patrón o ejemplo presentado a la red existe una respuesta deseada. La respuesta de la red se compara con su salida deseada, y en base a esa comparación se ajustan los pesos de la red.
  - Aprendizaje no supervisado: no se especifica a la red cuál es la respuesta correcta. La red descubre las relaciones presentes en los ejemplos mediante reglas de aprendizaje.

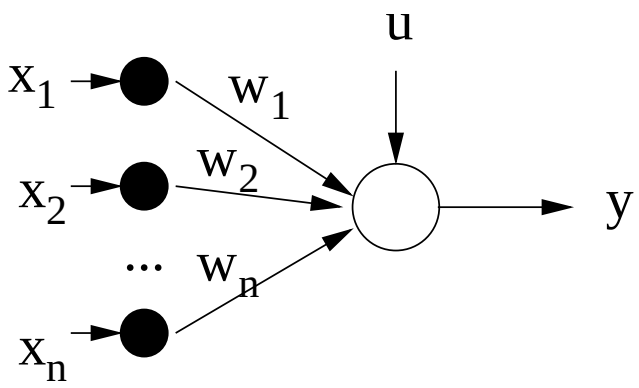
Introducción

- Forma más simple de red de neuronas
- Adaptación supervisada
- Ejemplos vistos como puntos en el espacio $\mathcal{R}^n$, junto con su clase asociada
- Tareas de clasificación lineal: dado un conjunto de patrones o ejemplos, determinar el hiperplano capaz de discriminar los patrones en dos clases:
- Hiperplano: $x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + w_0 = 0$



Arquitectura

- Arquitectura:



- $y = f(x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + w_0)$
- $f(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases}$
- La red puede utilizarse para clasificación supervisada:
  - Si $y = 1$ entonces $(x_1, \ldots, x_n) \in C_1$
  - Si $y = -1$ entonces $(x_1, \ldots, x_n) \in C_2$

Aprendizaje

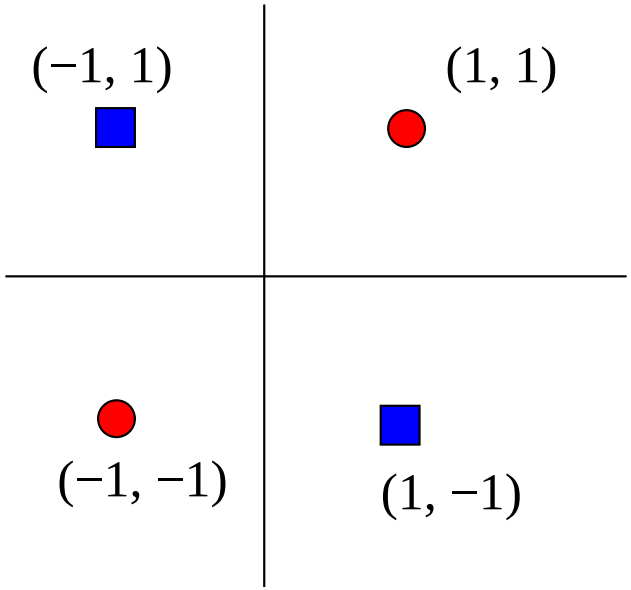
- Proceso iterativo supervisado de presentación de patrones: modificación de los parámetros de la red (pesos y umbral) hasta encontrar el hiperplano discriminante
- Número finito de iteraciones
- Entrada: conjunto de pares $< \vec{x}, d(\vec{x}) >$, donde
  - $\vec{x} = (x_1, \ldots, x_n)$
  - Salida deseada, $d(\vec{x})$, donde:
    - Si $d(\vec{x}) = 1$ entonces $\vec{x} \in C_1$
    - Si $d(\vec{x}) = -1$ entonces $\vec{x} \in C_2$
- Salida del proceso de aprendizaje:
  - Pesos y umbral, $w_0, \ldots, w_n$

Algoritmo de Aprendizaje

1. Inicialización aleatoria de los pesos y del umbral, $w_0, \ldots, w_n$
2. Elegir un patrón de entrada, con su salida deseada, $< \vec{x} = (x_1, \ldots, x_n), d(\vec{x}) >$
3. Calcular la salida de la red: $y = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + w_0$
4. Actualizar los pesos de la red:
   - Si $y = d(\vec{x})$ (clasificación correcta), volver al paso 2
   - Si $y \neq d(\vec{x})$ (clasificación incorrecta), actualizar según la siguiente ley de aprendizaje:
     - Caso 1: $d(\vec{x}) = 1, y = -1 \Rightarrow$
       $w_i(t+1) = w_i(t) + x_i$
       $u(t+1) = u(t) + 1$
     - Caso 2: $d(\vec{x}) = -1, y = 1 \Rightarrow$
       $w_i(t+1) = w_i(t) - x_i$
       $u(t+1) = u(t) - 1$

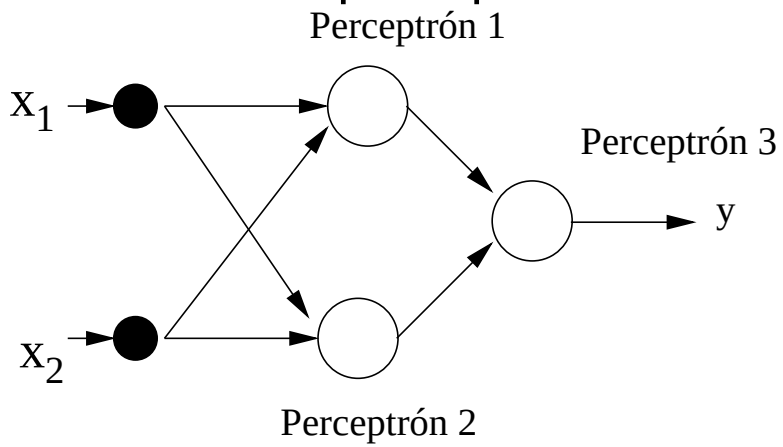
Limitaciones del perceptrón simple

- Si no existe un hiperplano, la solución no existe.
- Ejemplo de la función xor:



Limitaciones del perceptrón simple

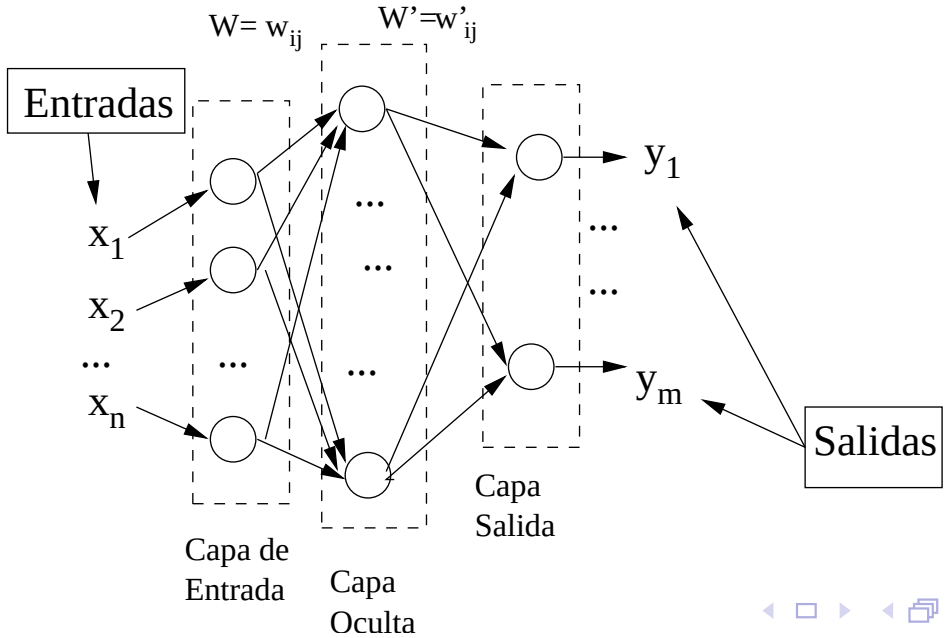
- Solución: combinar varios perceptrones:



- Problema: La ley de aprendizaje no es aplicable, puesto que no se conocen las salidas deseadas de los perceptrones interiores (en el ejemplo, del 1 y del 2). Por tanto, los pesos deberían ser calculados mediante un proceso manual.

Definición

- Neuronas agrupadas en capas
- Cada neurona en cada capa está agrupada a todas las neuronas de la capa siguiente
- Cada neurona procesa la información recibida y propaga la respuesta a través de la conexión con todas las neuronas de la capa siguiente



Arquitectura

- Capas:
  - Capa de entrada: recibe los patrones del exterior: $a_i = x_i; i = 1, \ldots, n$
  - Capa oculta: $b_j = f(\sum_{i=1}^{n} w_{ij} a_i + u_j), j = 1, \ldots, r$
    f: función sigmoidal
  - Capa salida: proporciona la salida de la red: $y = f(\sum_{i=1}^{r} w'_{ij} b_i + v_j), j = 1, \ldots, m$
- Extensible a más de una capa oculta
- Número de neuronas en las capas de entrada y salida viene definido por el problema
- Número de capas ocultas y neuronas en cada capa: definir por prueba y error

Aprendizaje

- Diferencia con perceptrón simple: función de activación sigmoidal (rango de salidas continuo)
- Ahora se busca minimizar el error de salida
- Entrada: conjunto de pares $< \vec{x}, \vec{t}(\vec{x}) >$, donde
  - $\vec{x} = (x_1, \ldots, x_n)$
  - Salida deseada, $\vec{t}(\vec{x})$, donde:
    - Si $\vec{t}(\vec{x}) = (t_1, \ldots, t_m)$
- Salida del aprendizaje:
  - Matrices de pesos $W = (w_{ij})$, $W' = (w'_{ij})$ y vectores de umbrales, $U = (u_i)$, $V = (v_i)$
  - Tales que se minimice el error entre la salida de la red y la salida deseada, es decir, minimizar $E = \sum_{\vec{x}} \| t(\vec{x}) - y(\vec{x}) \|$

Aprendizaje por descenso de gradiente

- Problema de optimización no lineal (función sigmoidal) resuelto mediante método de descenso de gradiente
- Descenso de gradiente: modificar los parámetros de la red siguiendo la dirección negativa del gradiente del error:

$$w^{nuevo} = w^{anterior} + \alpha(-\frac{\delta e}{\delta w}) = w^{anterior} - \alpha\frac{\delta e}{\delta w}, \forall w \quad (36)$$

- El algoritmo de retropropagación es el resultado de aplicar dicho método al perceptrón multicapa
- Ahora los errores sí pueden propagarse desde la capa de salida hasta el resto de las capas.

Condición de parada y razón de aprendizaje

- ¿Cuándo se debe parar el aprendizaje??
- Relación entre mínimos locales y globales
- Relación entre entrenamiento y test
- Relación con la razón de aprendizaje, α

Clasificación

- Sea $X = (\vec{x}_1, \ldots, \vec{x}_k)$ el conjunto de todos los patrones de entrenamiento, con $\vec{x}_i = (x_{i1}, \ldots, x_{in})$
- Sea $C = C_1, \ldots, C_m$ el conjunto finito de posibles clases a las que puede pertenecer cada patrón
- Generar una red con:
  - n neuronas en la capa de entrada
  - m neuronas en la capa de salida
  - La salida deseada para cada patrón de entrada $\vec{x}_i$ es una m-tupla $(a_1, \ldots, a_m)$ donde:
    - Si $\vec{x}_i$ pertenece a la clase C_j, entonces $a_i = 1$
    - Si $\vec{x}_i$ no pertenece a la clase C_j, entonces $a_i = 0$

    Dado un nuevo patrón en la red, se dará la clase cuya neurona asociada reciba una mayor activación
- Problema de la codificación de los patrones de entrada

Predicción de series temporales (I)

- Redes de neuronas muy útiles para aproximar cualquier función en general.
- El comportamiento temporal viene dado por ecuaciones en diferencias. Ejemplo:
  Serie temporal logística: $x(t+1) = ax(t)(1-x(t))$
- Problema de predicción surge cuando la relación entre x(t+1) y sus valores anteriores es desconocida

Predicción de series temporales (II)

- Definición del problema: Determinar f tal que $x(t+1) = f(x(t), x(t-1), \ldots, x(t-d))$, para todo $t = d, d+1, d+2, \ldots$
- Determinar el valor de d: determina el número de capas de entrada
- Una única neurona de salida
- Una vez aprendida la función, puede utilizarse para predicciones unitarias, o para predecir series enteras a partir de d valores iniciales correspondientes con d instantes de tiempo sucesivos.

Aproximación de funciones en aprendizaje por refuerzo

- Problemas de aprendizaje por refuerzo:
  - Definición de tabla $Q(s, a)$
  - Posibilidad de conjunto de estados y acciones infinitos
- Utilizar una red de neuronas para aproximar la función $Q(s, a)$:
  - Requiere de un proceso iterativo
  - Capa de entrada: tantas neuronas como dimensión del estado más la dimensión de la acción
  - Capa de salida: única neurona generando el valor de la función

Bibliografía

- Pedro Isasi and Inés Galván. Redes de neuronas artificiales : un enfoque práctico. Pearson Prentice Hall. 2004.
- Tom Mitchell. Machine Learning, capítulo 4, McGraw-Hill 1997