Control Inteligente

Fundamentos de las Redes Neuronales
L6

Luis Moreno, Santiago Garrido, Dorin Copaci

Dpto. Ing. de Sistemas y Automática
Universidad Carlos III
Madrid

Oct 2019

Contenido

Un poco de historia sobre las redes neuronales

- Los comienzos:
  - Arranca en los años 40 del siglo XX, con los trabajos de McCulloch y Pitt [1943] que muestran que redes de neuronas artificiales podían, en principio, calcular funciones aritméticas y lógicas.
  - Posteriormente Hebb en 1949 propuso un mecanismo de aprendizaje para las neuronas.
  - En 1950 Rosenblatt construyó la primera aplicación práctica de las redes de neuronas artificiales y demostró la capacidad de estas para el reconocimiento de patrones.
  - Widrow y Hoff en 1960 introdujeron un nuevo método de aprendizaje y lo usaron para entrenar el *adaptive linear neural networks* (un tipo de estructura).
  - Se denominó regla de aprendizaje Widrow-Hoff (aún en uso).

Más historia

- El parón:
  - Debido a la influencia de Minsky y Papert [1969] en su trabajo sobre las limitaciones que presentaban las redes de Rosenblatt y Widrow condujeron a que se consideraran las redes neuronales una vía sin futuro y disminuyó el interés por estas.
  - La ausencia de computadores sobre los que experimentar disminuyó hasta casí cero el interés en la investigación sobre redes neuronales.
  - Aún así, Kohonen [1972] y Anderson [1972] desarrollaron redes neuronales que podían actuar como memorias (actualmente sus ideas se usan en la memorias cache de los computadores).
  - Este parón continuó hasta la década de los 80, donde se producen dos hechos claves:
    - Aparecen los microprocesadores.
    - Surgen nuevas ideas.

Lo último de historia

- La explosión científica:
  - En el 1982 Hopfield (físico) utilizó la mecánica estadística para explicar la operación de un cierto tipo de redes neuronales recurrentes.
  - Se propuso el algoritmo de entrenamiento backpropagation para las redes multicapa de tipo *perceptrón*. El trabajo más significativo en esta vía fue el de Rumelhart [1986].
  - A partir de aquí, su aplicación, desarrollo y nuevas ideas se han ido sucediendo y si no con regularidad ya que han existido altibajos al menos siempre han estado presentes en muchos campos de aplicación.

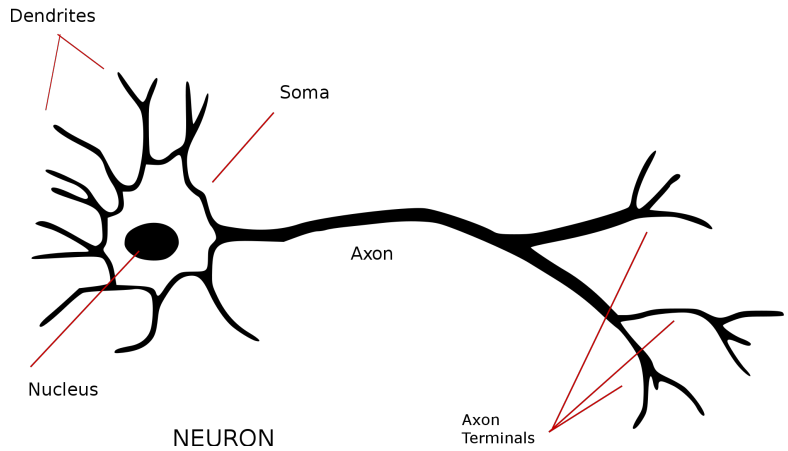
Potencial de aplicación

- Algunos ejemplos
  - Microsoft ha desarrollado neural networks para ayudar a convertir Inglés hablado en Chino hablado.
  - Google usa neural networks para etiquetado automático de imágenes (image tagging), se identifica la imagen y se le asigna palabras clave automáticamente (keywords)
  - La Lund University y el Skaen (Suecia) usan neural networks para mejorar las tasas de supervivencia a largo plazo de los receptores de trasplantes de corazón mediante la identificaciónó del receptor óptimo para un órgano donado.

La neurona

Inspiración biológica de las redes neuronales

- El elemento básico del sistema nervioso es la neurona, cuya estructura es de la forma:



  - El cerebro humano contiene un gran número de neuronas (aprox. 10^{11}) altamente conectadas entre sí (aprox. 10^4 conexiones por neurona).
  - Las dendritas son redes receptivas ramificadas de fibras nerviosas que transportan las señales eléctricas al cuerpo de la célula, donde se suman y se comparan con los umbrales de activación.
  - El axón es una única y larga fibra que lleva la señal de salida de la célula a otras neuronas. El punto de contacto del axón de una neurona con la dendrita de otra es en la sinápsis.
  - La disposición de las neuronas y la potencia de sus sinápsis son determinadas por unos complejos procesos químicos que establecen la función de la red neuronal.

La neurona

Inspiración biológica de las redes neuronales

- Algunas de las estructuras neuronales se definen al nacer.
- Otras se desarrollan durante el aprendizaje, se crean nuevas conexiones y otras desaparecen.
- Este proceso es muy perceptible en los primeros meses de vida. Los lingüistas han descubierto que los bebés de menos de seis meses no pueden discriminar ciertos tipos de sonidos a menos que hayan sido expuestos a ellos antes de esa edad.
- Las estructuras neuronales continúan cambiando a lo largo de la vida, este cambio consiste normalmente en reforzar o debilitar las uniones sinápticas. Se cree que los nuevos recuerdos se forman por modificación de estos pesos sinápticos
- Neurocientíficos han descubierto que el hipocampo de los taxistas de Londres es significativamente mayor en promedio. Se supone que es porque deben de memorizar un mayor número de información para la navegación - un proceso que requiere dos años.

La neurona

Inspiración biológica de las redes neuronales

- La redes neuronales artificiales no se aproximan a la complejidad del cerebro.
- Pero hay dos similitudes entre la redes neuronales biológicas y las artificiales.
  - La primera es que los bloques que constituyen ambos tipos de redes son dispositivos computacionales simples (las neuronas artificiales son mucho más simples que las naturales) que están fuertemente interconectados.
  - En segundo lugar, las conexiones entre neuronas determinan la función de la red.
- Aunque las redes neuronales biológicas son muy lentas comparadas con los circuitos eléctricos (10^{-3} s por 10^{-10} s), el cerebro es capaz de realizar las tareas mucho más rápido que los computadores convencionales debido a su estructura masívamente paralela.

La neurona

Activación

- El mecanismo básico de activación de la neurona es algo del tipo:

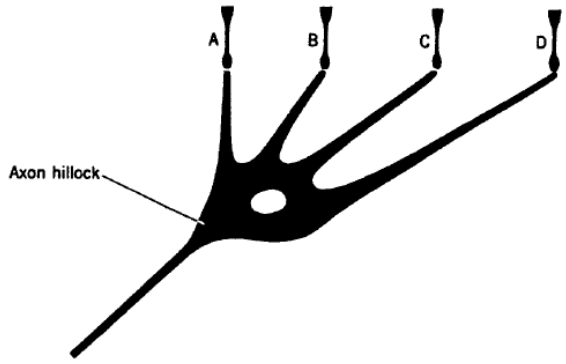
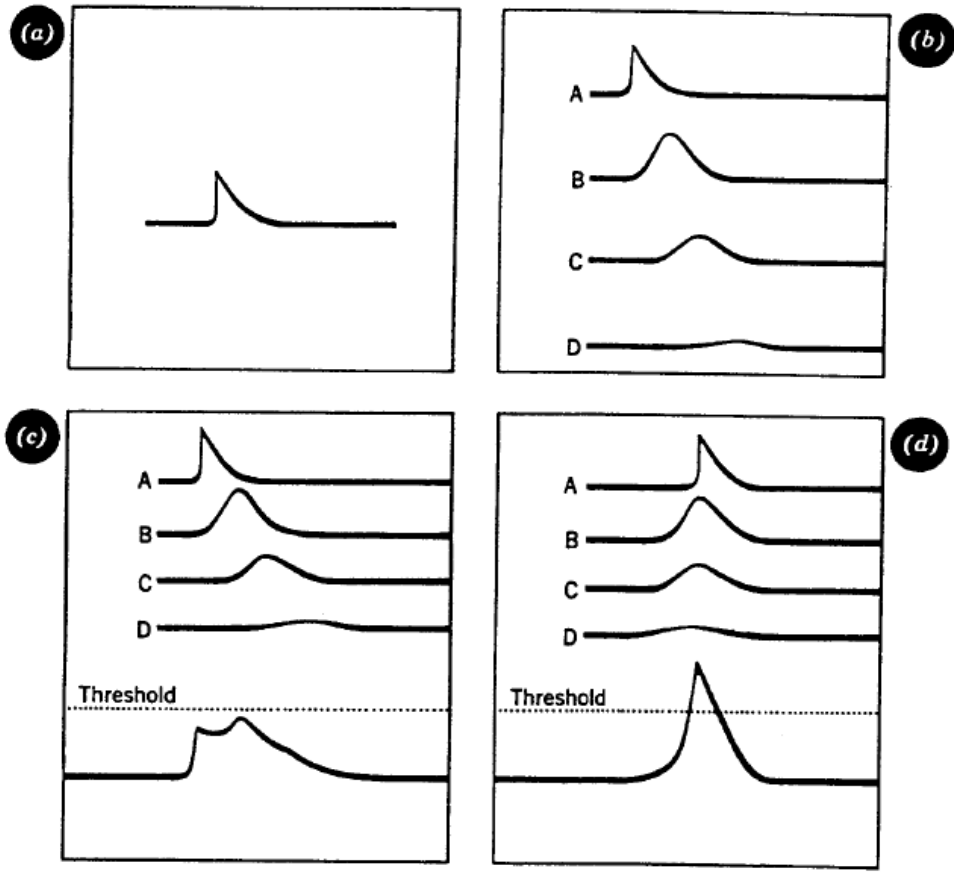


Figura: Cálculos dendríticos [Arbib(1989)]



El cerebro humano

Estructura

- Esta jerarquizado, y hay funciones que se ejecutan a distintos niveles.
- Primer nivel (neurona)
  - Sinapsis, es el nivel más básico y opera con reacciones a nivel iónico.
  - Microcircuitos neuronales, son ensamblajes de sinapsis en patrones de conexión determinados.
  - Subunidades dendríticas que agrupan microcircuitos neuronales
  - Árboles dendríticos dentro de las neuronas individuales.
  - Una neurona (100 mm) contiene diferentes subunidades dendríticas

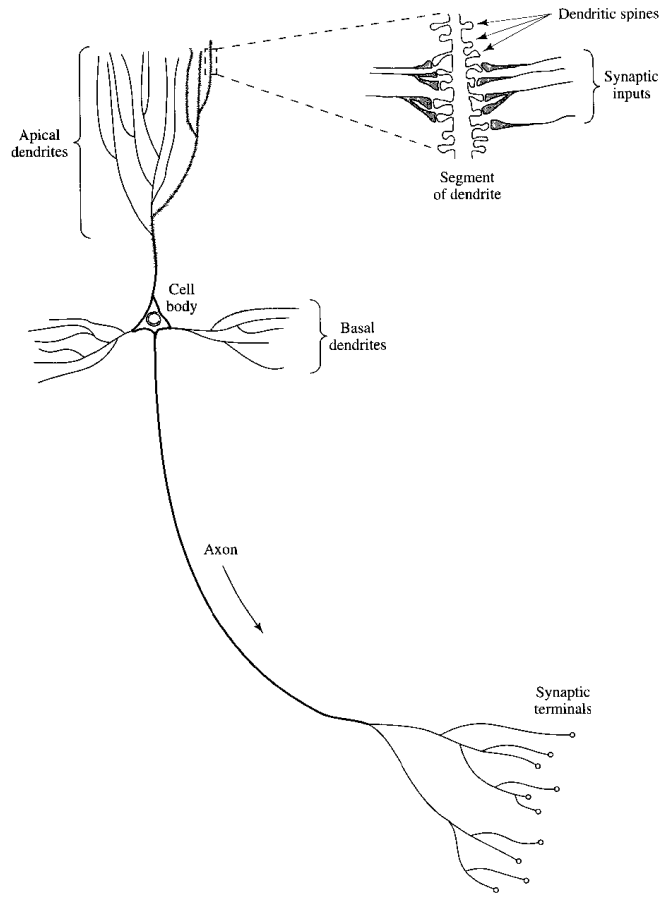
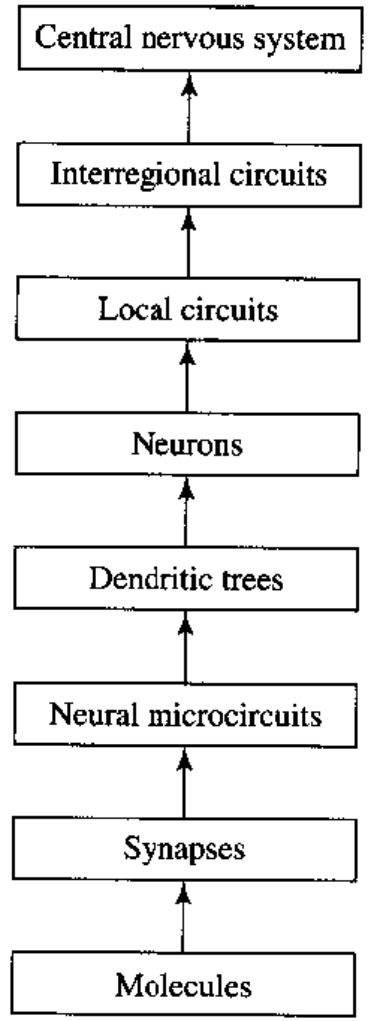


Figura: Neurona biológica [Haykin(2001)]

El cerebro humano

Estructura

- Segundo nivel (circuitos locales):
  - Un circuito local (1mm) contiene muchas neuronas del mismo o de distinto tipo, y realizan operaciones características localizadas en distintas zonas del cerebro.
- Tercer nivel:
  - Circuitos interregionales involucran rutas de datos, columnas, mapas topográficos de areas que involucran múltiples regiones localizadas en distintas partes del cerebro.
  - Los mapas topográficos se organizan para responder a estímulos de información sensorial
- Cuarto nivel:
  - El sistema nervioso central



El cerebro humano

Zonas del cortex cerebral

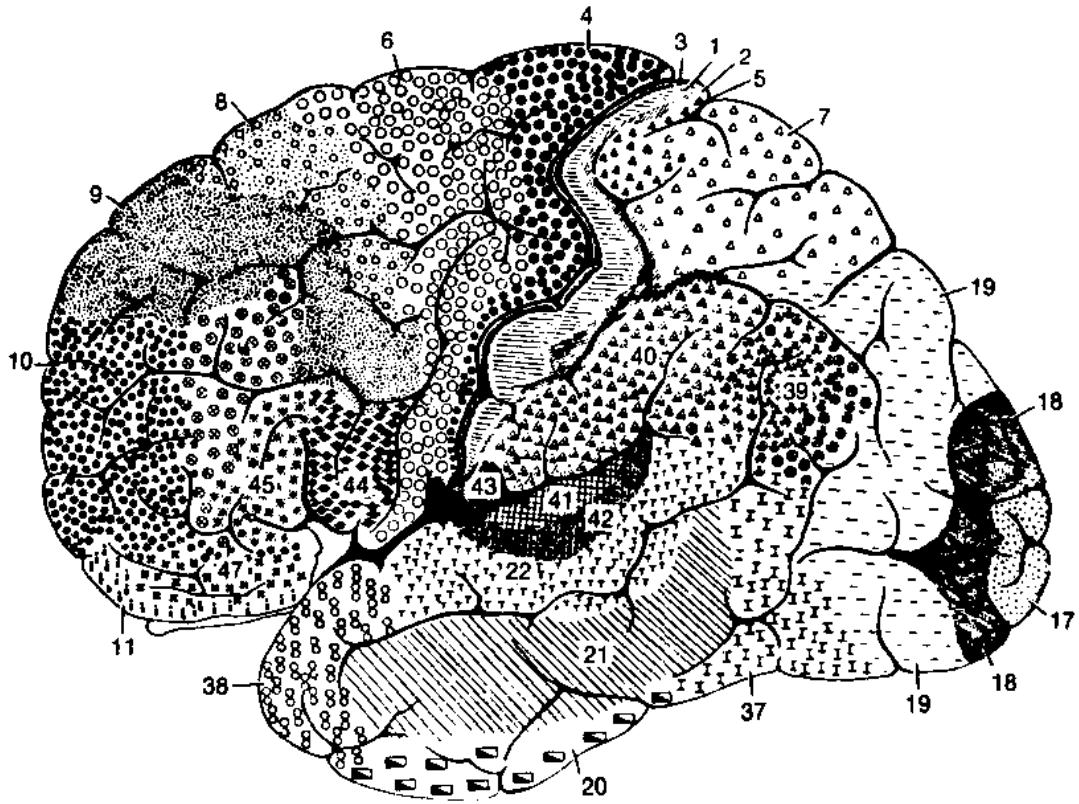


Figura: Mapa de Brodmann basasda en citoarquitectura de la corteza cerebral [Brodal(1981)]. Corteza motora primaria area 4, corteza somatosensorial asociativa area 6, area sensitiva primaria area 3,1 y 2, corteza visual primaria area 17, corteza visual asociativa area 18, corteza visual asociativa area 19

Redes neuronales artificiales

- Una Red Neuronal es básicamente una estructura de procesamiento de información con las siguientes características:
  - Procesamiento paralelo y distribuido.
  - Constituida por elementos de procesamiento interconectados por canales unidireccionales de información denominados conexiones.
  - Cada elemento de procesamiento tiene una conexión de salida con diferentes ramas (tantas como sea necesario) que transmiten la misma señal.
  - La señal de salida puede ser de un tipo matemático cualquiera.
  - Todo el procesamiento de la información es local y depende solamente de los valores de las entradas al elemento y de los posibles valores almacenados en la memoria local.

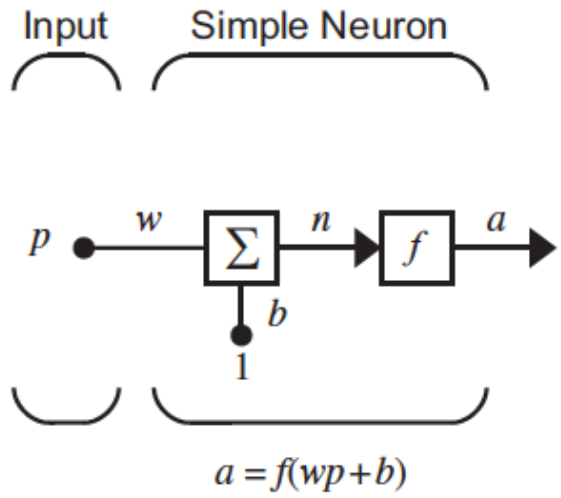
Redes neuronales artificiales

Ventajas

- No-linealidad. Una neurona artificial puede ser lineal o no-lineal. Una red neuronal, que interconecta neuronas no lineales es nolineal. Esta no-linealidad está además distribuida por toda la red. Esto es importante ya que los sistemas físicos subyacentes en muchos problemas son no lineales.
- Mapeo Entradas/Salidas . En el modelo de aprendizaje denominado aprendizaje supervisado la red es entrenada aplicando un conjunto de ejemplos de entrenamiento. Cada ejemplo consta de una señal de entrada (o entradas) y la respuesta de salida deseada para la red. En el aprendizaje la red neuronal construye un mapeo de entrada/salida para el problema.
- Adaptividad. Las redes neuronales son muy readaptables con pequeñas variaciones de sus pesos sinápticos a cambios en el entorno de operación. Para esto basta con un pequeño reentrenamiento adicional.
- Tolerantes a fallos. Una red neuronal, implementada en hardware, tiene potencial para ser tolerante a fallos ya que su rendimiento se degrada suavemente bajo circunstancias adversas.

Neurona de una entrada

- En este modelo simplificado:
  - la entrada a la neurona p es multiplicada por un peso w y enviada al sumador
  - la segunda entrada es un 1 multiplicado por un bias, offset o sesgo b y enviada al sumador
  - al resultado obtenido en el sumador $w.p + 1.b$ se le aplica una función de activación no-lineal en la mayoría de los casos $f(w.p + 1.b)$ y el resultado nos da la salida a de la neurona
  - Los parámetros w y b son los *parámetros ajustables* de la neurona.



Función de transferencia o activación

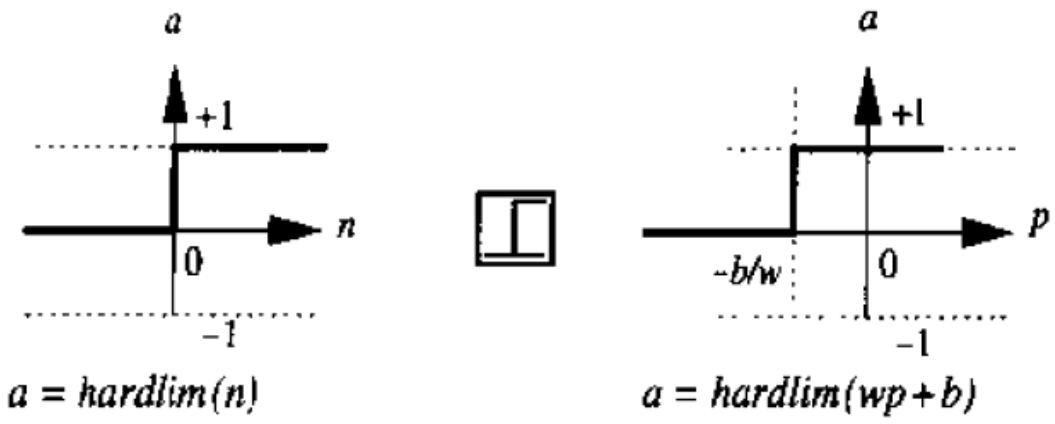
- Asocia un dominio de entrada, cuyo rango puede llegar hasta infinito, a un dominio de salida de rango previamente especificado.

$$a_j = f(\sum_{i=1}^{n} p_j \omega_{ji} - b_j \omega_{j0}) \tag{1}$$

- Entre las funciones de transferencia más usuales tenemos:
  - Lineal.
  - Rampa.
  - Escalón.
  - Sigmoide.

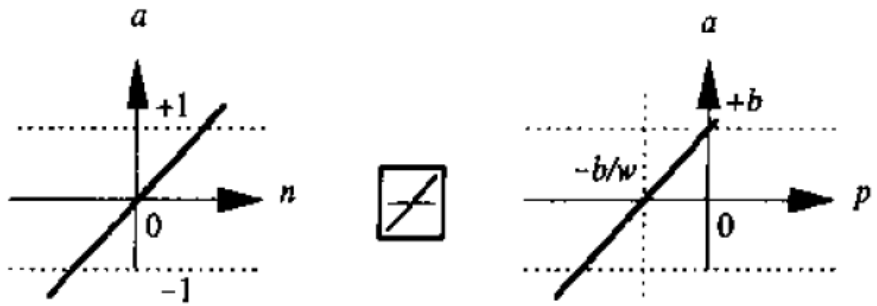
Función de transferencia

- La función de transferencia $f()$ de la neurona nos da la relación que existe entra la salida y la entrada.
- Puede ser de diversos tipos:
  - La denominada **hard limit** funciona del siguiente modo:



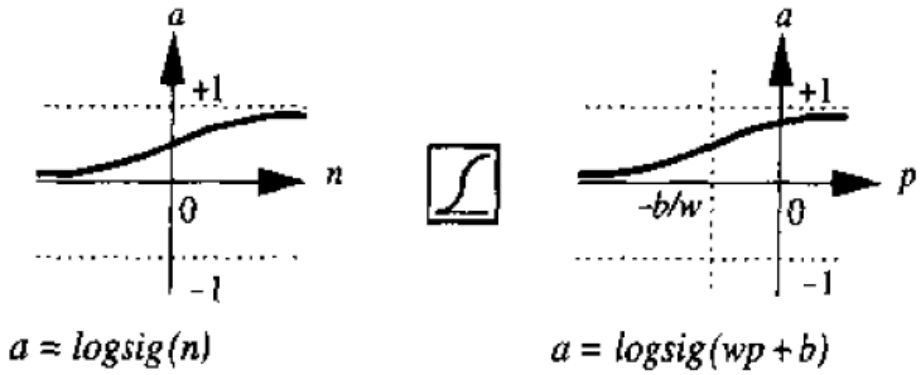
Función de transferencia

- Otros tipos son:
  - Lineal (tipo adaline):



  - Log-sigmoide (muy usada en redes multicapa por ser derivable).

$$a = \frac{1}{1 + e^{-n}} \tag{2}$$



Función de transferencia

Tabla

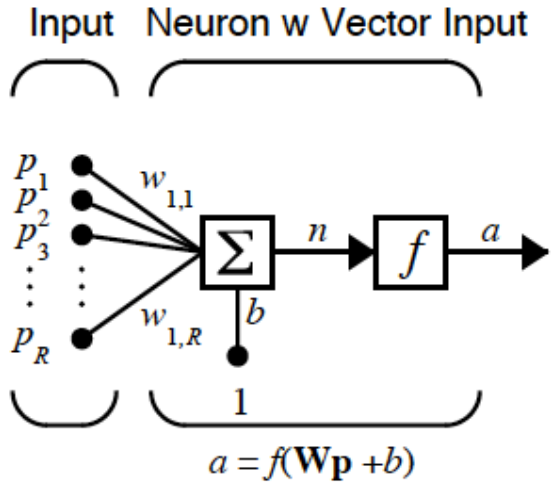
Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1+e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons		compet

Neurona multi-entrada

- Estructura elemental:
  - $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ es el vector de señales de entrada a la neurona.
  - $\mathbf{W}_j = (\omega_{j1}, \omega_{j2}, \ldots, \omega_{jn})$ es el vector de pesos que ponderan las señales de entrada
  - b es un parámetro que funciona como umbral de activación interno de la neurona.
  - $f()$ función de activación.
- El funcionamiento básico es similar:
  - La suma de los estímulos de entrada es:

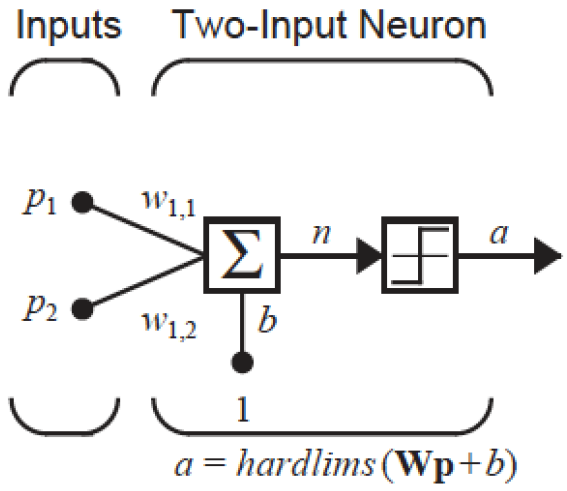
    $$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b = \mathbf{Wp} + b$$

  - La salida es $a = f(\mathbf{Wp} + b)$



Neurona artificial

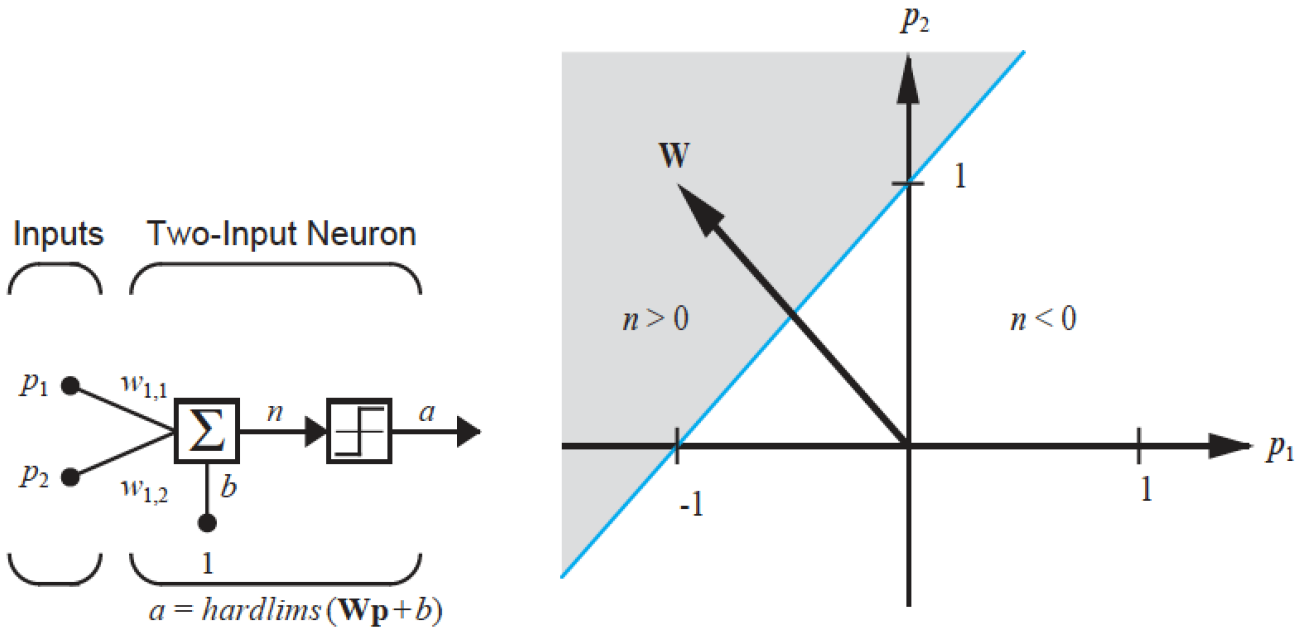
Qué hace?



- Función básica.
  - Separa el espacio de entrada en dos zonas diferentes, a cada una de las cuales les asigna un valor, que típicamente suele tomarse 0 y 1.
  - La zona de transición entre las zonas con valor de salida 0 y 1 depende de los coeficientes que ponderan las entradas y de la función de activación escogida.
  - Por tanto, una neurona es capaz de separar (discriminar) un cierto espacio de entrada en dos conjuntos linealmente separables.

Neurona artificial

Ejemplo: perceptrón



- Una neurona tipo perceptrón puede clasificar vectores de entrada en dos categorías. Por ejemplo en el perceptron de dos entradas, si $w_{1,1} = -1$ y $w_{1,2} = 1$ entonces

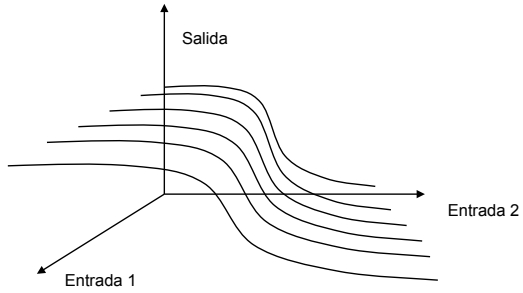
$$a = hardlims(n) = hardlims([-1\ 1]\mathbf{p} + b)$$

- Si el producto interno de la matriz de pesos (un vector fila) por el vector de entrada es mayor o igual que $-b$, la salida es 1 y si es menor que $-b$, la salida es -1. Esto divide el espacio de entrada en dos partes . La línea azul son los puntos para los que la entrada n de la red es igual a 0.

$$n = [-1\ 1]\mathbf{p} - 1 = 0$$

Superficie de decisión

Salida de una neurona con dos entradas y una salida



- La propiedad clave de un perceptrón de una neurona es su capacidad para separar los vectores de entrada en dos categorías. La frontera de decisión entre las categorías viene dada por

$$\mathbf{Wp} + b = 0$$

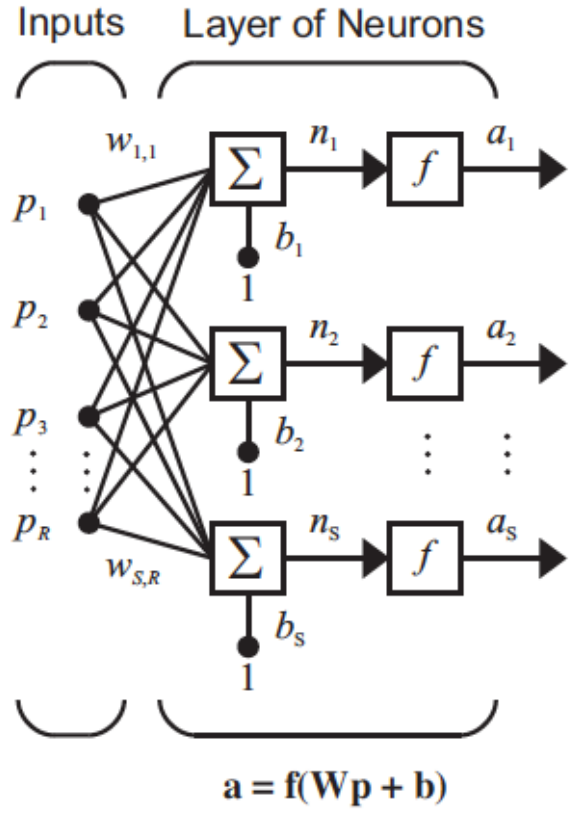
- Como el contorno de decisión debe de ser lineal, los perceptrones de una capa sólo pueden usarse para reconocer patrones que sean linealmente separables.
- Los problemas de clasificación no suelen ser linealmente separables (superficie de separación muy compleja en la mayoría de los casos).
- Para resolverlo se utilizan conjuntos de neuronas dispuestas en capas, de forma que cada neurona forme un borde lineal y que como resultado de la combinación de todas ellas obtenemos la superficie de decisión deseada.

Capa de neuronas

- Estructura elemental de la capa:
  - R es el número de elementos en el vector de entrada
  - S es el número de neuronas en la capa.
  - Los pesos son la matriz $\mathbf{W}$:

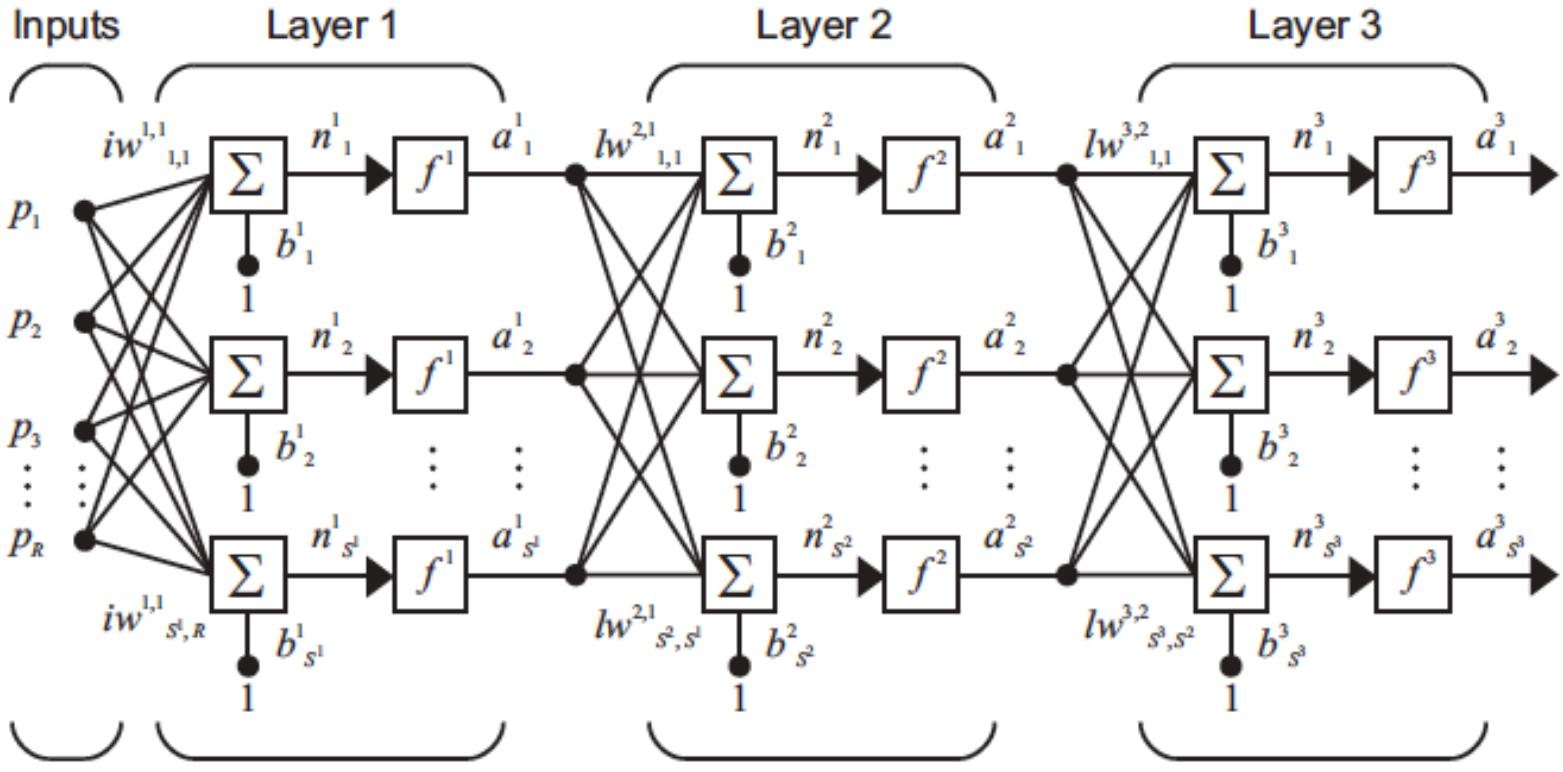
$$W = \begin{bmatrix} \omega_{11}(x) & \omega_{12}(x) & \dots & \omega_{1R}(x) \\ \omega_{21}(x) & \omega_{22}(x) & \dots & \omega_{2R}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{S1}(x) & \omega_{S2}(x) & \dots & \omega_{SR}(x) \end{bmatrix} \quad (3)$$

  - $\mathbf{b}$ es el vector de bias (uno por neurona de la capa)
  - $f()$ es la función de transferencia.
  - $\mathbf{a}$ es el vector de salida, $\mathbf{a} = f(\mathbf{W}.\mathbf{p} + \mathbf{b})$.



Redes multicapa

Estructura



$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p}+\mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1+\mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2+\mathbf{b}^3)$$

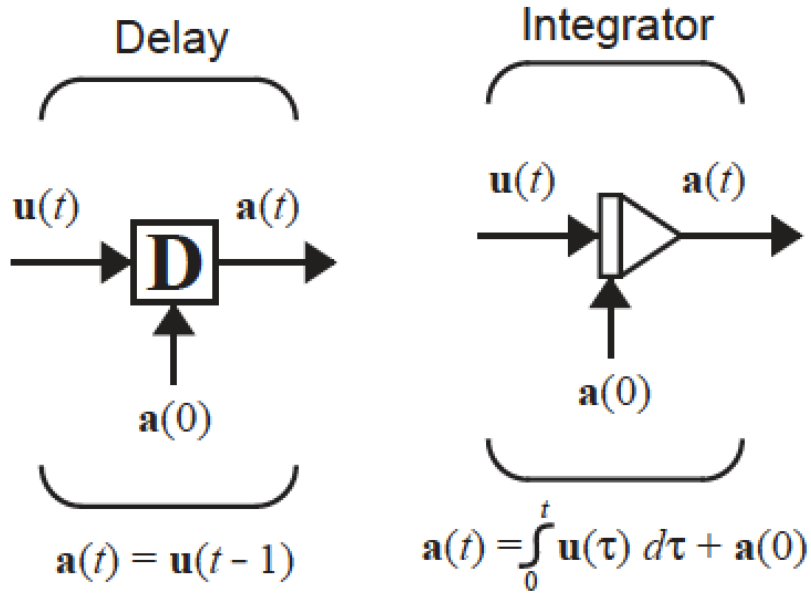
$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Redes multicapa

- La red anterior tiene:
  - R^1 entradas, S^1 neuronas en la primera capa, S^2 neuronas en la segunda capa, etc
  - La matriz $\mathbf{W}^2$ es de dimensiones $S^2 \times S^1$
  - La entrada a la capa 2 es la salida de la capa 1 es decir $\mathbf{a}^1$ y la salida de la capa 2 es $\mathbf{a}^2$.
  - La capa que produce la salida denomina capa de salida (output layer) las demás se denominan capas ocultas (hidden layers).
  - En la red de la figura la capa de salida es la 3 y las capas 1 y 2 son capas ocultas.
  - La arquitectura se suele especificar como: $R - S^1 - S^2 - .. - SM$.

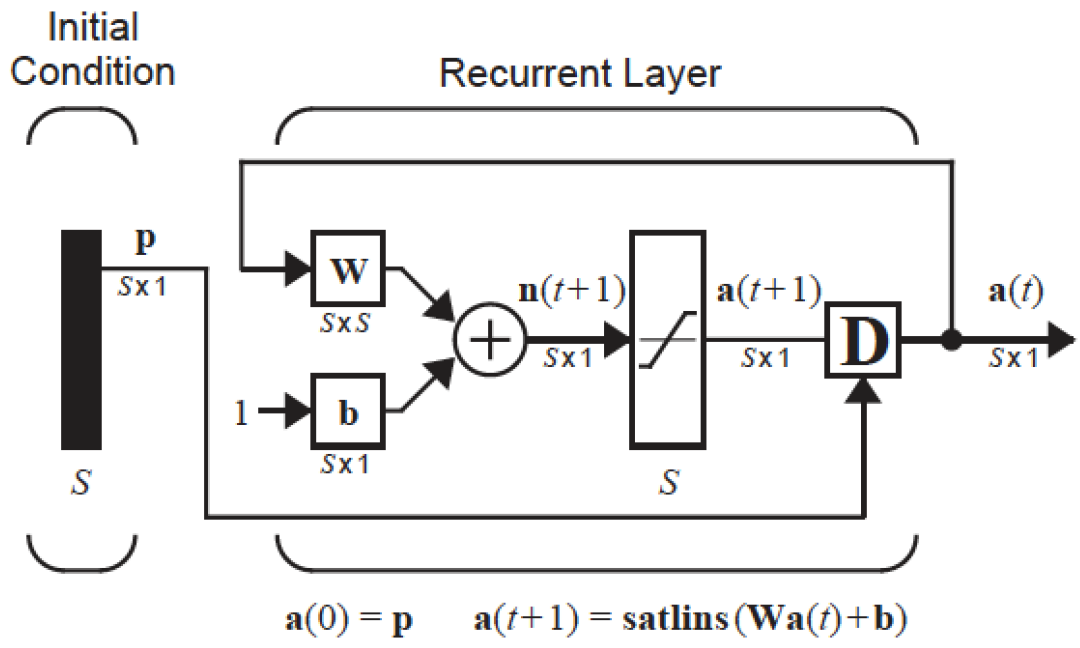
Redes recurrentes

- Utilizan retardos
  - $a(t) = u(t-1)$
  - Valor inicial: $a(0)$
- E integradores
  - $a(t) = \int_0^t u(\tau)\mathrm{d}\tau + a(0)$



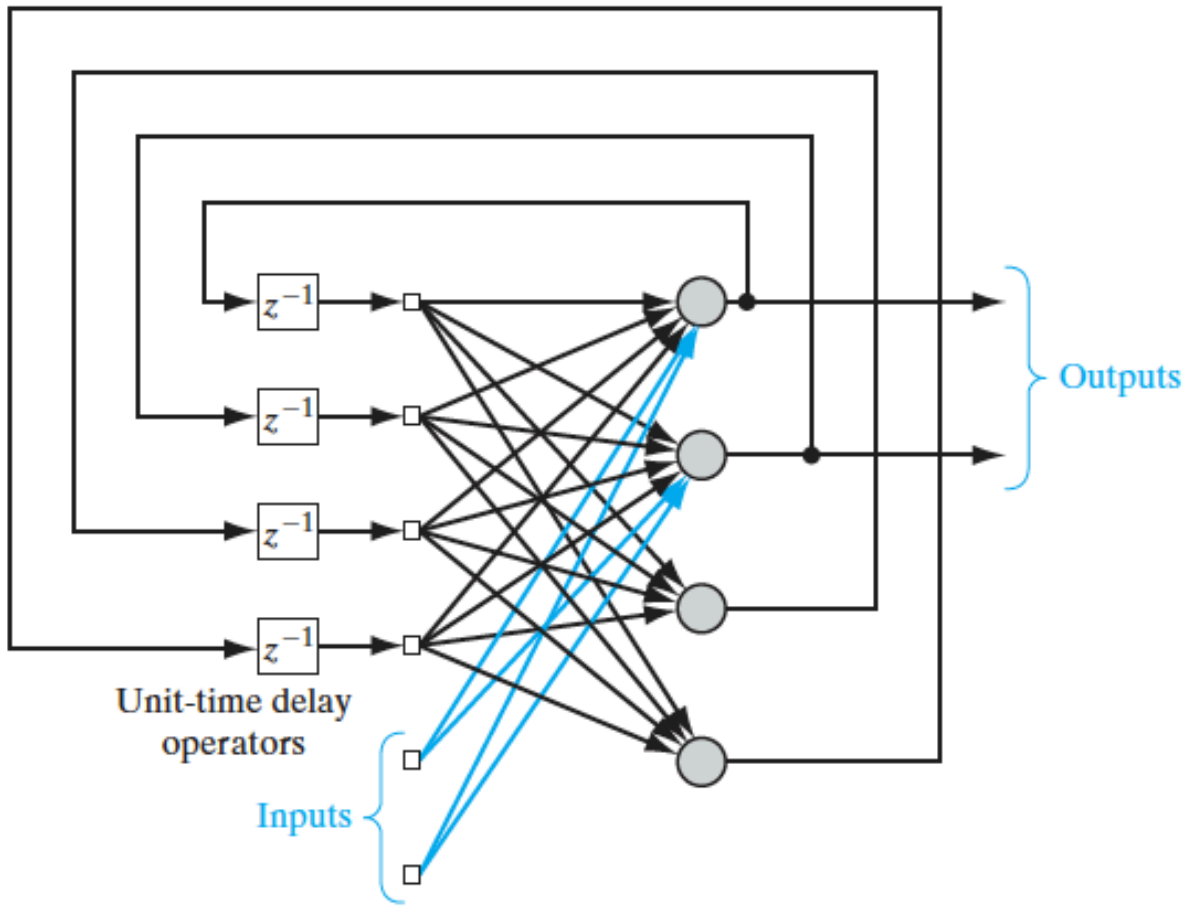
Redes recurrentes

- Es una red que presenta realimentación.
  - Es decir alguna de sus salidas está conectada a sus entradas.
  - Presentan diferencias notables con las de flujo lineal o feedforward.



Redes recurrentes

- Ejemplo: Red neuronal recurrente con neuronas ocultas



Representación de la información

- Cómo se representa la información dentro de una red neuronal? La respuesta precisa a esta pregunta es muy compleja. Pero hay algunas reglas para la representación del conocimiento.
    - Regla 1. Entradas similares de clases similares deberían producir, por lo general, representaciones similares dentro de la red neuronal, y deberían por tanto ser clasificadas como pertenecientes a la misma clase [Camperos(2006)].

Representación de la información

Medidas de similaridad

- Existen diversas medidas de similaridad entre entradas.
  - Una muy común es la distancia Euclídea entre las entradas. Si $\mathbf{x}_i$ es el vector de entrada $\mathbf{x}_i : m \times 1$. Esto representa un pto en un espacio m-dimensional, y la distancia euclidea entre dos entradas, $\mathbf{x}_i$ y $\mathbf{x}_j$ es

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i, \mathbf{x}_j\| = \Big[\sum_{k=1}^{m}(x_{ik} - x_{jk})^2\Big]^{\frac{1}{2}}$$

    Cuanto más próximos estén $\mathbf{x}_i$ y $\mathbf{x}_j$, menor es la distancia Euclídea $d(\mathbf{x}_i, \mathbf{x}_j)$ y por tanto mayor es la similitud (similitud=$1/d_{\mathbf{x}_i\mathbf{x}_j}$).
  - La Regla 1 nos dice que: Si los vectores de entrada $\mathbf{x}_i$ y $\mathbf{x}_j$ son similares deben de ser asignados a la misma clase.

Representación de la información

Medidas de similaridad 2

- Otra medida de similaridad entre entradas es:
  - Una común es el producto escalar entre las entradas. Si $\mathbf{x}_i$ es el vector de entrada $\mathbf{x}_i : m \times 1$. Esto representa un pto en un espacio m-dimensional, y el producto escalar entre dos entradas, $\mathbf{x}_i$ y $\mathbf{x}_j$ es

$$(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j = \sum_{k=1}^{m} x_{ik} x_{jk}$$

el producto escalar $\mathbf{x}_i^T \mathbf{x}_j$ se define como la proyección del vector $\mathbf{x}_i$ y $\mathbf{x}_j$. Ambas medidas están relacionadas ya que

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j) = \mathbf{x}_i^2 - 2\mathbf{x}_i^T \mathbf{x}_j + \mathbf{x}_j^2$$

Representación de la información

Medidas de similaridad 3

- Otra medida de similaridad entre entradas es:
  - La distancia de Mahalanobis entre las entradas (distancia entre dos distribuciones Gaussianas).

Representación de la información

- Cómo se representa la información dentro de una red neuronal?.
  - Regla 2. Los elementos a ser categorizados como clases separadas deberían tener representaciones ampliamente separadas en la red (de acuerdo con la distancia de similaridad usada).
  - Regla 3. Si una característica particular es importante debería tener un número grande de neuronas involucradas en su representación en la red neuronal.
  - Regla 4. La información a priori debe ser integrada en el diseño de la red cuando esta esté disponibles, de forma que se simplifique el diseño de la red neuronal al no tener que aprenderse esa información.

Representación de la información

Regla 4

- Ventajas de incluir la información a priori.
  - Evidencia biológica.
  - Menos parámetros a ajustar.
  - Menos numero de ejemplos y tiempo de aprendizaje.
  - Acceleracion del proceso de la información.
  - Reducción de costos de implementación.
- Como se puede incorporar la información a priori.
  - Seleccionando la estructura de la red con conexiones locales (parcialmente conectadas)
  - Restrigiendo la selección de pesos.
  - Por procesamiento para extraer características.

Redes neuronales clásicas

- Veamos a continuación varios tipos de redes que nos permitan entender el funcionamiento general de una red:
- Redes de flujo lineal
  - Perceptrón
  - Adaline
- Redes recurrentes
  - Red Hamming
  - Red Hopfield

Perceptrón

- F. Rosenblatt [Bishop(2006)].

Frank Rosenblatt
1928–1969

Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minksy, whose objections were published in the book "Perceptrons", co-authored with Seymour Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

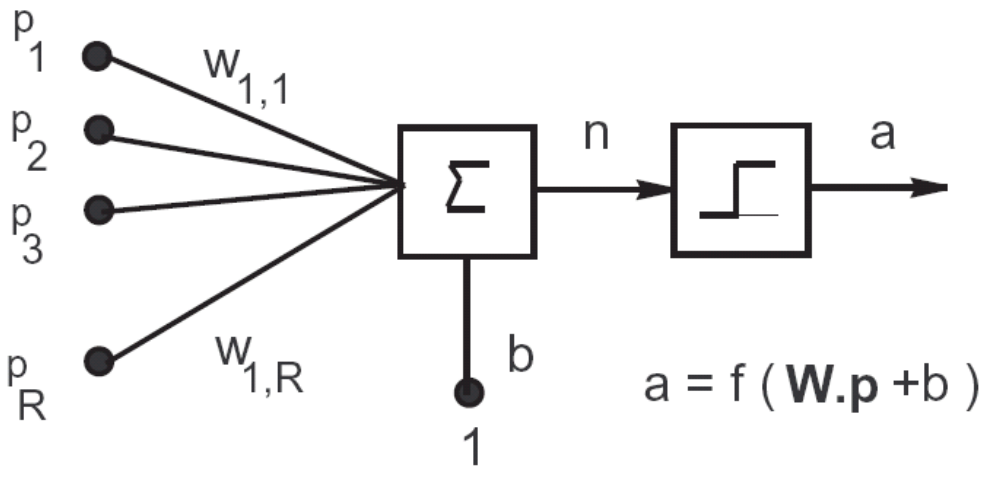
Perceptrón

- El primer perceptrón [Bishop(2006)].

Figure 4.8 Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focussed onto a 20×20 array of cadmium sulphide photocells, giving a primitive 400 pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of adaptive weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm.

Perceptrón

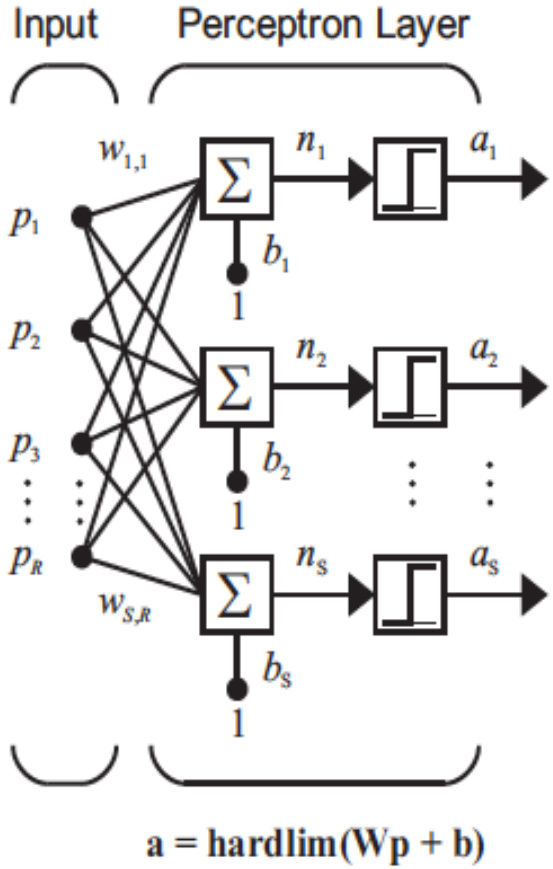
- El esquema básico de un perceptrón es el siguiente:



  - La función de salida divide el espacio de entrada en dos regiones por medio de un hiperplano, a una de las cuales el sistema le asigna salida 1, y al otro semiplano y a plano divisor se le asigna valor 0.
  - El bias *b* permite que el plano divisor no pase por el origen.

Perceptrón de 1 capa

Arquitectura



- Cada neurona divide el espacio de entradas en dos regiones. Es decir a cada región del espacio de entradas separada por los hiperplanos se le asocia una combinaciónón de valores diferentes de salidas (una codificación).

Formulación matemática

Problema de categorización / regresión

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$
- Buscamos encontrar una función $y = f(x)$ usando los datos de entrenamiento
- Sujeta a que f sea correcta sobre los test de datos
- Si las salidas buscadas $\{y_i\} = 1$ o 0 el problema es de categorización, y si la la salida $\{y_i\} \in \mathcal{R}$ es de regresión.

. . . pero que clase de función?

Formulación matemática

Problema de categorización / regresión

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$
- Buscamos encontrar una función $y = f(x) \in \mathcal{H}$ usando los datos de entrenamiento

  Donde $\mathcal{H}$ es una cierta clase de funciones por hipótesis (lineal por ejemplo).
- Sujeta a que f sea correcta sobre los test de datos

Formulación matemática

Problema de categorización / regresión

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Buscamos encontrar una función $y = f(x) \in \mathcal{H}$ usando los datos de entrenamiento
- Sujeta a que f sea correcta sobre los test de datos i.i.d. de una distribución $\mathcal{D}$.

  Suele asumirse también que los datos de test y los de entrenamiento tienen la misma distribución y son i.i.d. (independientes e idénticamente distribuidos).

Formulación matemática

Problema de categorización / regresión

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Buscamos encontrar una función $y = f(x) \in \mathcal{H}$ usando los datos de entrenamiento
- Sujeta a que f sea correcta sobre los test de datos, i.i.d. de una distribución $\mathcal{D}$.

  Pero para decir que son correctos que medida de prestaciones tomamos?.

Formulación matemática

Problema de categorización / regresión

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Buscamos encontrar una función $y = f(x) \in \mathcal{H}$ usando los datos de entrenamiento
- Sujeta a que la expected loss sea pequeña

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

Existen diferentes funciones de coste o pérdidas:

- $0-1$ loss: $l(f, x, x) = \mathbb{I}[f(x) \neq y]$ y $L(f) = Pr[f(x) \neq y]$.
- l_2 loss: $l(f, x, x) = [f(x) - y]^2$ y $L(f) = \mathbb{E}[f(x) - y]^2$.

Formulación matemática

Problema de categorización / regresión

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Buscamos encontrar una función $y = f(x) \in \mathcal{H}$ usando los datos de entrenamiento
- Sujeta a que la expected loss sea pequeña

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

Cómo usamos los datos de entrenamiento?:

- Buscamos encontrar una función $y = f(x) \in \mathcal{H}$ que minimice las pérdidas o coste empíricos (experimentales)

$$\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} l(f, x_i, y_i)$$

A modo de resumen

Proceso de aprendizaje

- Obtención de los datos y extracción de las features (características)
- Construción del modelo: se elige la clase de hipótesis H y la función de pérdidas l.
- Optimización: minimizamos las pérdidas empíricas (las generadas con los datos experimentales)

Estos tres procesos hay que seguirlos tanto si se usan redes neuronales como otro método.

Regresión lineal

Formulación

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Buscamos una función $y = f_w(x) = w^T x$ que minimice:

$$\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

  - La función $f_w(x)$ es la clase de hipótesis H
  - La función de pérdidas o coste es la l_2 también conocida como error medio cuadrático o MSE (mean square error)

Regresión lineal

Optimización

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Buscamos una función $y = f_w(x) = w^T x$ que minimice:

$$\hat{L}(f_w) = \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

- Si X es una matriz cuya fila i-ésima es x_i^T , e y es el vector $y = (y_1, \ldots, y_n)^T$

$$\hat{L}(f_w) = \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 = \frac{1}{n} \|Xw - y\|_2^2$$

Regresión lineal

Optimización

- Se deriva la función de coste a minimizar (se obtiene un gradiente en este caso) y se iguala a cero para obtener el mínimo:

$$\begin{aligned}
\bigtriangledown_w \hat{L}(f_w) = \bigtriangledown_w \frac{1}{n} \|Xw - y\|_2^2 = 0 \\
\bigtriangledown_w [(Xw - y)^T (Xw - y)] = 0 \\
\bigtriangledown_w [w^T X^T Xw - 2w^T X^T y + y^T y] = 0 \\
2X^T Xw - 2X^T y = 0 \\
w = (X^T X)^{-1} X^T y
\end{aligned}$$

Regresión lineal con bias

Formulación

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Buscamos una función $y = f_{w,b}(x) = w^T x + b$ para minimizar las pérdidas.
- Se reduce al caso anterior sin ibas
  - Haciendo $w' = [w; b],\ x' = [x; 1]$.
  - Entonces $y = f_{w,b}(x) = w^T x + b = (w')^T (x')$.

Por qué usar coste l_2

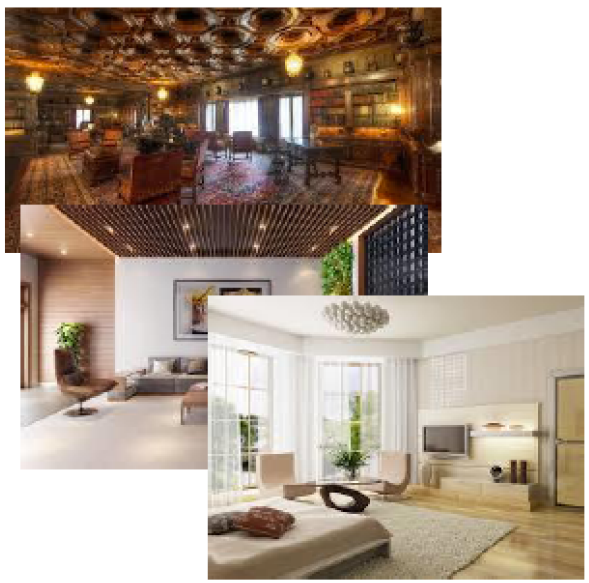
- Por qué no usar otra función de pérdidas ?
  - pérdidas l_1, $0-1$, exponencial, . . .,
- Empíricamente: es fácil de optimizar
  - Para el caso lineal: $w = (X^T X)^{-1} X^T y$
- Teóricamente: es una forma de codificar el conocimiento previo.

Pero:

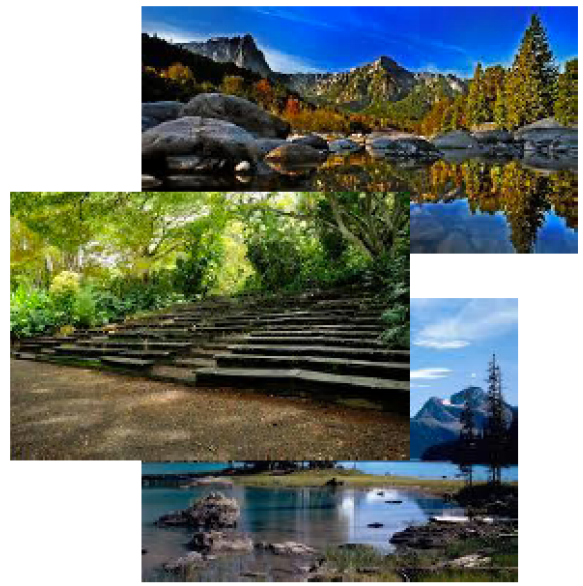
- Que tipo de conocimiento a priori?. .
- Como derivar las pérdidas?.

Classificación lineal

Ejemplo 1: clasificación de imágenes



Indoor



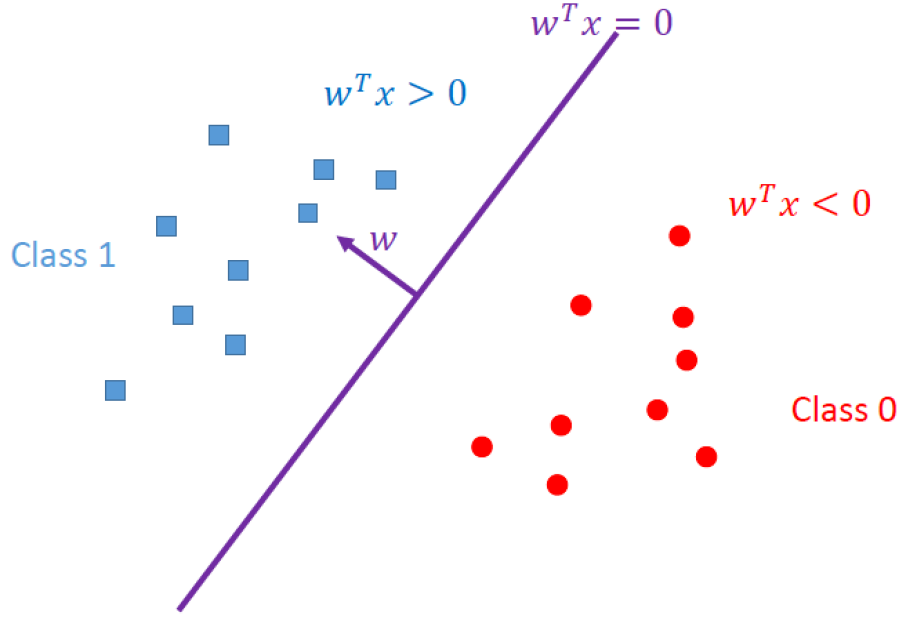
outdoor

Classificación lineal

Ejemplo 2: detección de spam

	#"$"	#"Mr."	#"sale"	...	Spam?
Email 1	2	1	1		Yes
Email 2	0	1	0		No
Email 3	1	1	1		Yes
...					
Email n	0	0	0		No
New email	0	0	1		??

Clasificación lineal



- La clasificación es una especie de resumen
- Es fácil de interpretar
- Es fácil de usar en toma de decisiones

Clasificación lineal

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución D.
- Hipótesis: $y = f_w(x) = w^T x$ (modelo lineal de $\mathcal{H}$) :
  - $y = 1$ si $w^T x > 0$.
  - $y = 0$ si $w^T x < 0$.
- Predicción: $y = step(f_w(x)) = step(w^T x)$

Clasificación lineal

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Buscamos una $y = f_w(x) = w^T x$ que minimice las pérdidas 0 -1

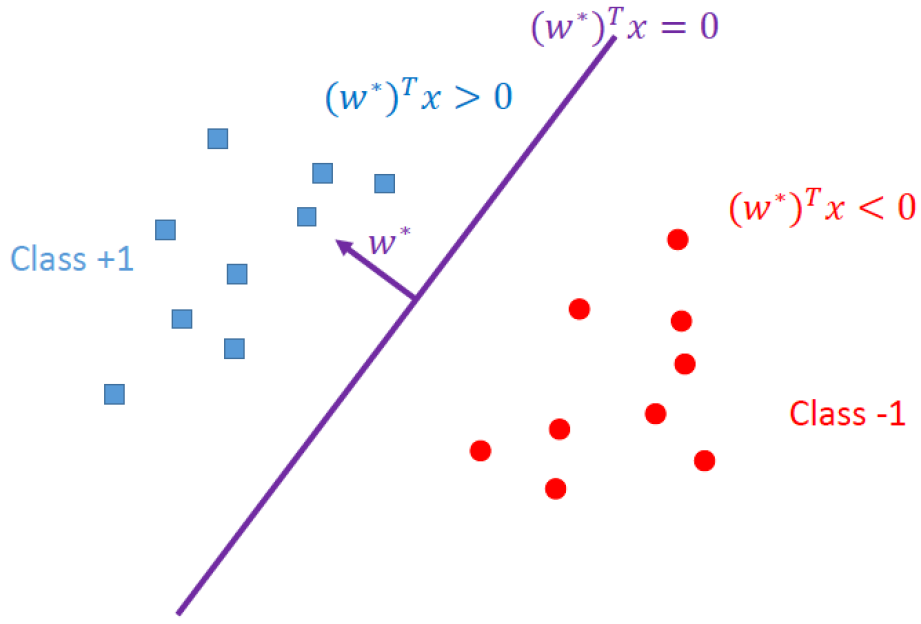
$$\hat{L}(f_w) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[step(w^T x_i) \neq y_i]$$

- Inconveniente: difícil de optimizar
  - Es NP-hard en el peor caso

Es esta idea la que se utilizó en el aprendizaje del perceptrón.

Clasificación lineal

Objetivo del Perceptrón



- La clasificación busca clasificar los datos en dos clases $[+1, -1]$. Este es el objetivo que se busca en el preceptrón.

Método de aprendizaje del perceptrón

Objetivo

- Dados unos datos de entrenamiento $\{(x_i, y_i)\} : 1 \leq i \leq n$ i.i.d. de una distribución $\mathcal{D}$.
- Hipótesis: $y = f_w(x) = w^T x$:
  - $y = +1$ si $w^T x > 0$.
  - $y = -1$ si $w^T x < 0$.
- Predicción: $y = sign(f_w(x)) = sign(w^T x)$
- Objetivo: minimizar el error de clasificación.

Método de aprendizaje del perceptrón

Primera idea

Asumamos por simplicidad que los x_i tienen longitud 1.

1. Se empieza con todos los vectores de pesos a cero $w_1 = 0$, i se iniciativa $t = 1$.
2. Dado un ejemplo x, se predice positivo si y sólo si $w_t.x > 0$.
3. En caso de error, se actualizan los pesos del siguiente modo:
   - Error en positivo: $w_{t+1} \leftarrow w_t + x$
   - Error en negativo: $w_{t+1} \leftarrow w_t - x$
   - $t \leftarrow t + 1$

Método de aprendizaje del perceptrón

Primera idea: corregir el error actual

- Si hay error en un ejemplo positivo.

$$w_{t+1}^T x = (w_t + x)^T . x = w_t^T x + x^T x$$

- Si hay error en un ejemplo negativo

$$w_{t+1}^T x = (w_t - x)^T . x = w_t^T x - x^T x$$

Teorema del perceptrón

- Supongamos que existe un w^* que clasifica correctamente $\{(x_i, y_i)\}$ (no necesitan ser i.i.d).
- Sin pérdida de generalidad, todo x_i y w^* tienen longitud 1, de forma que la distancia mínima de cualquier ejemplo a la frontera de decisión es

$$\gamma = \min_i |(w^*)^T . x_i|$$

- Entonces el perceptrón comete como mucho $(\frac{1}{\gamma})^2$ errores, que no depende de n, la longitud de la secuencia de datos.

Análisis

- Un vistazo a la cantidad $w_{t+1}^T w^*$.
- Primera observación $w_{t+1}^T w^* \geqslant w_t^T w^* + \gamma$.
  - Prueba, en caso de que el error sea en un ejemplo positivo de x

$$w_{t+1}^T w^* = (w_t + x)^T w^* = w_t^T w^* + x^T w^* \geqslant w_t^T w^* + \gamma$$

.

  - Si el error sea en un ejemplo negativo de x

$$w_{t+1}^T w^* = (w_t - x)^T w^* = w_t^T w^* - x^T w^* \geqslant w_t^T w^* + \gamma$$

.

Análisis

- Demos un vistazo ahora a la cantidad $||w_t^T||$.
- Segunda observación $||w_{t+1}^T||^2 \geqslant ||w_t^T|| + 1$.
  - Prueba, en caso de que el error sea en un ejemplo positivo de x

$$||w_{t+1}^T||^2 = ||w_t + x||^2 = ||w_t||^2 + ||x||^2 + 2w_t^T x$$

.

este último término $2w_t^T x$ es negativo dado que cometemos un error con x.

Método de aprendizaje

Supervisado

- El aprendizaje es de tipo **supervisado**, es decir se le presentan ejemplos de como debe de funcionar la red:

$$(p_1, t_1), (p_2, t_2), \ldots, (p_n, t_n) \tag{4}$$

  donde:
  - p_i son los ejemplos que se le presentan a la red.
  - t_i son las salidas que se desea que de la red para cada entrada de ejemplo.
- Si definimos el error que comete la red en un cierto instante como $\mathbf{e} = \mathbf{t} - \mathbf{a}$, el algoritmo de aprendizaje es el siguiente:

$$\Delta\omega = (\mathbf{t} - \mathbf{a}).\mathbf{p}^T = \mathbf{e}.\mathbf{p}^T \tag{5}$$

$$\Delta\mathbf{b} = (\mathbf{t} - \mathbf{a}) = \mathbf{e} \tag{6}$$

- Como la salida toma valores $(0, 1)$ el error será $(-1, 0, 1)$.

Perceptrón

Limitaciones

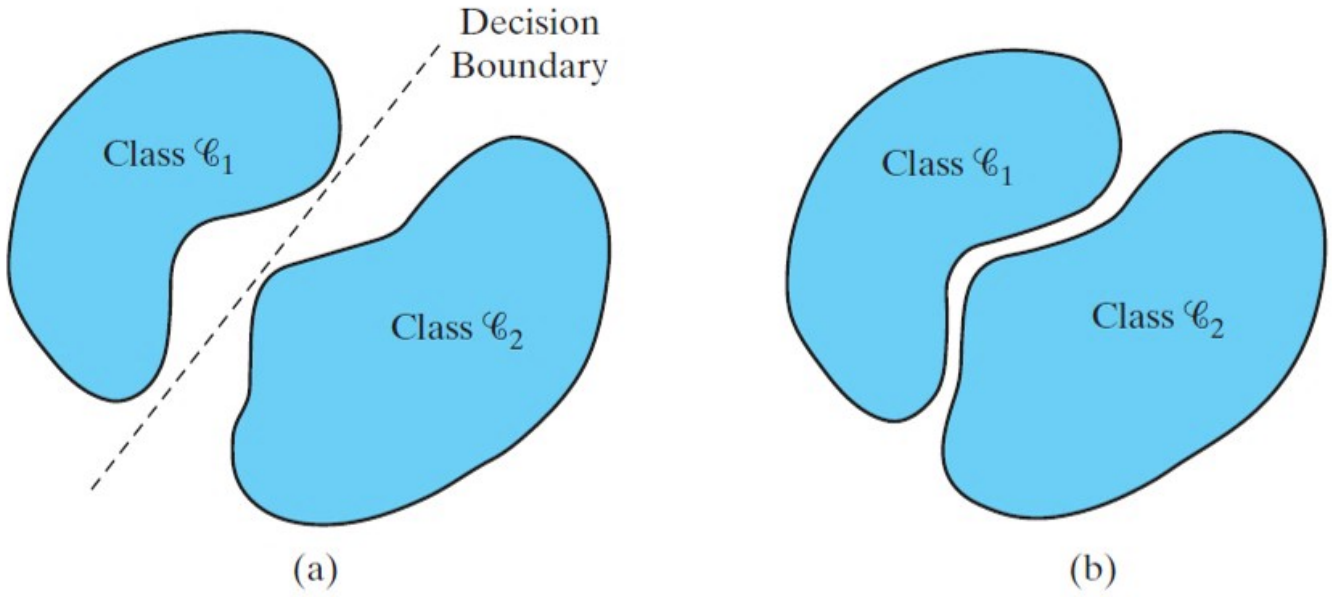
- La salida de una neurona sólo puede tomar valores $(0, 1)$.
  - El perceptrón sólo puede clasificar **problemas separables linealmente**.
  - Si el problema no es linealmente separable no llega a clasificar los valores de entrada correctamente.
  - La presencia de **outliers** en los vectores de entrada alarga considerablemente los tiempos de entrenamiento.
  - Para evitar este efecto puede alterarse un poco el método de aprendizaje de forma que se normalice el efecto del vector de entrada en el cambio de peso:

$$\Delta\omega = (\mathbf{t} - \mathbf{a}).\frac{\mathbf{p}^T}{\|\mathbf{p}\|} = \mathbf{e}.\frac{\mathbf{p}^T}{\|\mathbf{p}\|} \tag{7}$$

Perceptrón

Problemas linealmente separables

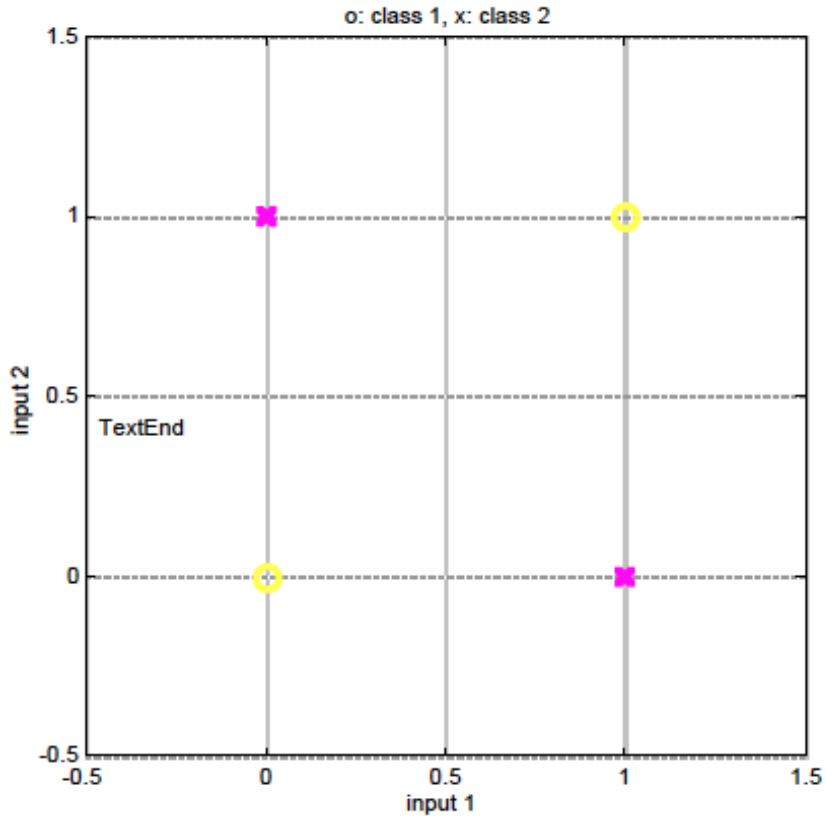
- Ejemplo de problemas linealmente separable y no-linealmente separable [Haykin(2001)].



Perceptrón

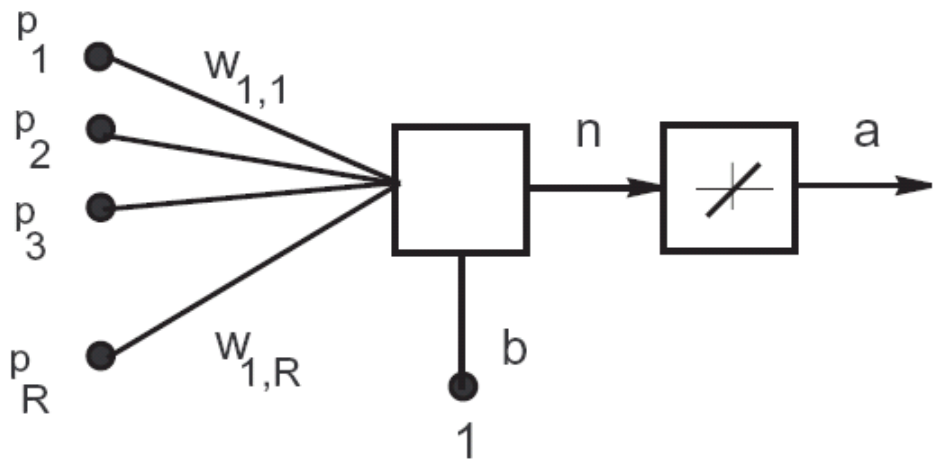
Problema no resoluble

- Ejemplo de problema no resoluble por un Perceptrón (X-Or).



Adaptive Linear Filters (ADALINE)

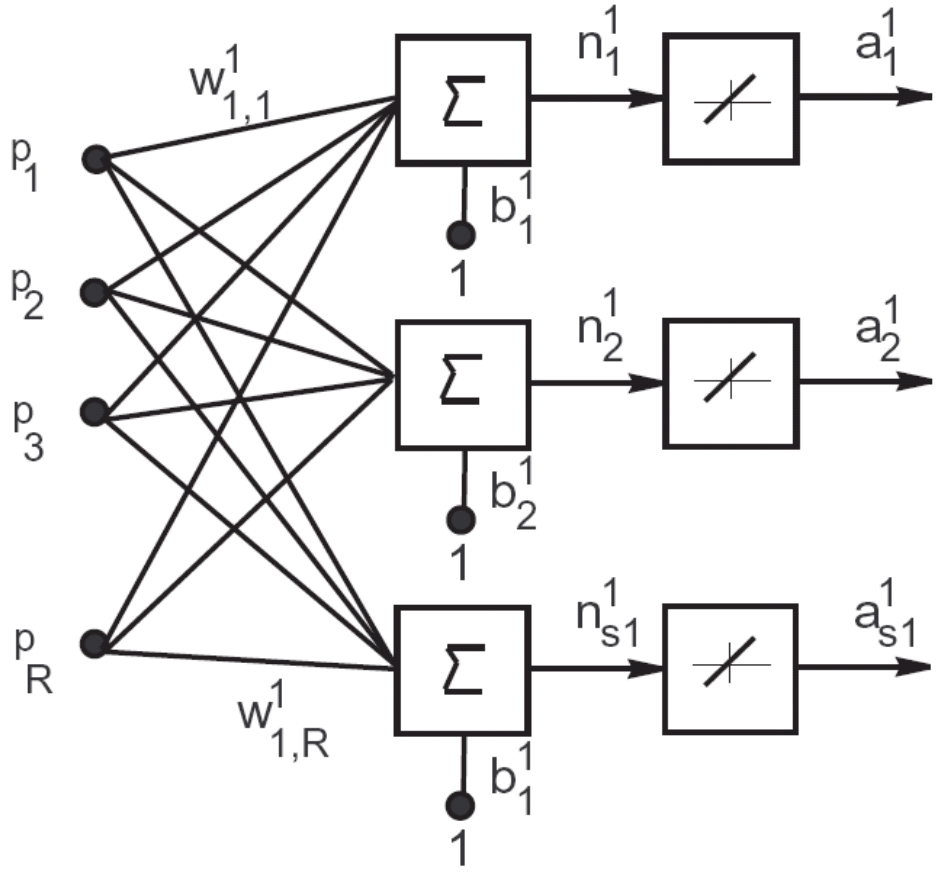
- Modelo de Neurona:



$$a = \mathbf{W}.\mathbf{p} + b$$

  - La función de salida divide el espacio de entrada en dos regiones por medio de un plano, pero la salida varía linealmente en función de la distancia al plano del vector de entrada.
  - El bias *b* permite que el plano divisor no pase por el origen.

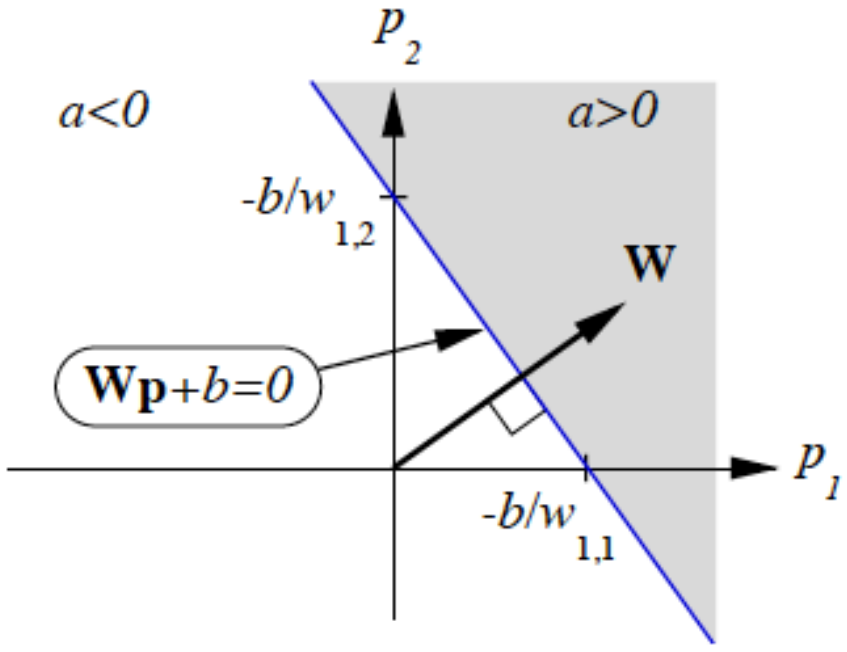
Adaptive Linear Filter: Arquitectura



Adaptive Linear Filter

Ejemplo

- División del espacio de entrada



$$a = \omega_{11} p_1 + \omega_{12} p_2 + b$$

Método de aprendizaje

Supervisado

- Utiliza un aprendizaje supervisado:

$$(p_1, t_1), (p_2, t_2), \ldots, (p_Q, t_Q) \tag{8}$$

  - El criterio a optimizar es **error cuadrático medio** (mse).

$$e_{MSE} = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q}(t(k) - a(k))^2 \tag{9}$$

  - Y el algoritmo de aprendizaje es el *método de Mínimos Cuadrados* (LMS least mean squares) que se denomina también **aprendizaje Widrow-Hoff**:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\mathbf{e}(k)\mathbf{p}^T(k) \tag{10}$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k) \tag{11}$$

  - Siendo α a la tasa de aprendizaje (learning rate).

Algoritmo de Widrow-Hoff

- Consiste básicamente en un método de aproximación de máxima pendiente.
  - Si se deriva el error cuadrático medio con respecto a los pesos y a los sesgos (bias) para la k-ésima iteración tenemos:

$$\frac{\partial e^2(k)}{\partial \omega_{1j}} = 2e(k)\frac{\partial e(k)}{\partial \omega_{1j}} \tag{12}$$

  - para $j = 1, 2, \ldots, R$ y

$$\frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b} \tag{13}$$

Algoritmo de Widrow-Hoff

- Por otra parte se tiene que:

$$\frac{\partial e(k)}{\partial \omega_{1j}} = \frac{\partial [t(k) - a(k)]}{\partial \omega_{1j}} = \frac{\partial}{\partial \omega_{1j}} [t(k) - (\mathbf{W}\mathbf{p}(k) + b)] \quad (14)$$

$$\frac{\partial e(k)}{\partial \omega_{1j}} = \frac{\partial}{\partial \omega_{1j}} \lfloor t(k) - \left(\sum_{i=1}^{R} \omega_{1j} p_i(k) + b \right) \rfloor \quad (15)$$

- Que simplificando queda respectivamente

$$\frac{\partial e(k)}{\partial \omega_{1j}} = -p_j(k) \quad (16)$$

$$\frac{\partial e(k)}{\partial b} = -1 \quad (17)$$

- De estas expresiones y las anteriores se obtiene la expresión que se indicó para el aprendizaje.

Algoritmo de Widrow-Hoff

- Y de aquí los cambios a realizar en los pesos y el bias son

$$2\alpha \mathbf{e}(k)\mathbf{p}^T(k) \tag{18}$$

$$2\alpha \mathbf{e}(k) \tag{19}$$

- Obteniéndose las expresiones del aprendizaje LMS o Widrow-Hoff

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha \mathbf{e}(k)\mathbf{p}^T(k) \tag{20}$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha \mathbf{e}(k) \tag{21}$$

Learning Rate (tasa de aprendizaje)

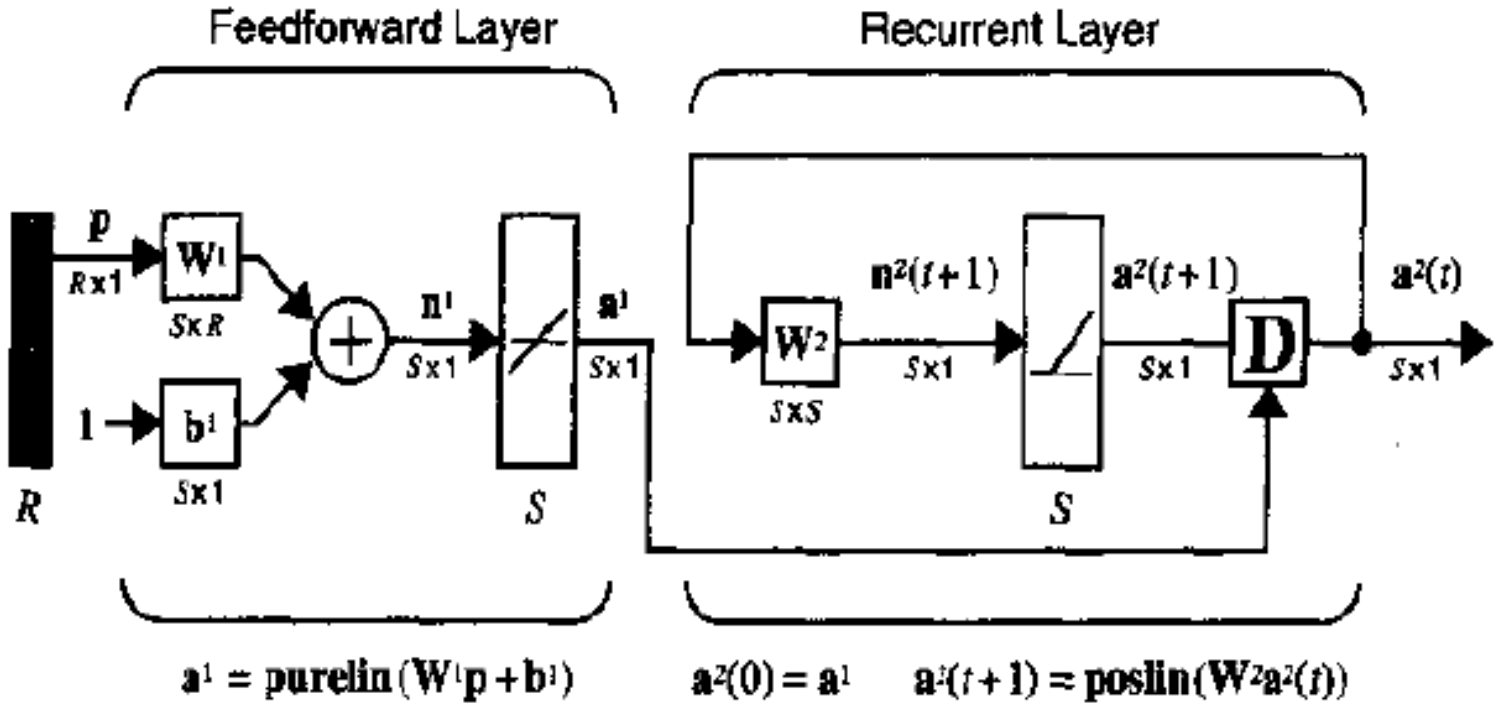
- El parámetro α denominado learning rate (tasa de aprendizaje) modula la velocidad de aprendizaje de la red.
  - Si es grande el sistema aprende muy rápidamente
    - Problema: puede conducirnos a inestabilidades que aumentan el error.
  - Si es muy pequeña el aprendizaje puede resultar demasiado lento.
    - Problema: lentitud.

Red Hamming

- Esta red se diseño para resolver explícitamente problemas de reconocimiento de patrones binarios (donde cada elemento del vector de entrada tiene sólo dos posibles valores, en el ejemplo siguiente +1 o -1)
- Tiene dos capas una capa es feedforward y otra capa es recurrente
- El objetivo de la red es determinar cual de los vectores prototipo es más próximo al vector de entrada.
- En la capa recurrente hay una neurona por cada patrón prototipo.
- Cuando la red converge sólo hay una neurona de salida con salida distinta de cero.

Red Hamming

- Está estructurada en dos capas:



Red Hamming

Funcionamiento

- *Capa feed-forward.*
  - Esta capa realiza una correlación, o producto interno entre cada patrón prototipo y el patrón de entrada.
  - Para hacer esta correlación cada fila de la matriz de pesos $\mathbf{W}^1$ se inicializa al un patrón de referencia

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \tag{22}$$

  - La capa feedforward usa una función de activación lineal y el vector de bias es igual a R, el número de elementos del vector de entrada (en el ejemplo es 3).
  - Y a la salida tenemos

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} \mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p} \\ \mathbf{p}_2^T\mathbf{p} \end{bmatrix} \tag{23}$$

Red Hamming

Funcionamiento

- Esta red se llama Hamming debido a que:
  - La neurona de la primera capa con la mayor salida corresponderá al patrón prototipo que es más corto en el sentido de la distancia Hamming al patrón de entrada (número de elementos distintos entre dos vectores , ejemplo: 2**1**73**8**96 y 2**23**3**7**96 es $d_H = 3$)
- Capa recurrente.
  - Recibe el nombre de capa competitiva .
  - Las neuronas de esta capa son inicializadas con las salidas de la capa feed-forward que indicaba la correlación entre la entrada y cada patrón.
  - Las neuronas compiten entre sí hasta que sólo queda un ganador, es decir que al final sólo queda una neurona con salida distinta de cero.
  - Las ecuaciones que describen el proceso competitivo son:

$$\mathbf{a}^2(0) = \mathbf{a}^1, \quad \text{Cond. inicial} \tag{24}$$

$$\mathbf{a}^2(t+1) = \mathit{poslin}(\mathbf{W}^2\mathbf{a}^2(t)) \tag{25}$$

Red Hamming

Funcionamiento

- Cont.
  - La función de activación *poslin* es lineal para los valores positivos y cero para los negativos.
  - La matriz de pesos tiene la forma:

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\epsilon \\ -\epsilon & 1 \end{bmatrix} \tag{26}$$

donde ϵ es un número menor que $1/(S-1)$, y S es el número de neuronas en la capa recurrente.

  - Una iteración de la capa recurrente opera de la forma:

$$\mathbf{a}^2(t+1) = \textit{poslin}\left(\begin{bmatrix} 1 & -\epsilon \\ -\epsilon & 1 \end{bmatrix} \mathbf{a}^2(t)\right) = \textit{poslin}\left(\begin{bmatrix} a_1^2(t) - \epsilon a_2^2(t) \\ a_2^2(t) - \epsilon a_1^2(t) \end{bmatrix}\right) \tag{27}$$

Red Hamming

Funcionamiento

- Para el ejemplo,
  - Dada la entrada:
  - Salida de la primera etapa, que es condición inicial para la segunda capa:
  - Salida después de la primera iteración de la segunda etapa:
  - Salida despuès de la 2 iteración de la segunda etapa,
  - A partir de aquí se mantiene estable la salida.

$$\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}.$$

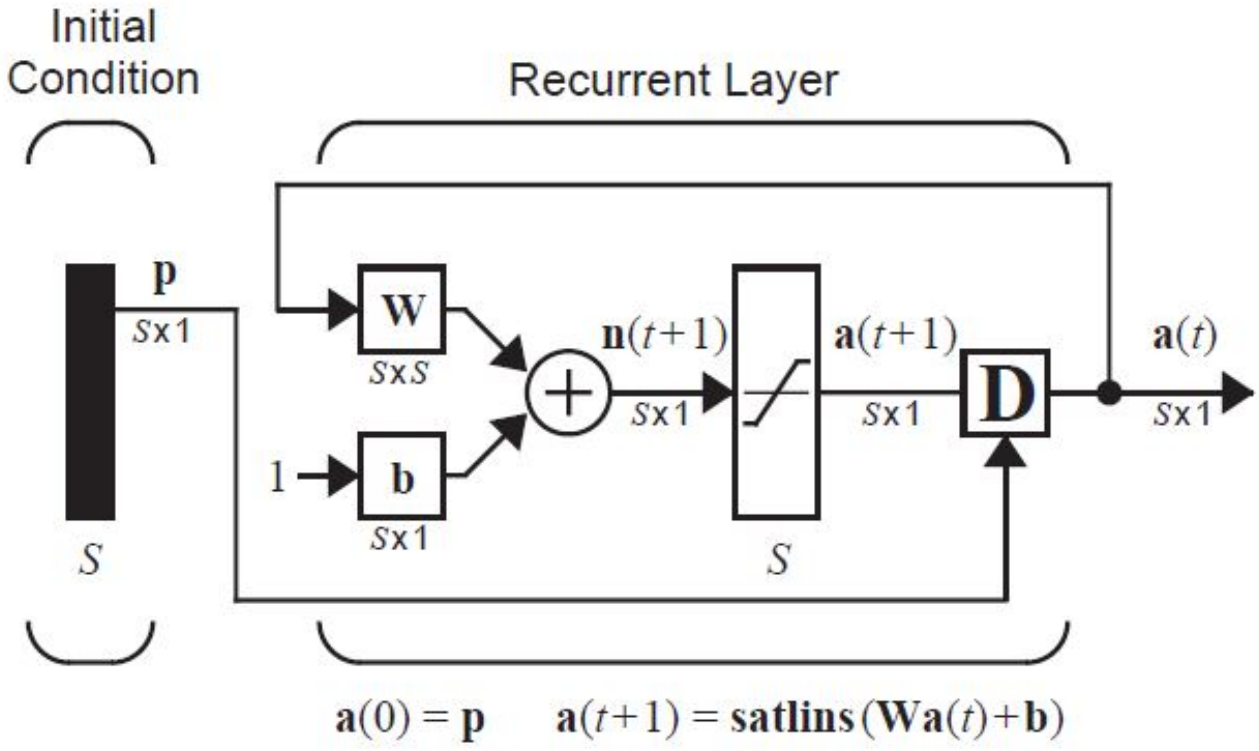
$$\mathbf{a}^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} (1+3) \\ (-1+3) \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix},$$

$$\mathbf{a}^2(1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(0)) = \begin{cases} \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix} \right) \\ \mathbf{poslin}\left(\begin{bmatrix} 3 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{cases}.$$

$$\mathbf{a}^2(2) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(1)) = \begin{cases} \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix} \right) \\ \mathbf{poslin}\left(\begin{bmatrix} 3 \\ -1.5 \end{bmatrix} \right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{cases}.$$

Red Hopfield

- Es una red recurrente capaz de hacer en una sola capa lo que hace la Hamming [Hagan(2014)]



Red Hopfield

- Aquí las neuronas de la red se inicializan con el vector de entrada.
  - La red opera hasta alcanzar la convergencia.
  - Cuando la red funciona correctamente la salida será uno de los patrones prototipo.
  - Las ecuaciones que describen el funcionamiento de la red son:

$$\mathbf{a}(0) = \mathbf{p} \tag{28}$$

$$\mathbf{a}(t+1) = satlin(\mathbf{W}\mathbf{a}(t) + \mathbf{b}) \tag{29}$$

  - El diseño de la matriz de pesos y la de bias es más complejo que en la de Hamming y no es del interés de este curso por lo que no se verá en detalle.

Red Hopfield

Funcionamiento

- Por ejemplo,
  - Con estos pesos y bias las funciones de cálculo son
  - y la salida toma los valores
  - sí recordamos que la entrada era
  - y la salida es el patrón más próximo de los dos definidos $p_1^T = (1 \ -1 \ -1)$ y $p_2^T = (1 \ 1 \ -1)$.

$$\mathbf{W} = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 1.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0.9 \\ 0 \\ -0.9 \end{bmatrix}$$

$$\begin{aligned} a_1(t+1) &= satlins(0.2a_1(t) + 0.9) \\ a_2(t+1) &= satlins(1.2a_2(t)) \\ a_3(t+1) &= satlins(0.2a_3(t) - 0.9) \end{aligned}$$

$$\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}.$$

$$\mathbf{a}(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(1) = \begin{bmatrix} 0.7 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(2) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(3) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

Bibliografía

- Arbib,M. A., 1989, The Metaphorical Brain 2: Neural Networks and Beyond, New York:Wiley-Interscience, p. 60. Livingstone MS.Mechanisms of direction selectivity in macaque V1. Neuron. 1998, 20(3):509-26.
- HAYKIN, Simon. Redes Neurais, princípios e práticas. Bookman. Porto Alegre, 2001.
- Brodal, A. (1981). Neurological anatomy. Relation to Clinical Anatomy.
- Bishop, C. M.. Pattern recognition and machine learning. Springer 2006.
- Sánchez Camperos, E. N., Alanís García, A. Y. (2006). Redes Neuronales: Conceptos fundamentales y aplicaciones a control automático. Edgar Nelson Sánchez Camperos y Alma Yolanda Alanís García.
- Demuth, H. B., Beale, M. H., De Jess, O., Hagan, M. T. (2014). Neural network design. Martin Hagan.
- Haykin, S. S. (2009). Neural networks and learning machines/Simon Haykin. New York: Prentice Hall.

- Fin L6