UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

ESCUELA POLITÉCNICA NACIONAL

REDES NEURONALES ARTIFICIALES

Jose Aguilar
Cemisid, Facultad de Ingeniería
Universidad de los Andes
Mérida, Venezuela
aguilar@ula.ve

INTRODUCCIÓN A LAS RNAs

¿CÓMO LA RED NEURONAL HUMANA ESTA DISEÑADA?

¿CÓMO EL CEREBRO PROCESA LA INFORMACIÓN?

¿CON QUÉ ALGORITMOS Y ARITMÉTICA EL CEREBRO CALCULA?

¿CÓMO PUEDE EL CEREBRO IMAGINAR?

¿CÓMO PUEDE EL CEREBRO INVENTAR?

¿QUÉ ES PENSAR?

¿QUÉ ES SENTIR?

INTRODUCCIÓN A LAS RNAs

El SISTEMA NEURONAL TIENE COMO PROPOSITO EL CONTROL CENTRALIZADO DE VARIAS FUNCIONES BIOLOGICAS.

POR EJEMPLO: EL ABASTECIMIENTO DE ENERGIA, FUNCIONES MOTORAS Y SENSORIALES, ETC.

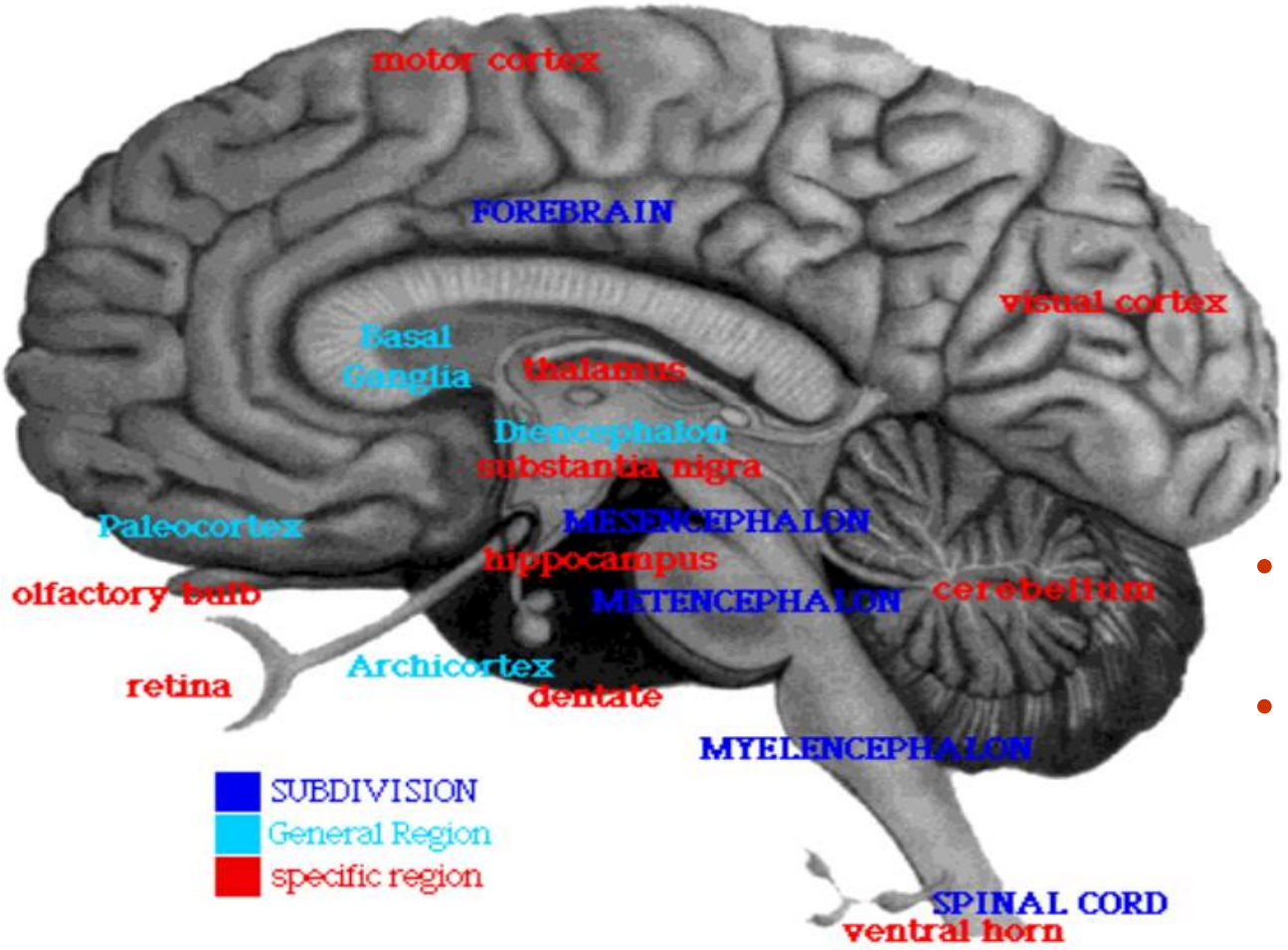
REDES NEURONALES ARTIFICIALES

Están basadas en el funcionamiento de las neuronas biológicas que componen el cerebro de los animales.

- El cerebro es el órgano central del sistema nervioso animal y **contiene** de 50 a 100 mil millones de neuronas con funciones especializadas, las cuales **transmiten señales** por medio de 1000 billones de **conexiones sinápticas**.

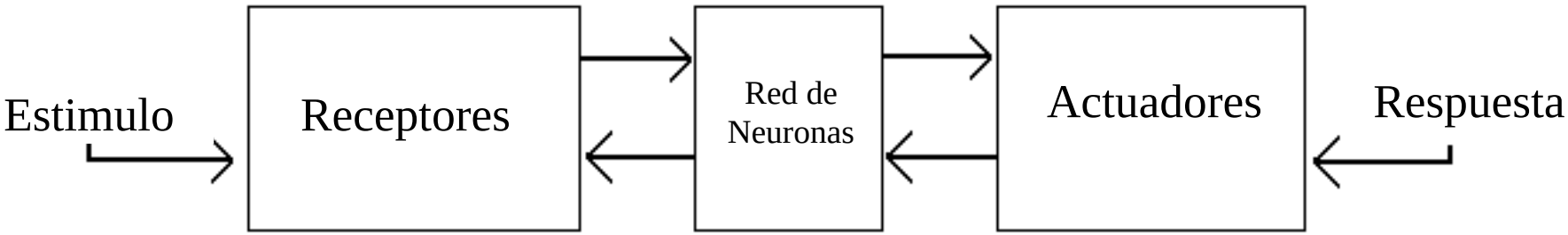
- **Reciben información** proveniente de **los sentidos**, la cual **es procesada** con una gran velocidad, mediante la combinación de la información almacenada, dando respuestas que **controlan y regulan** las acciones y reacciones **del cuerpo**.

Cerebro Humano



- 10^{11} Neuronas (procesadores)
- 1000 – 10000 conexiones por neurona

SISTEMA NERVIOSO



MODELO BIOLOGICO

SISTEMA NEURONAL

CONTROL CENTRALIZADO DE LAS FUNCIONES BIOLOGICAS

CEREBRO ~ 100 MIL MILLONES DE **NEURONAS** Y 10000 CONEXIONES POR NEURONA

MODELO BIOLOGICO

- **NEURONAS**: CELULAS VIVAS

- **CARACTERISTICAS:**
  - ELEMENTOS SIMPLES INTERCONECTADOS
  - FUNCIONAMIENTO EN PARALELO, ASINCRÓNICA Y NO ALGORÍTMICAMENTE
  - INTERACCIONES COMPLEJAS

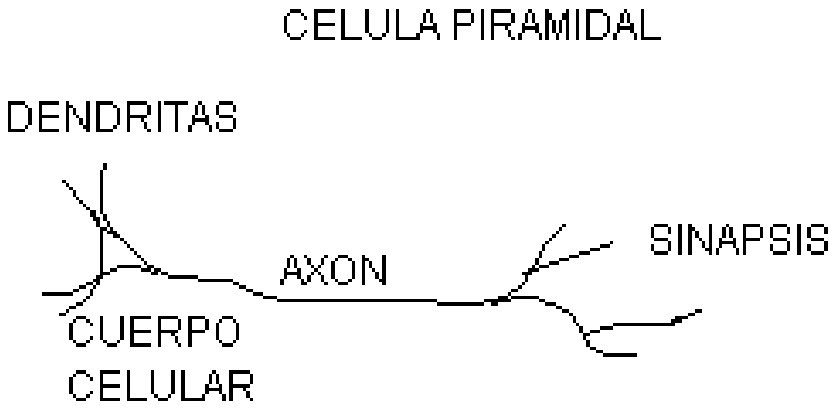
NEURONA

- **UNIDAD FUNDAMENTAL DEL SISTEMA NERVIOSO** ESPECIALIZADAS EN CIERTAS TAREAS

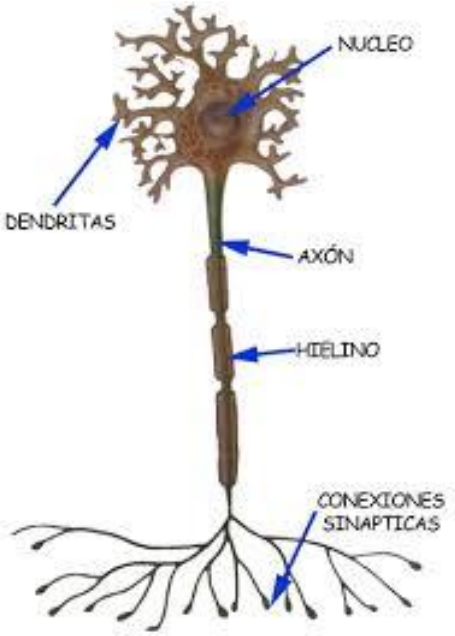
- **PROCESADOR DE SEÑALES ELÉCTRICAS** (DESCARGAS EN EL CUERPO CELULAR) **Y BIOQUÍMICAS** (NEUROTRANSMISORES)

- **RECIBE Y COMBINA SEÑALES** DESDE MUCHAS NEURONAS

NEURONA



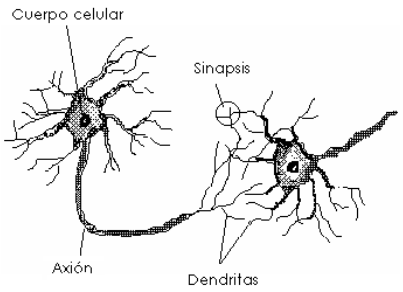
- **AXÓN:** LINEA DE TRANSMISIÓN
- **DENDRITAS:** ZONAS RECEPTORAS
- **SINAPSIS:** EXCITADORAS E INHIBIDORAS
- SEÑALES ELECTRICAS Y QUIMICAS



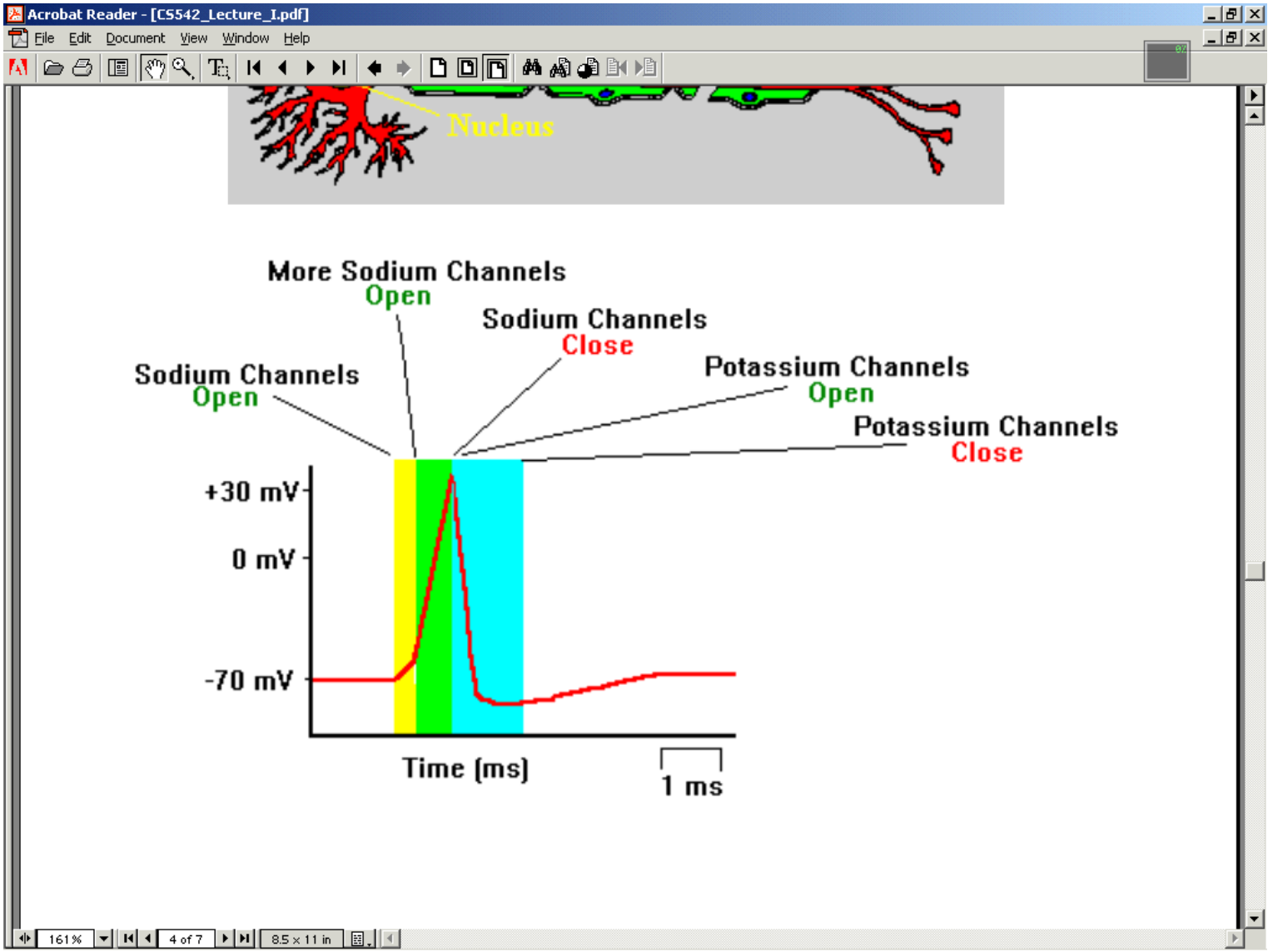
SINAPSIS

UNIDAD FUNCIONAL QUE INTERRELACIONA LAS NEURONAS

- **NEUROTRANSMISOR:** GENERA POLARIZACIÓN PARA LA MEMBRANA POSTSINÁPTICA
- **POTENCIAL POSTSINÁPTICO:** PUEDE SER POSITIVO (EXCITACIÓN) O NEGATIVO (INHIBICIÓN)



Neuronas


Acrobat Reader - [CS542_Lecture_I.pdf]
File Edit Document View Window Help
Nucleus
More Sodium Channels Open
Sodium Channels Close
Potassium Channels Open
Sodium Channels Open
Potassium Channels Close
+30 mV
0 mV
-70 mV
Time (ms)
1 ms
161%
4 of 7
8.5 x 11 in

REDES NEURONALES

- MUCHAS **CONEXIONES PARALELAS** ENTRE NEURONAS
- MUCHAS CONEXIONES PROVEEN **MECANISMOS DE RETROALIMENTACIÓN** PARA LAS NEURONAS
- EJECUTAN UN PROGRAMA QUE ES **DISTRIBUIDO**

- TIENEN PARTES **PRE-HECHAS** Y OTRAS QUE **EVOLUCIONAN**

CAPACIDADES RED NEURONAL

- Procesamiento paralelo
- Adaptativa
- Asociativa
- Auto-organización
- Generalización, clasificación, extracción y optimización
- Tolerancia a Fallas
- Operación en tiempo real

RN biológica

- Se nace con alguna **estructura neural**

- Se **crean nuevas conexiones y otras se gastan**; se desarrollan a través del **aprendizaje** propio de la etapa de crecimiento.

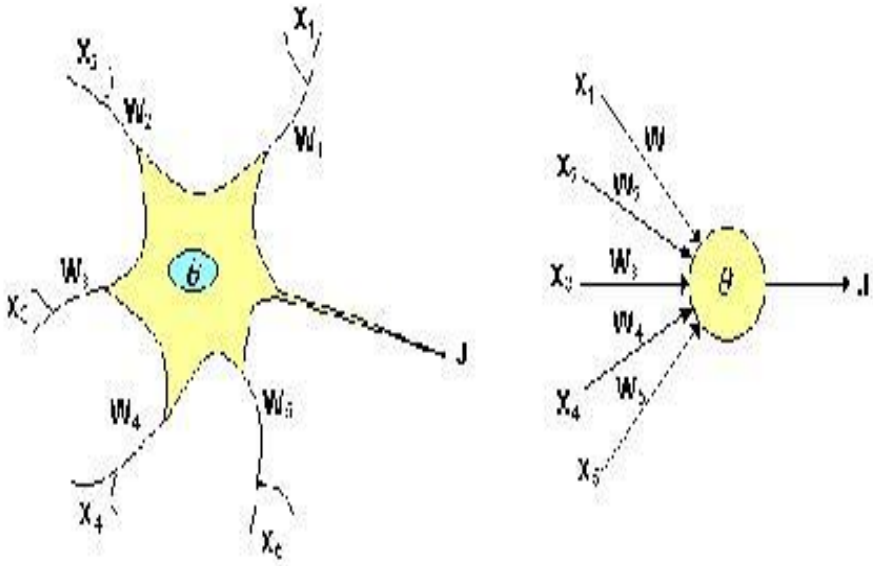
- La **estructura neural cambia a través de la vida**. Esos cambios consisten en **reforzamiento o debilitamiento** de las junturas sinápticas.

- Se forman **nuevas memorias** al modificar o reforzar algunas sinápsis.

- Por ejemplo, **memorizar** la cara de una persona que nos presentan, **consiste en alterar varias sinápsis**.

Red Neuronal Artificial

- ✓ Una **nueva forma de computación**, inspirada en el cerebro.
- ✓ Un **modelo matemático** compuesto por un gran **número de elementos de procesamiento,** eventualmente en niveles.
- ✓ Son **redes interconectadas masivamente** en paralelo de elementos simples y organización diversa.



Modelo de una RNA vs. el Modelo Biológico

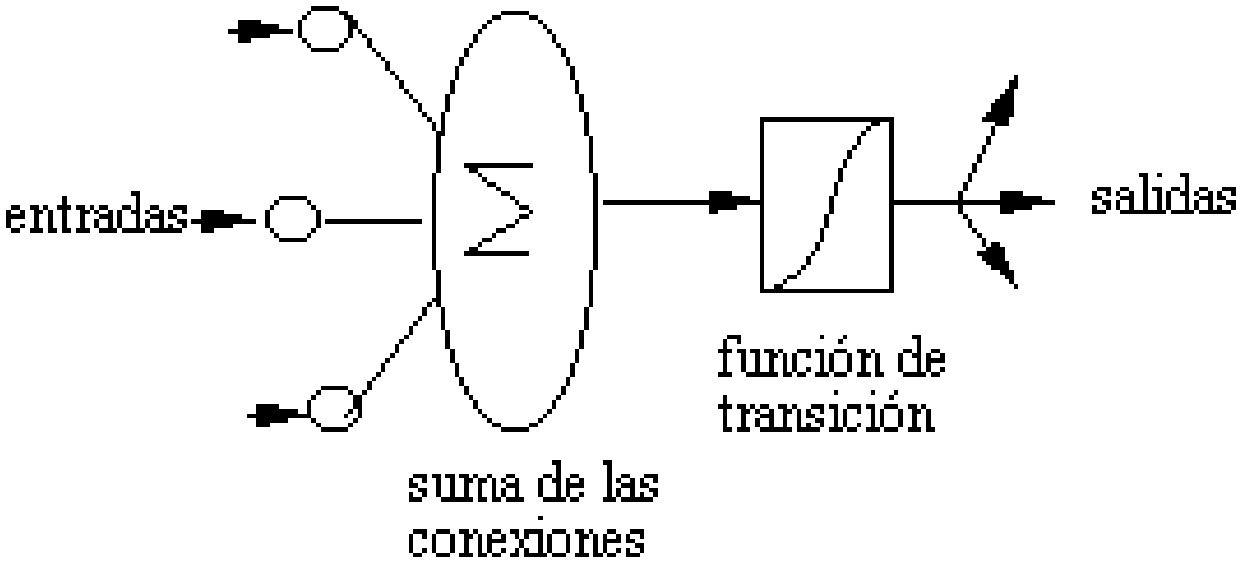
COMPARACION RED NEURONAL

Neurona Biológica	Neurona Artificial
Señales que llegan a la sinapsis	Entradas a la neurona
Carácter excitador o inhibidor de la sinapsis de entrada	Pesos de entrada
Estimulo total de la neurona	Sumatoria de pesos por entradas
Activación o no de la neurona	Función de activación
Respuesta de la neurona	Función de salida

COMPARACION RED NEURONAL

Aspectos	Computador	Cerebro Humano
Unidades de Cálculo	CPUs	10^{11} neuronas
Unidades de Almacenamiento	RAM y disco duro	10^{11} neuronas Y 10^{14} sinapsis
Ciclos	Mherz	10^{-3} segundos
Banda Ancha	Capacidad de transmisión	10^{14} conex. (bits)/segundo
Actualización/seg.	Capacidad de procesamiento paralelo	10^{14}

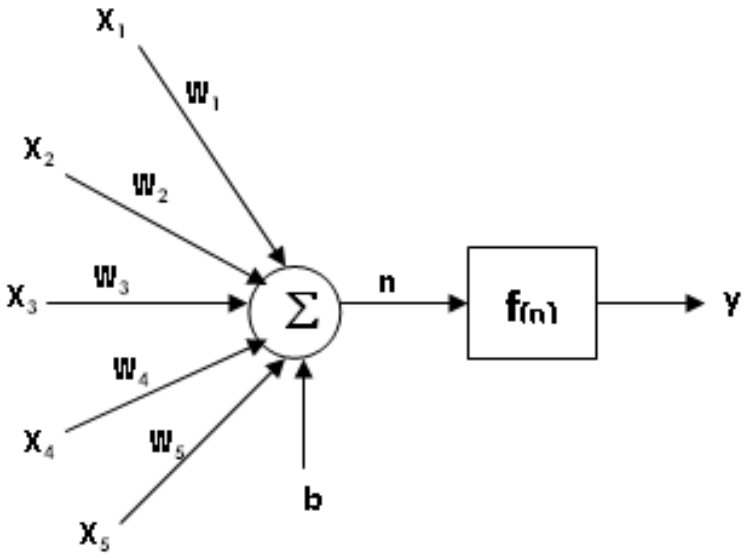
COMO TRABAJA UNA NEURONA ARTIFICIAL



COMO TRABAJA UNA NEURONA ARTIFICIAL

X1, X2,.., Xn son las **señales de entrada** y cada una pasa a través de un peso W, llamado **peso sináptico** de la conexión, cuya función es análoga a la de la **función sináptica de la neurona biológica**

El **nodo sumatorio** acumula todas las señales de entrada multiplicadas por los pesos y las pasa a la salida a través de una **función de activación** o transferencia f(n), (b es el sesgo).



- Súper-simplificación
- Analogía Metafórica
- Sorprendente poder de cómputo

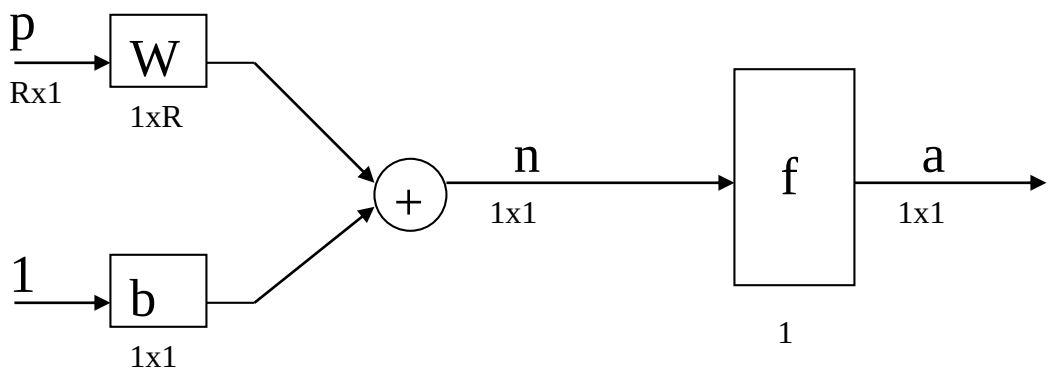
COMO TRABAJA UNA NEURONA ARTIFICIAL

- **Entradas** (x1, x2,…, xp): son escalares que se le suministran a la red para ser procesados dentro de ella, provienen del exterior o de las salidas de otras neuronas de la misma red.
- **Salidas** (y): son escalares que genera la red como resultado del problema en estudio.
- **Pesos Sinápticos** (w1, w2,…, wp): son valores que representan la influencia de las entradas sobre la neurona, cada entrada tiene su peso sináptico. Los pesos pueden ser positivos (llamados excitatorios), o negativos (denominados inhibitorios).
- **Nodo Sumatorio de las Entradas**: se encarga de la sumatoria de todas las entradas a la neurona multiplicados/ponderados por sus correspondiente peso sináptico.
- **Función de Activación o de Transferencia** (f(n)): es una función cualquiera que limita el rango de la salida de la neurona, puede ser lineal o no lineal. El tipo de esta función va a depender del problema en estudio.
- **Sesgo** (b): es el peso de una entrada fija cuyo valor es constante e igual a 1. Este valor permite que haya cierta flexibilidad en la salida de cada neurona, lo que genera un mejor ajuste de la salida obtenida con la salida deseada.

Modelo matemático de la neurona

Neurona de múltiples entradas

Típicamente una neurona tiene más de una entrada.



$$\mathbf{n = w_{11}p_1 + w_{12}p_2 + ... + b}$$

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

La salida de la neurona es:

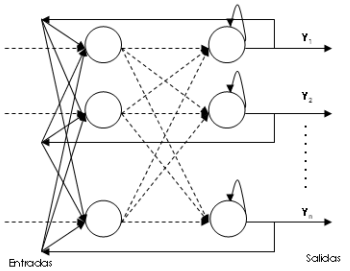
$$\mathbf{a = f(Wp+b).}$$

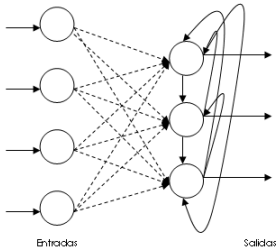
El número de entradas a la red depende del problema.

Por ejemplo, si se desea predecir las condiciones para volar las entradas podrían ser la **temperatura del aire, la velocidad del viento y la humedad** (la red necesitará tres entradas).

COMO TRABAJA UNA RED NEURONAL

1. El conjunto de **unidades de procesamiento** (neuronas formales).
2. El **estado interno o de activación** de las neuronas.
3. Las **conexiones entre las neuronas**.
4. Las **conexiones con el ambiente**.
5. **La regla de propagación**
6. **La función de activación**
7. **La función de transición o de salida**
8. La **topología o arquitectura** de la red
9 El **algoritmo de Aprendizaje**





COMO TRABAJA UNA RED NEURONAL

1. El conjunto de **unidades de procesamiento** (neuronas formales).

2. El **estado interno o de activación** de las neuronas.

3. Las **conexiones entre las neuronas**.

4. Las **conexiones con el ambiente**.

COMO TRABAJA UNA RED NEURONAL

5. **La regla de propagación** $h_i(t)= g(w_{ij}, x_j(t))$

Ej. $h_i(t)= \sum_j w_{ij}x_j(t)$

6. **La función de activación**

$$a_i(t)= f_i(a_i(t-1), h_i(t))$$

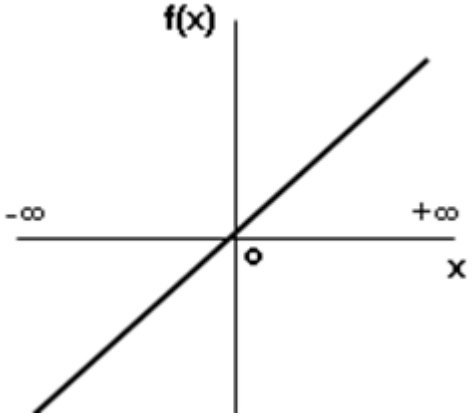
7. **La función de transición o de salida**

$$y_i(t)=F_i(a_i(t))$$

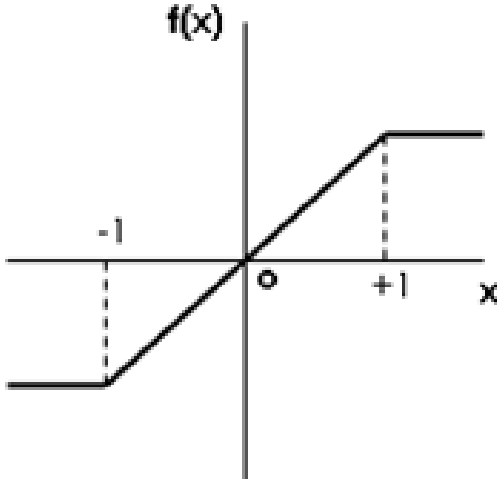
Función de activación

Función identidad o función lineal:

$$f(x) = x$$



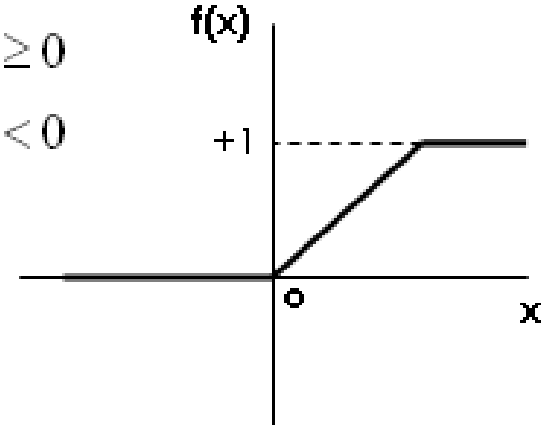
Función lineal por tramos



$$f(x) = \begin{cases} -1, x < -1 \\ x, +1 \le x \le -1 \\ +1, x > +1 \end{cases}$$

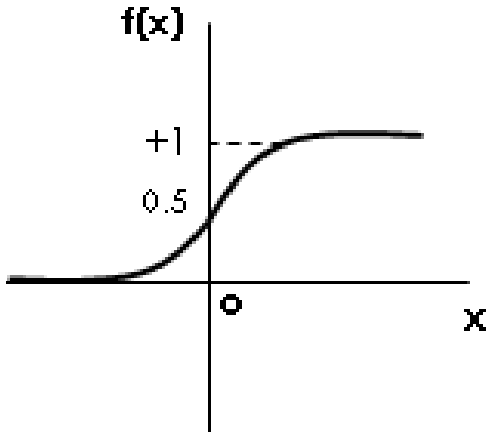
Función escalón

$$f(x) = \begin{cases} 1 & x \ge 0 \\ 0 & x < 0 \end{cases}$$



Función sigmoidal

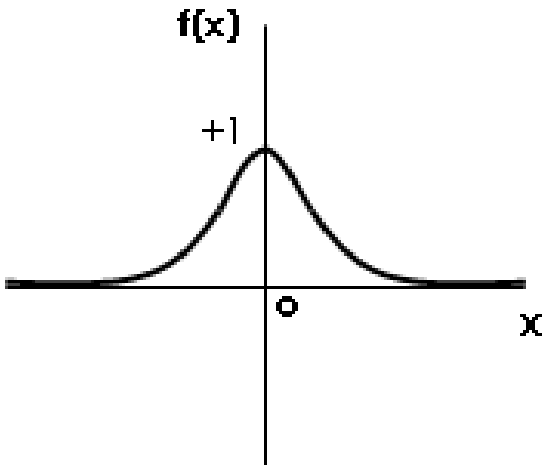
$$f(x) = \frac{1}{1+e^{-x}}$$



Función de activación

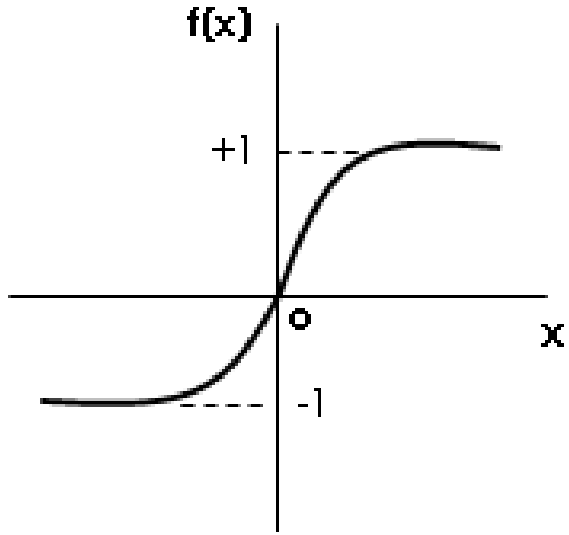
Función gaussiana:

$$f(x) = Ae^{-Bx^2}$$



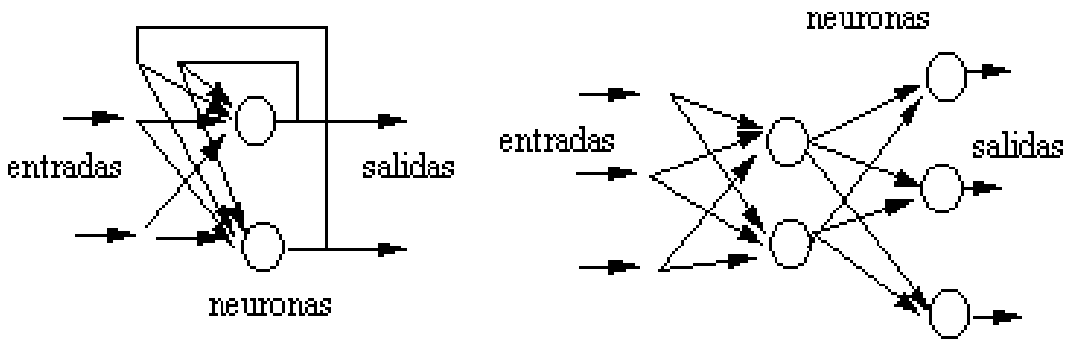
Función tangencial hiperbólica:

$$f(x) = \tanh(x)$$



COMO TRABAJA UNA RED DE NEURONAS

8. La **topología o arquitectura** de la red
   - **conexión total** (todas las neuronas interconectadas) o **conexión parcial** (por ejemplo, las redes de capas).
   - **Realimentada** o **unidireccional**



Topologías de las RNA

Redes monocapa:

- Redes con una sola capa.
- Para unirse las neuronas crean conexiones laterales para conectar con otras neuronas de la única capa.

Redes multicapas:

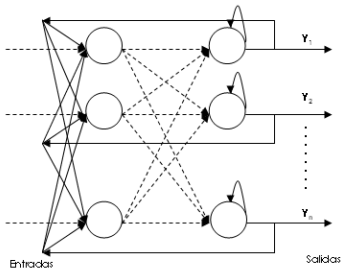
- Generalización de las anteriores donde existe un conjunto de capas intermedias entre la entrada y la salida llamadas *capas ocultas.*
- Pueden ser:

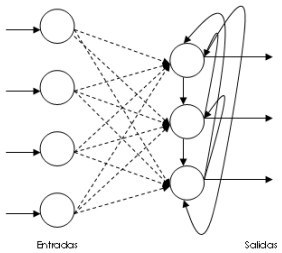
Propagación hacia adelante

Propagación hacia atrás

Redes recurrentes

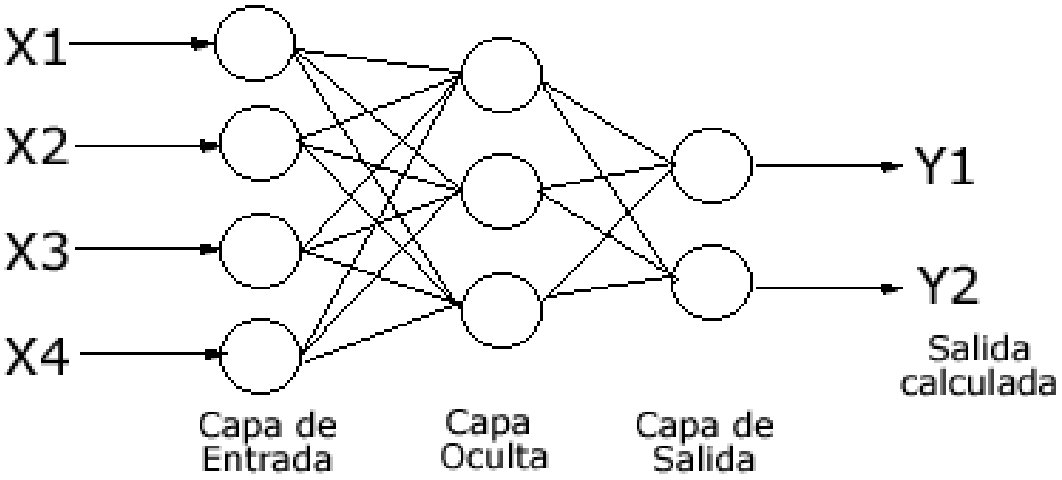
Redes de alimentación lateral

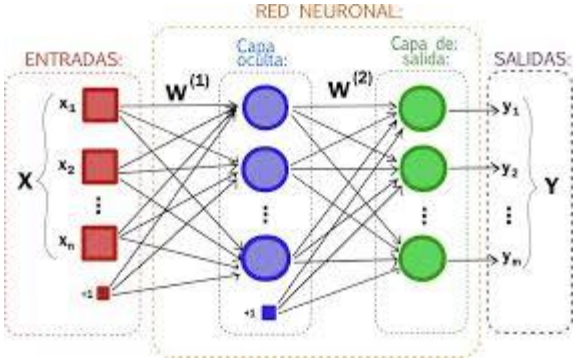




Redes Multicapas

- **Capa de Entrada:** está constituida por los nodos de entrada, que reciben directamente la información de las fuentes externas a la red.
- **Capas Ocultas:** no tienen contacto con el exterior ya que se encuentran ubicadas entre la capa de entrada y la capa de salida. La cantidad de capas ocultas dependerá del problema en estudio y deben especificarse en la arquitectura.
- **Capa de Salida:** está constituida por los nodos que transfieren la información a la salida de la red y de acuerdo al tipo de problema en estudio se determinará el número de neuronas de salida.





COMO TRABAJA UNA NEURONA

- ***Tipo de Asociación entre las Informaciones de Entrada y Salida:*** Las RNA son sistemas que almacenan cierta información aprendida; está información se registra de forma distribuida en los pesos asociados a las conexiones entre neuronas de entrada y salida.

AL APLICARLE UNA ENTRADA LA RN RESPONDE CON UNA SALIDA ASOCIADA A DICHA INFORMACIÓN DE ENTRADA

- Este mecanismo da lugar a dos tipos de RNA:
  - **Red Heteroasociativa:** es aquella que computa cierta función, entre un conjunto de entradas y un conjunto de salidas, correspondiendo a cada posible entrada una determinada salida.
  - **Red Autoasociativa:** es una red cuya misión es reconstruir una determinada información de entrada que se presenta incompleta o distorsionada.

MEMORIAS ASOCIATIVAS

AUTOASOCIATIVA

1. RED APRENDE Ai
2. SI SE PRESENTA Ai, RN RESPONDE Ai
3. ASOCIA INFORMACION DE ENTRADA CON LO MAS PARECIDO QUE TENGA GUARDAD
4. PUEDE SER DE 1 CAPA
5. APRENDIZAJE NO SUPERVISADO => 0 CONEXIONES RECURRENTES Y/O LATERALES

MEMORIAS ASOCIATIVAS

HETEROASOCIATIVAS:

1. RED APRENDE PAR DE DATOS (Ai, Bi)
2. SI SE PRESENTA Ai, RN RESPONDE Bi
3. POR LO MENOS DOS CAPAS
4. TODOS LOS TIPOS DE APRENDIZAJE

COMO TRABAJA UNA NEURONA: APRENDIZAJE

- ***REDES NEURALES:***
  - ***CAPACIDAD PARA APRENDER***
  - ***OPTIMIZA SU RENDIMIENTO ASI***

APRENDIZAJE: *MODIFICAR PESOS DE LAS CONEXIONES DE LAS NEURONAS (CREAR, DESTRUIR, MODIFICAR)*

$$Wij = Wij + \Delta Wij$$

Modelos Neuronales

Realimentados :

feed-propagation

ART,

HOPFIELD

Unidireccionales

PERCEPTRON,

M RN,

BOLTZMAN,

backpropagation

KOHONEN

Híbridos:

RBF (RADIAL BASIC FUNCTION)

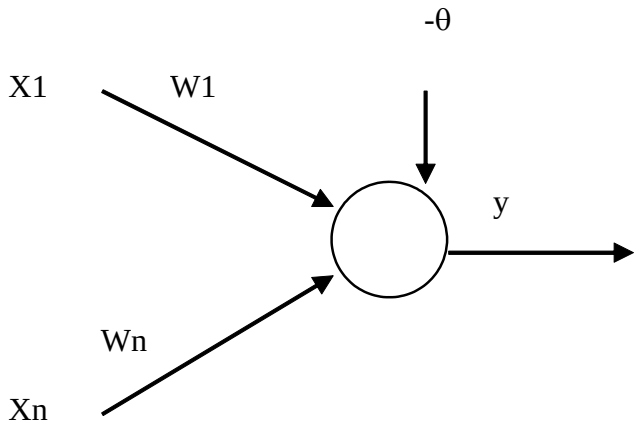
Redes basadas en DEEP LEARNING

Redes de Convolución

Extreme Learning

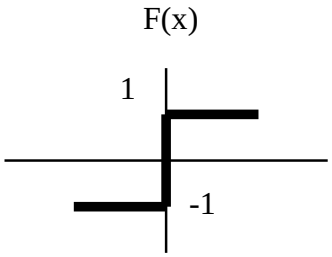
PERCEPTRÓN

- 1ER MODELO DE RED DE NEURONAS ARTIFICIALES (ROSEMBLATT 1958)
- APRENDE PATRONES SENCILLOS (2 CLASES)
- DOS ENTRADAS CON 1 NEURONA



✓ Es un modelo unidireccional compuesto por dos capas de neuronas, una de entrada y otra de salida, con 1 neurona.

✓ Se aplica en el calculo del AND, OR, XOR, pero no resuelve el OR-exclusivo



$$Y=F(\sum WiXi-\theta)$$

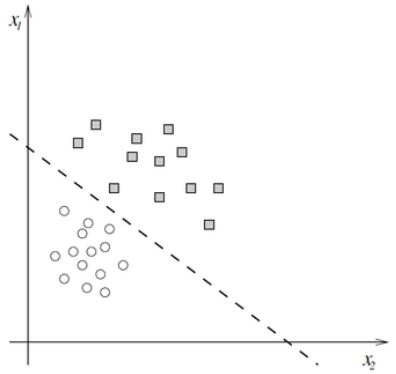
PERCEPTRÓN

- REGIONES QUE INDICA A QUE PATRÓN PERTENECE CADA CLASE SEPARADAS POR UN HIPERPLANO

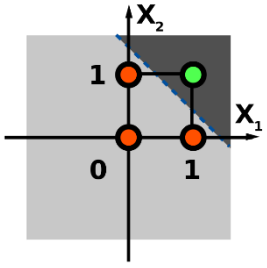
  => PATRONES SEPARABLES GEOMÉTRICAMENTE

  => DOS ENTRADAS LINEA RECTA $X2=W1X1/W2+\theta/W2$

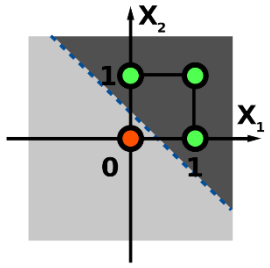
  => TRES ENTRADAS PLANO



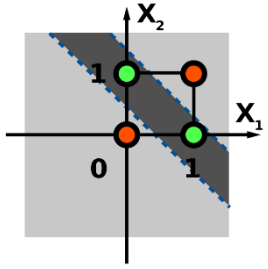
- LAS FUNCIONES **AND** Y **OR** SON LINEALMENTE SEPARABLES, POR LO TANTO LAS PUEDE APRENDER. EL **XOR** NO LO PUEDE APRENDER



AND ($X_1 \cap X_2$)



OR ($X_1 \cup X_2$)



XOR

PERCEPTRÓN

- **APRENDIZAJE:** SUPERVISADO

- **ALGORÍTMO:**

1. INICIAR PESO Y UMBRAL

2. PRESENTAR PAR ENTRADA-SALIDA

3. CALCULAR SALIDA ACTUAL
   Y(t)

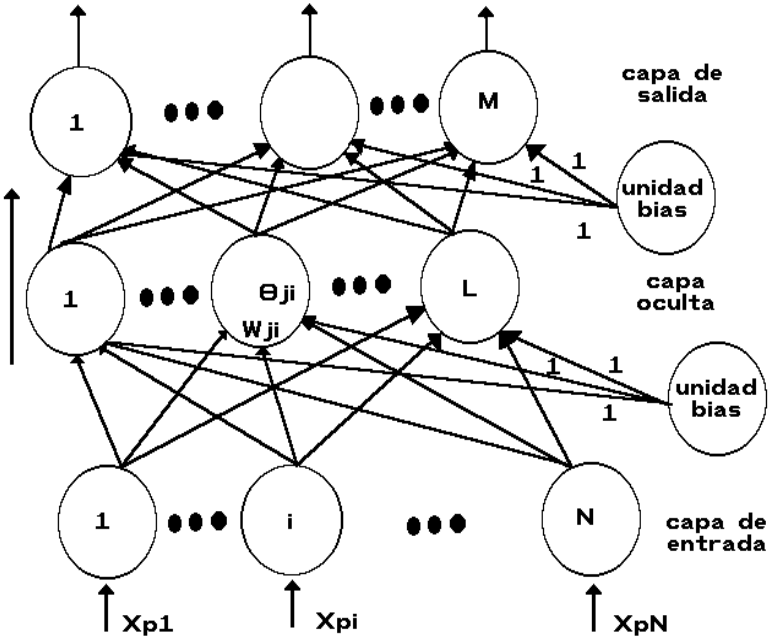
4. ADAPTAR LOS PESOS
   $Wi(t)=Wi(t)+\alpha[d(t)-Y(t)]Xi(t)$

HASTA QUE **$d(t)-y(t)^2$** valor pequeño

5. REGRESAR AL PASO 2

Backpropagation

- ✓ Fue propuesto por Rumelhat, Hinton y Williams, 1986.
- ✓ Aprendizaje pares de Entrada- Salida.
- ✓ Capacidad de autoadaptación pesos capas escondidas
- ✓ Al aplicar un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida.
- ✓ Se aplica en la codificación de información, traducción de texto en lenguaje hablado, clasificación de señales electrocardiográficas y en la comprensión y descompresión de los datos.



Modelo de Backpropagation

CAPACIDAD DE AUTOADAPTACIÓN PESOS CAPAS ESCONDIDAS => GENERALIZACIÓN

BACKPROPAGATION

- **ALGORÍTMO:**
  1. **INICIAR PESOS**
  2. **PRESENTAR PAR ENTRADA-SALIDA**
  3. **CALCULAR SALIDA RED**

CAPA OCULTA:

$$net_{pj}^{k} = \sum_{i=1}^{N} w_{ji}^{k} x_{pi} + \Theta_{j}^{k} \qquad ; y_{pj} = f_{j}^{k}\left(net_{pj}^{k}\right)$$

n: capa n-esima

p: patrón p

j: neurona j de la capa n

BACKPROPAGATION (cont)

CAPA SALIDA:

$$net_{pk}^{o} = \sum_{i=1}^{I} w_{kj}^{o} y_{pj} + \Theta_{k}^{o}$$

$$y_{pk} = f_{k}^{o}\left(net_{pk}^{o}\right)$$

o: capa de salida

4. CALCULAR ERROR PARA TODAS LAS NEURONAS

CAPA DE SALIDA:

$$\delta_{pk}^{o} = \left(d_{pk} - y_{pk}\right) f_{k}^{o}\left(net_{pk}^{o}\right)$$

$$\delta_{pk}^{o} = \left(d_{pk} - y_{pk}\right) \qquad f_{k}^{o} = 1$$

$$\delta_{pk}^{o} = \left(d_{pk} - y_{pk}\right) * y_{pk}\left(1 - y_{pk}\right) \qquad f_{k}^{o'} = f_{k}^{o}\left(1 - f_{k}^{o}\right) = y_{pk}\left(1 - y_{pk}\right)$$

f_ko' derivable lineal sigmoidal

BACKPROPAGATION (cont)

CAPA OCULTA

$$\delta_{pj}^{k} = f_j^{k}\left(net_{pj}^{k}\right)\sum_{k}\delta_{pk}^{o} w_{kj}^{o}$$

$$\delta_{pj}^{k} = x_{pj}\left(1 - x_{pj}\right)\sum_{k}\delta_{pk}^{o} w_{kj}^{o}$$

5. ACTUALIZAR PESOS

CAPA SALIDA

$$w_{kj}^{o}\left(t+1\right) = w_{kj}^{o}\left(t\right) + \Delta w_{kj}^{o}\left(t+1\right) \quad ;$$

$$\Delta w_{kj}^{o}\left(t+1\right) = \alpha\delta_{pk}^{o} y_{pj}$$

CAPA OCULTA

$$w_{ji}^{k}\left(t+1\right) = w_{ji}^{k}\left(t\right) + \Delta w_{ji}^{k}\left(t+1\right)$$

$$\Delta w_{ji}^{k}\left(t+1\right) = \alpha\delta_{pj}^{k} x_{pi}$$

BACKPROPAGATION

6. REPETIR PROCESO HASTA QUE

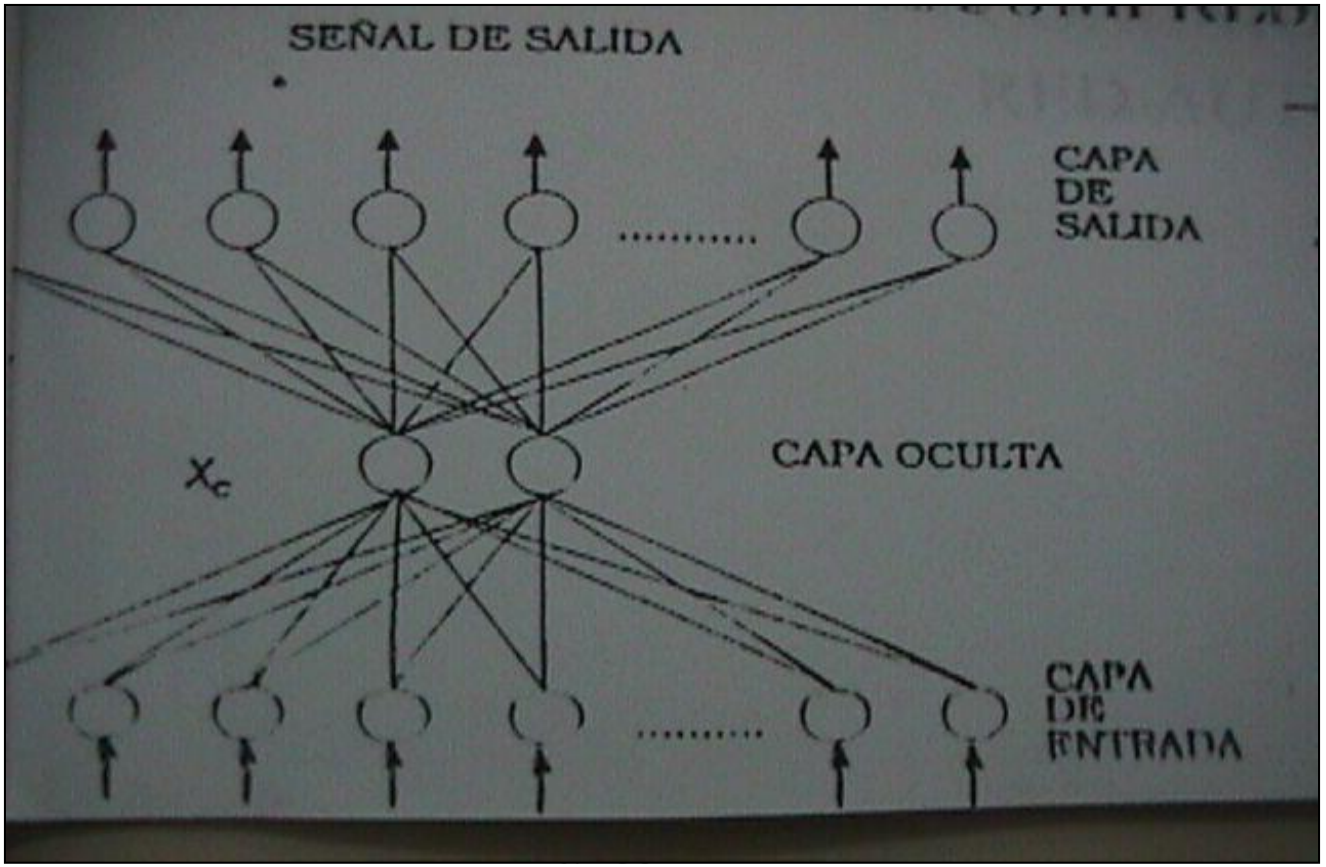
$$E_p = \frac{1}{2} \sum_{k=1}^{M} \delta_{pk}^2$$

M: número neuronas salidas

PARA CADA PATRÓN

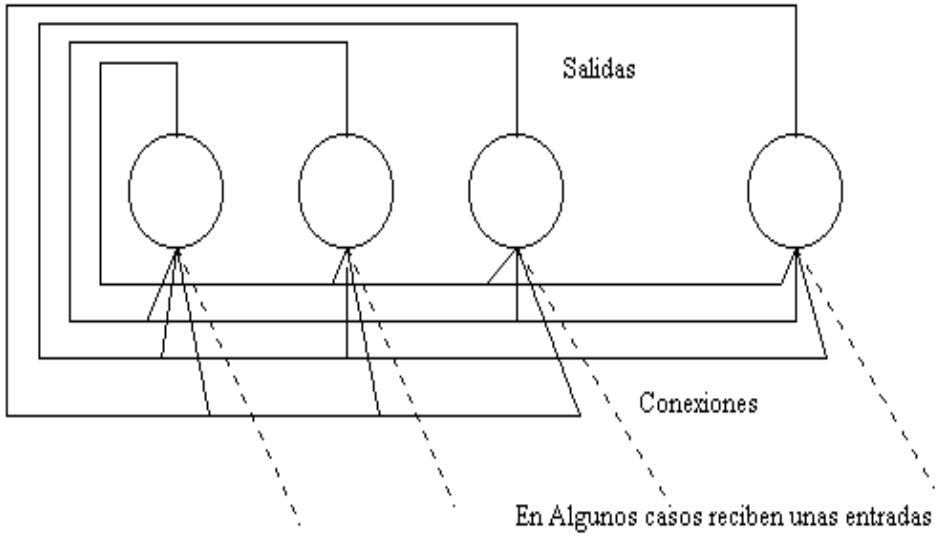
BACKPROPAGATION (aplicación)

- **COMPRESIÓN Y DESCOMPRESIÓN DE DATOS**



Modelo de Hopfield

- John Hofield, desarrolló este modelo en 1982.
- Red Autoasociativa.
- Red monocapa con N neuronas
- Salidas binarias 0/1 ó -1/+1.
- Conexiones simétricas entre pares.
- Aprendizaje no supervisado de tipo hebbiano.
- Se aplica en el reconocimiento de imágenes y de voz, en control de motores y, sobre todo en la resolución de problemas de optimización.



MODELO DE HOPFIELD

GUARDAR INFORMACIÓN EN UNA CONFIGURACIÓN DINÁMICA ESTABLE

ESTABLECE PARALELISMO ENTRE SU MODELO Y SISTEMAS ESTUDIADOS EN FÍSICA ESTADÍSTICA

MODELO DE HOPFIELD

ARQUITECTURA

- RED MONOCAPA CON N NEURONAS
- SALIDAS BINARIAS [0, 1] O [-1, 1]
- FUNCIÓN ACTIVACIÓN:
  - ESCALÓN (DISCRETA)
  - SIGMOIDAL (CONTINUA)
- TODAS LAS NEURONAS CONECTADAS
- CONEXIONES SIMÉTRICAS ENTRE PARES
- RED AUTOASOCIATIVA

MODELO DE HOPFIELD

FUNCIONAMIENTO

1.- EN EL INSTANTE INICIAL SE APLICA INFORMACIÓN DE ENTRADA (e1, ..., eN)

$$S_i(t=0)=e_i \; ; \; 1<=i<=N$$

2.- RED ITERA HASTA CONVERGER => $s_i(t+1)=s_i(t)$

f : FUNC $s_i(t+1) = f\left(\sum_{j=1}^{N} w_{ij} s_j(t) - \Theta_i\right) \quad ;1 \le i \le N$

MODELO DE HOPFIELD

FUNCIONAMIENTO

3.- SALIDA (si) DESPUES CONVERGENCIA

=> INFORMACIÓN ALMACENADA MAS PARECIDA A LA ENTRADA

- **- APRENDIZAJE:**
  - **FUERA DE LINEA**
  - **NO SUPERVISADO HEBBIANO**

MODELO DE HOPFIELD

APRENDIZAJE

[-1, 1]

$$w_{ij} = \begin{cases} \sum_{k=1}^{M} e_i^{(k)} e_j^{(k)} & ;1 \le i,j \le N; i \ne j \\ 0 & ;1 \le i,j \le N; i = j \end{cases}$$

[0, 1]

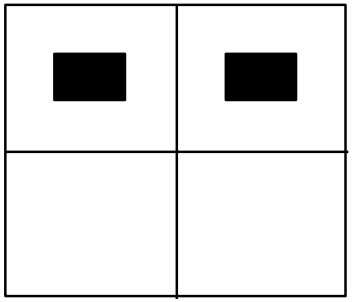
$$= \begin{cases} \sum_{k=1}^{M} (2e_i^{(k)} - 1)(2e_j^{(k)} - 1) & ;1 \le i,j \le N; i \ne j \\ 0 & ;1 \le i,j \le N; i = j \end{cases}$$

MODELO DE HOPFIELD

ENTRENAMIENTO

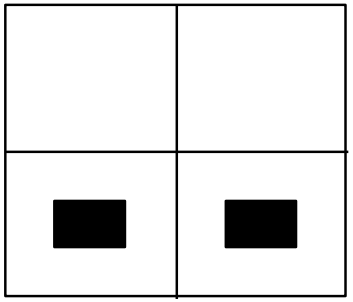
- FIGURA A, B, ...

Figura A



E1={1,1,-1,-1}

Figura B



E2={-1,-1,1,1}

MODELO DE HOPFIELD

ENTRENAMIENTO

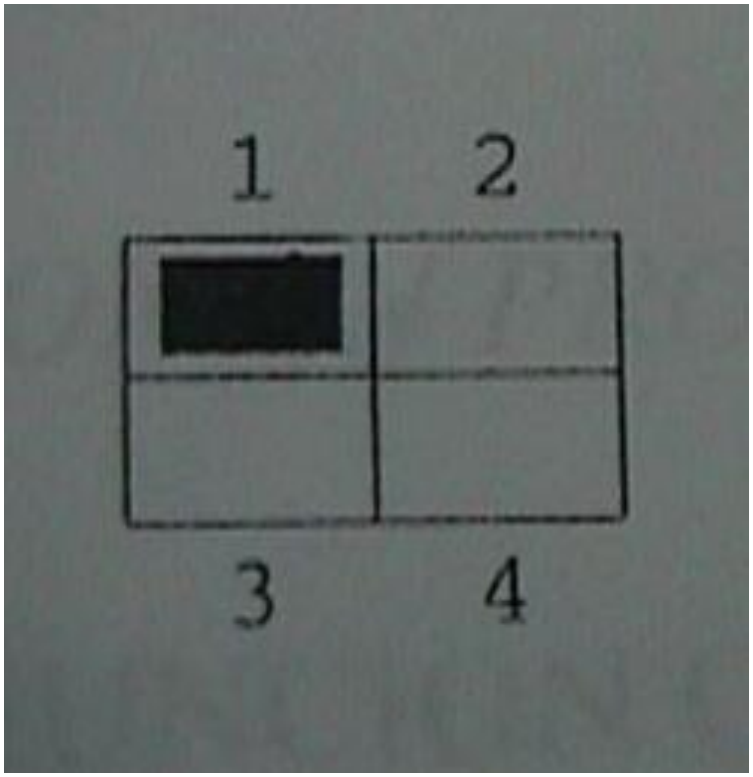
$$E_1 = [1,1,-1,-1] \qquad E_2 = [-1,-1,1,1]$$

$$W = \sum_{k=1}^{M} \left[E_k^T E_k - I \right]$$

$$E_1^T E_1 - I = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} [1\,1\,-1\,-1] - \begin{bmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \end{bmatrix} = \begin{bmatrix} 01-1-1 \\ 10-1-1 \\ -1-101 \\ -1-110 \end{bmatrix}$$

MODELO DE HOPFIELD

FUNCIONAMIENTO



E = [1 -1 -1 -1]

MODELO DE HOPFIELD

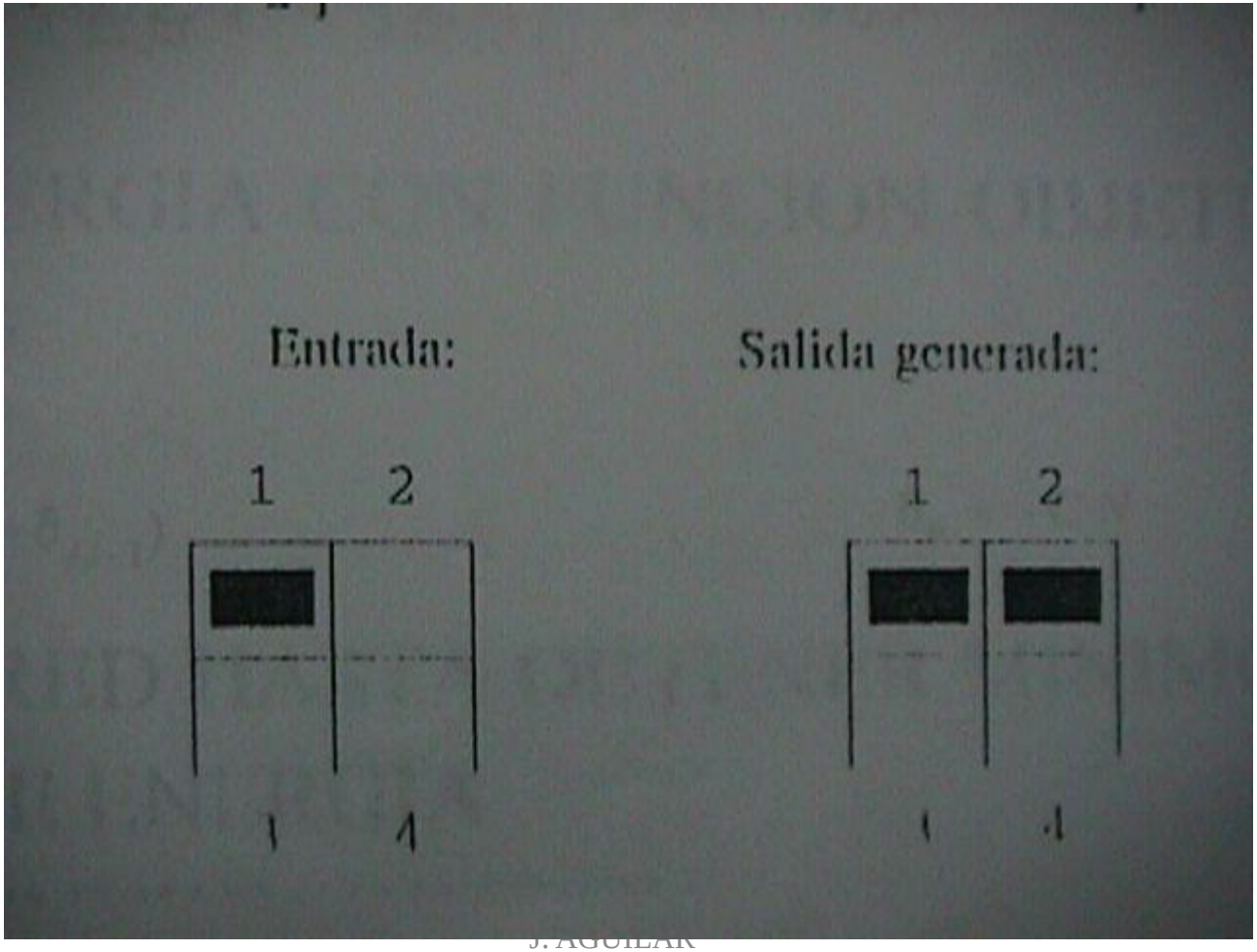
FUNCIONAMIENTO

$$E \cdot W = [1 \ \ -1 \ \ -1 \ \ -1] \begin{bmatrix} 0 & 2 & -2 & -2 \\ 2 & 0 & -2 & -2 \\ -2 & -2 & 0 & 2 \\ -2 & -2 & 2 & 0 \end{bmatrix} = [2 \ \ 6 \ \ -2 \ \ -2]$$

$$S = [1 \ \ 1 \ \ -1 \ \ -1]$$

MODELO DE HOPFIELD

FUNCIONAMIENTO



MODELO DE HOPFIELD

FUNCIÓN DE ENERGÍA

- ESPACIO CONFORMADO POR POSIBLESCONFIGURACIONES DE SALIDAS DE LAS NEURONAS
- ESTADO DE LA RED EN CADA MOMENTO ES UN PUNTO EN ESE ESPACIO
- DISCRETA:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ (\neq i)}}^{N} w_{ij} s_i s_j + \sum_{i=1}^{N} \Theta_i s_i$$

MODELO DE HOPFIELD
FUNCIÓN DE ENERGÍA

- CONTINUA:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ i\neq j}}^{N} w_{ij}s_i s_j + \sum_{i=1}^{N}\Theta_i s_i - \frac{1}{\alpha}\sum_{i=1}^{N}\int_0^{s_i} Ln\left(\frac{1}{s}-1\right)ds$$

MODELO DE HOPFIELD

APLICACIÓN EN PROBLEMAS DE OPTIMIZACIÓN

- FIJAR FUNCIÓN OBJETIVO DEL PROBLEMA

$$F = \frac{A}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{l=1\\ l\neq j}}^{N} s_{ij}s_{il} + \frac{B}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\ k\neq i}}^{N} s_{ij}s_{kj} + \frac{C}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij} - N\right)^2 + \frac{D}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\ k\neq i}}^{N} d_{ij}\left(s_{ij}s_{kj+1} + s_{ij}s_{kj-1}\right)$$

- COMPARAR FUNCIÓN ENERGÍA CON FUNCIÓN OBJETIVO Y DETERMINAR PESOS Y UMBRALES

$$w_{ij,kl} = -A\,\delta_{ik}\left(1-\delta_{jl}\right) - B\,\delta_{jl}\left(1-\delta_{ik}\right) - C - Dd_{ik}\left(\delta_{jl+1} + \delta_{jl-1}\right) \quad ; \Theta_{ij} = -C.N$$

MODELO DE HOPFIELD

APLICACIÓN EN PROBLEMAS DE OPTIMIZACIÓN

- PONER A FUNCIONAR LA RED HASTA OBTENER MÍNIMO VALOR DE LA FUNCIÓN DE ENERGÍA

=> MÍNIMO VALOR FUNCIÓN OBJETIVO

Random Neural Model

- INTRODUCIDO POR *Gelenbe* EN 1989.
- BASADO EN LA DISTRIBUCION DE PROBABILIDADES DE LAS NEURONAS
- CONSISTE DE N NEURONAS ENTRE LAS CUALES CIRCULAN SENALES POSITIVAS Y NEGATIVAS
- NEURONAS ACUMULAN SENALES Y PUEDEN EMITIR SI SU CONTADOR DE SENALES ES POSITIVO EN UN MOMENTO DADO

Random Neural Model

- UNA SENAL POSITIVA REDUCE EN 1 POTENCIAL DE LA NEURONA, O NO TIENE EFECTO SI EL MISMO ES 0

- UNA SENAL NEGATIVA AUMENTA EN 1 POTENCIAL DE LA NEURONA,

- CADA NEURONA i ES REPRESENTADA POR SU POTENCIAL AL INSTANTE t $k_i(t)$.

Random Neural Model

- CADA VEZ QUE UNA NEURONA i EMITE, SE RESETEA POTENCIAL, Y SALE UNA SENAL DE EL COMO SENAL POSITIVA SEGÚN PROBABILIDAD $p^+(i,j)$ O COMO SENAL NEGATIVA CON PROBABILIDAD $p^-(i,j)$ O HACIA FUERA CON PROBABILIDAD $d(i)$.

$$\Sigma^n_{j=1} [p^+(i,j)+p^-(i,j)] + d(i) = 1$$

Random Neural Model

- SENALES POSITIVAS LLEGAN A NEURONA i DEL EXTERIOR CON PROBABILIDAD $\Lambda(i)$ Y SENALES NEGATIVAS CON PROBABILIDAD $\lambda(i)$
- TASA DE EMISION DE LA NEURONA i ES r(i).
- PROBABILIDAD DE EXCITACIÓN DE LA NEURONA *i*, q(i), ES

$$q(i) = \lambda^{+}(i)/(r(i)+\lambda^{-}(i))$$

$$\lambda^{+}(i) = \sum_{j=1}^{n} q(j)r(j)p^{+}(j,i)+\Lambda(i)$$

$$\lambda^{-}(i) = \sum_{j=1}^{n} q(j)r(j)p^{-}(j,i)+\lambda(i)$$

Random Neural Model

PESOS POSITIVOS ($w^+(i,j)$) Y NEGATIVOS ($w^-(i,j)$)

$$w^+(i,j) = r(i)p^+(i,j)$$

$$w^-(i,j) = r(i)p^-(i,j)$$

Y

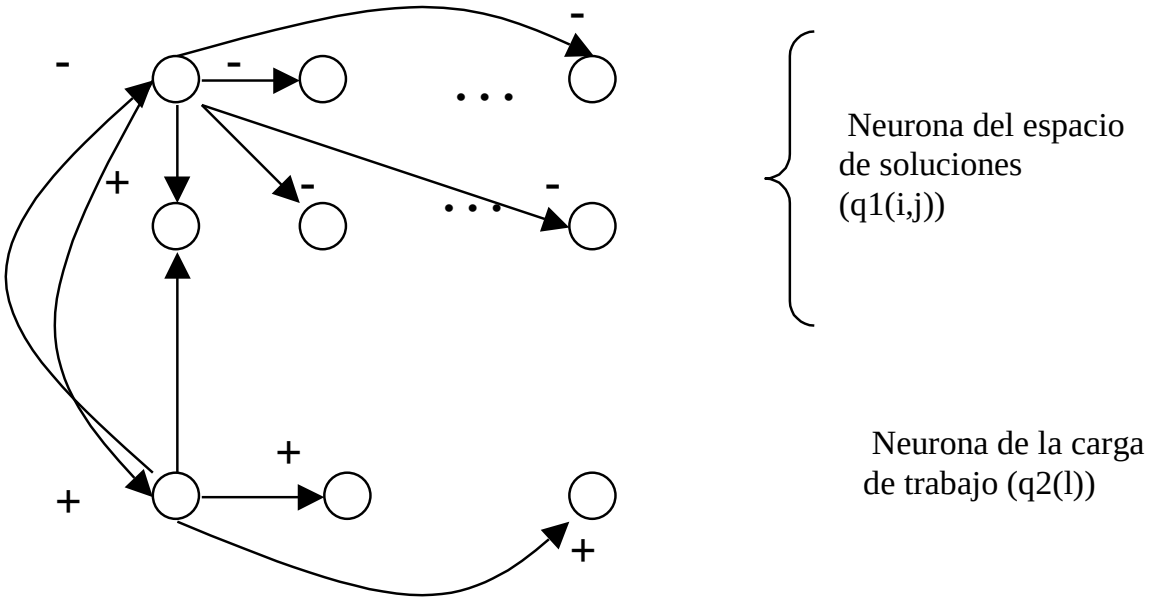
$$r(i) = \Sigma^n_{j=1} [w^+(i,j) + w^-(i,j)]$$

Random Neural Model

APLICACIONES

Partición de Grafos

$$FO = \Sigma_i\Sigma_j\Sigma_k\Sigma_{l\neq j}C_{ik}q1(i,j)q1(k,l) + \Sigma_i\Sigma_j(q1(i,j)-n/K)\ q2(i)$$



Neurona del espacio de soluciones (q1(i,j))

Neurona de la carga de trabajo (q2(l))

Multiple Classes Random Network Model

- Varias clases

$$\underline{K_i}=(K_{i1}, ..., K_{iC}),$$

K_{ic} "nivel de excitacion de la senal clase *c*",

- potencial total de neuron *i* es $K_i = \Sigma_{c=1}^{C} K_{ic}$.
- Senal negativa reduce por 1 al potencial de una clase aleatoriamente con probabilidad Kic/Ki
-

Multiple Classes Random Network Model

- Claramente:

$$\Sigma_{(j,\,\varphi)}\, p^{+}(i, c; j, \varphi) + \Sigma_{j}\, p^{-}(i, c; j) + d(i, c) = 1$$

- Probabilidad excitación de la "clase φ" de la neurona *j*

$$q(j, \varphi) = \lambda^{+}(j, \varphi)/(r(j, \varphi) + \lambda^{-}(j))$$

$$\lambda^{+}(j, \varphi) = \Sigma_{(i\,,\,c)}\, q(i, c) r(i, c) p^{+}(i, c; j, \varphi) + \Lambda(j, \varphi)$$

$$\lambda^{-}(j) = \Sigma_{(i\,,\,c)}\, q(i, c) r(i, c) p^{-}(i, c; j) + \lambda(j)$$

Multiple Classes Random Network Model

- Pesos:

$$w^{+}(i, c; j, \varphi) = r(i, c)p^{+}(i, c; j, \varphi)$$

$$w^{-}(i, c; j) = r(i, c)p^{-}(i, c; j)$$

- y, si d(i, c)=0, r(i, c) es

$$r(i, c) = [\Sigma_{(j, \varphi)} w^{+}(i, c; j, \varphi) + \Sigma_{(j)} w^{-}(i, c; j)]$$

Aprendizaje

en las RNs

APRENDIZAJE

- El aprendizaje de una RNA se basa en un proceso que **permite que la red aprenda a comportarse según unos objetivos específicos**.

- El aprendizaje le da la capacidad a la RNA de **cambiar su comportamiento**, es decir su proceso de entrada-salida, como resultado de los cambios en el medio.

- En particular, las reglas de aprendizaje son procedimientos que se siguen para **ajustar los parámetros de la red** a partir de un proceso de estimulación por el entorno de la red

- La mayoría de las veces consiste en **determinar un conjunto de pesos**

- El aprendizaje es esencial para la mayoría de las arquitecturas de RNA, por lo que la **elección de un algoritmo de aprendizaje** es algo de gran importancia en el diseño de una red.

Componente de aprendizaje

- Cuales componentes del componente de ejecución deben ser aprendidos
- Cual es la representación usada para esos elementos
- Cual es el mecanismo de retro-alimentación disponible
  - Aprendizaje Supervisado
  - Aprendizaje No supervisado => agrupamiento
  - Aprendizaje Reforzado
- Que conocimiento-a-priori esta disponible

Redes Neuronales Artificiales

RNA:

Habilidad de aprender del medio ambiente y mejorar su desempeño por medio del aprendizaje.

Aprendizaje:

Proceso por el cual los parámetros (pesos sinápticos) de una RNA se adaptan por estimulación del medio ambiente.

Algoritmo de Aprendizaje:

Conjunto bien definido de reglas para actualizar los pesos sinápticos.

Paradigma de Aprendizaje:

Modelo del medio ambiente en el cual la RNA opera.

APRENDIZAJE

- Al finalizar la fase de entrenamiento/aprendizaje de una RNA, se espera que la red haya aprendido lo suficiente para **resolver otro problema similar** satisfactoriamente.

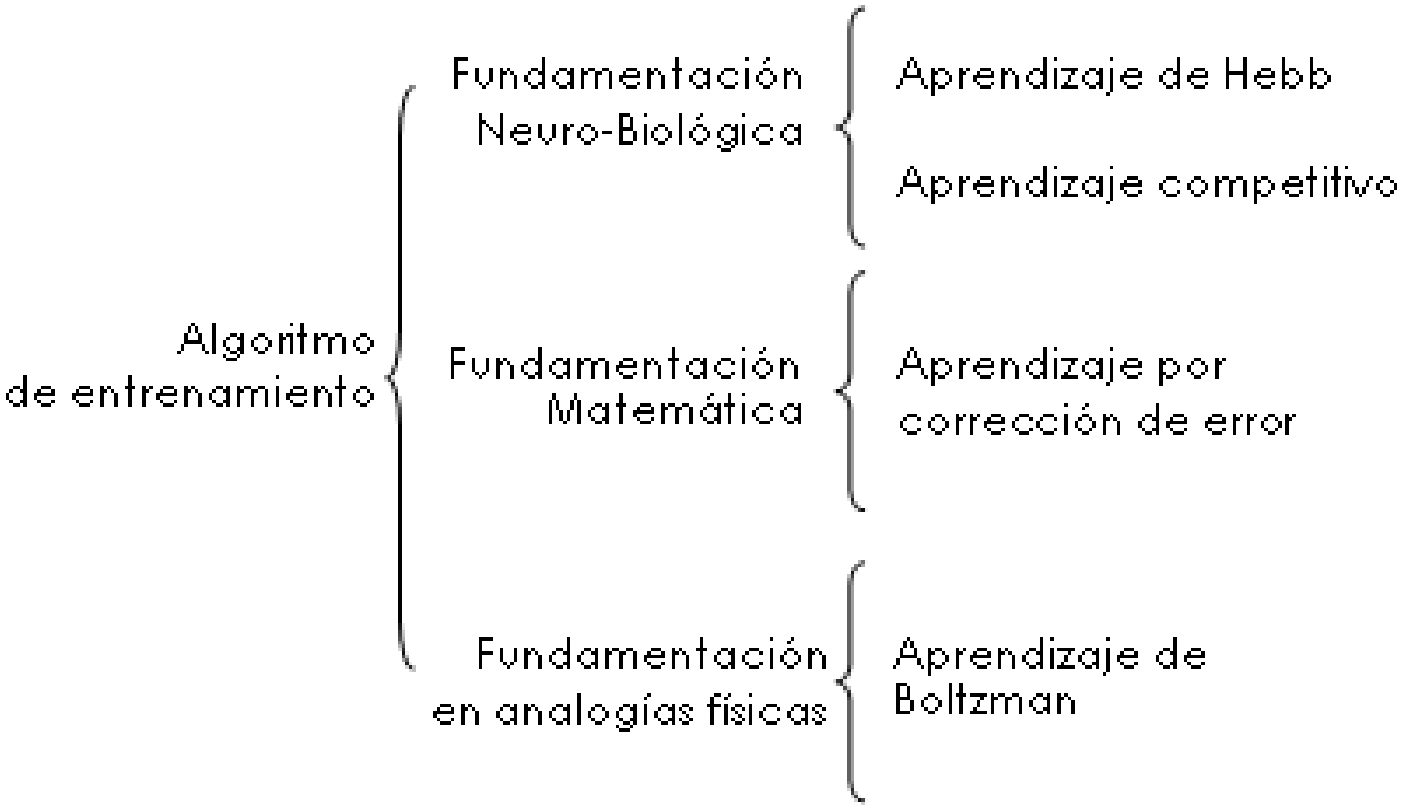
MODIFICAR PESOS DE LAS CONEXIONES DE LAS NEURONAS (CREAR, DESTRUIR, MODIFICAR)

$$w_{ij}(t+1)=w_{ij}(t) + \Delta w_{ij}(t)$$

Algoritmo clásico de un RNA

1. Presentación de las entradas
2. Calculo de la salida actual
3. Adaptación de los pesos

APRENDIZAJE



Clasificación de los Algoritmos de Aprendizaje basados en su fundamentación conceptual

APRENDIZAJE

A. ***PARADIGMAS DE APRENDIZAJE:*** *Define como se relaciona con su entorno.* Se distinguen por el tipo de retroalimentación que se le ofrece al alumno.

- **supervisado:** el crítico proporciona la salida correcta.
- **no supervisado,** no se proporciona retroalimentación en absoluto.
- **Basado en recompensa**: la crítica proporciona una evaluación de la calidad (el "premio") de lo hecho por el alumno.

APRENDIZAJE

B. ALGORÍTMOS DE APRENDIZAJE: *DEFINEN REGLAS DE APRENDIZAJE (MODIFICACIÓN DE LOS PESOS)*

CORRECCIÓN DE ERROR
BOLTZMAN
HEBBIANO
COMPETITIVO
EVOLUTIVO

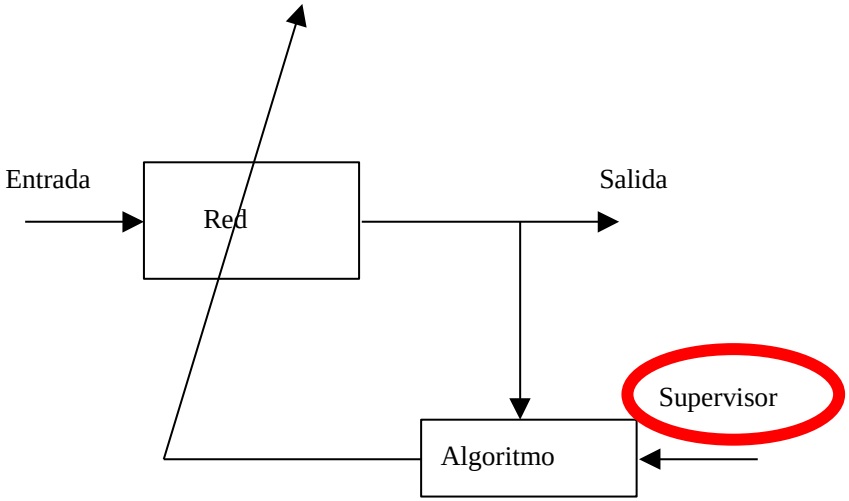
SUPERVISADO

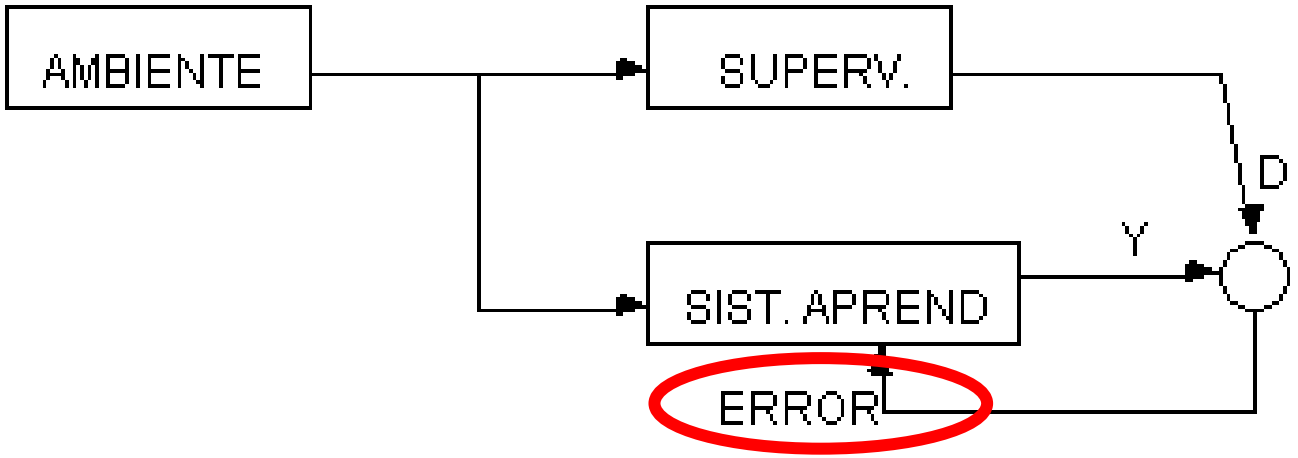
Respuesta correcta para cada ejemplo es dada

- SE DAN DATOS DE ENTRADA Y SALIDA OBJETIVO

- SALIDA RED DEBE CONCORDAR CON LA DESEADA

SUPERVISADO





SUPERVISADO

A. CONTROLADO POR SUPERVISOR

B. SALIDA RED DEBE CONCORDAR CON LA DESEADA

C. SE DAN DATOS DE ENTRADA Y SALIDA

CORRECCIÓN DE ERROR

CONOCIDO TAMBIEN COMO **DESCENSO DE GRADIENTE**

$$\mathbf{E_k(t) = D_k(t) - Y_k(t)}$$

D_k: respuesta deseada

Y_k: respuesta neurona k $\qquad Y_k=F(X_k)$

X_k: entrada neurona k

$$\mathbf{\Delta W_{ij}(t)= \alpha E_i(t)\ X_j(t)}$$

α: taza de aprendizaje

CORRECCIÓN DE ERROR

ALGORITMO

1. CALCULAR EDO. DE LA RED (Y_i)
2. CALCULAR ERROR (E_i)
3. AJUSTAR PESOS

$$w_{ij}(t+1)=w_{ij}(t) + \Delta w_{ij}(t)$$

REGLA DE WIDROW-HOLF

$$\text{ERROR GLOBAL} = \frac{1}{2P} \sum_{K=1}^{P} \sum_{J=1}^{N} \left(Y_J^K - d_J^K \right)$$

N: Número de Neuronas

P: Número de Información a Aprender

$$\Delta Wji = \alpha \, \partial \, \text{ERROR}_{\text{GLOBAL}} / \partial Wji$$

Algoritmo de aprendizaje fuera de linea de una RNA

1. Inicialización de los pesos y umbral

2. Fase de **Entrenamiento**
    1. Presentación de las entradas y salida deseada
    2. Adaptación de los pesos

3. Fase de **Reconocimiento**
    1. Presentación de una entrada dada
    2. Salida reconocida

Multiple Classes Random Network Model

Aprendizaje

- descenso de gradiente
- *m* pares de entrada-salida (X, Y):

$X = \{X_1, ..., X_m\}$ $X_k = \{X_k(1,1), ..., X_k(n, C)\}$,

y $X_k(i, c) = \{\Lambda_k(i, c), \lambda_k(i)\}$

$Y = \{Y_1, ..., Y_m\}$ $Y_k = \{Y_k(1,1), ..., Y_k(n, C)\}$,

y $Y_k(1,1) = \{0, 0.5, 1\}$

Multiple Classes Random Network Model

Aprendizaje

- Error a minimizar es E_k:

$$E_k = 1/2 \, \Sigma^{n}_{i=1} \, \Sigma^{C}_{c=1} \, [q_k(i,c) - Y_k(i,c)]^2$$

- Regla de actualizacion de pesos

$$w_k^{+}(u,p;\, v,z) = w_{k-1}^{+}(u,p;\, v,c) - \mu \Sigma^{n}_{i=1} \, \Sigma^{C}_{c=1} \, (q_k(i,c) - y_k(i,c))[\, \delta q(i,c) / \delta w^{+}(u,p;\, v,z)]_k$$

$$w_k^{-}(u,p;\, v) = w_{k-1}^{-}(u,p;\, v) - \mu \Sigma^{n}_{i=1} \, \Sigma^{C}_{c=1} \, (q_k(i,c) - y_k(i,c))[\, \delta q(i,c) / \delta w^{-}(u,p;\, v)]_k$$

Multiple Classes Random Network Model

Algoritmo de Aprendizaje

- *Initiate the matrices W_0^+ and W_0^- in some appropriate manner.*
- *For each successive value of m:*
  - *Set the input-output pair (X_k, Y_k)*
  - *Repeat*
    - *Solve the q(i,c) with these values*
    - *Update the matrices W_k^+ and W_k^-*
  - *Until the change in the new values of the weights is smaller than some predetermined valued.*

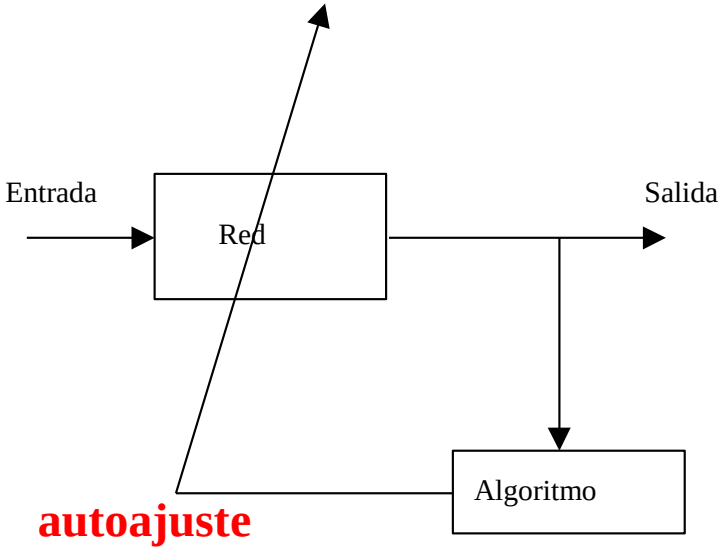
NO SUPERVISADO (AUTOORGANIZADO)

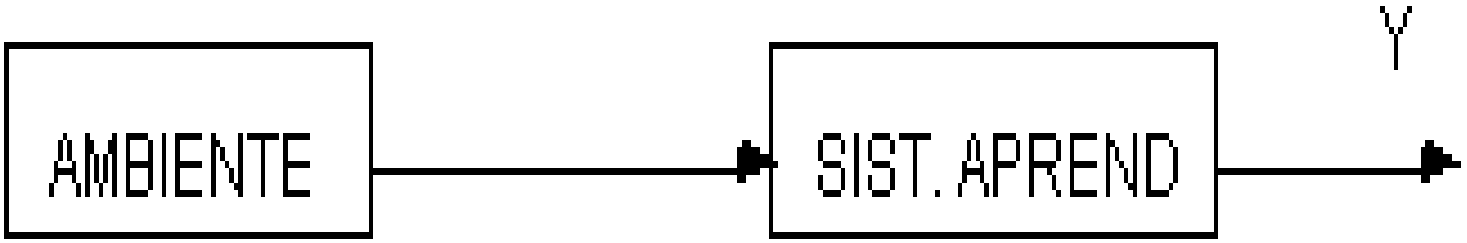
- NO RECIBE INFORMACIÓN DE SU ENTORNO (Se reciben patrones sin la respuesta deseada)

- CON LOS DATOS SE BUSCAN CORRELACIONES O REGULARIDADES EN EL CONJUNTO DE ENTRADAS:
  - EXTRAER RASGOS
  - AGRUPAR PATRONES SEGÚN SU SIMILITUD

- MAPAS AUTOORGANIZADOS

NO SUPERVISADO (AUTOORGANIZADO)





HEBBIANO

- MÁS VIEJO

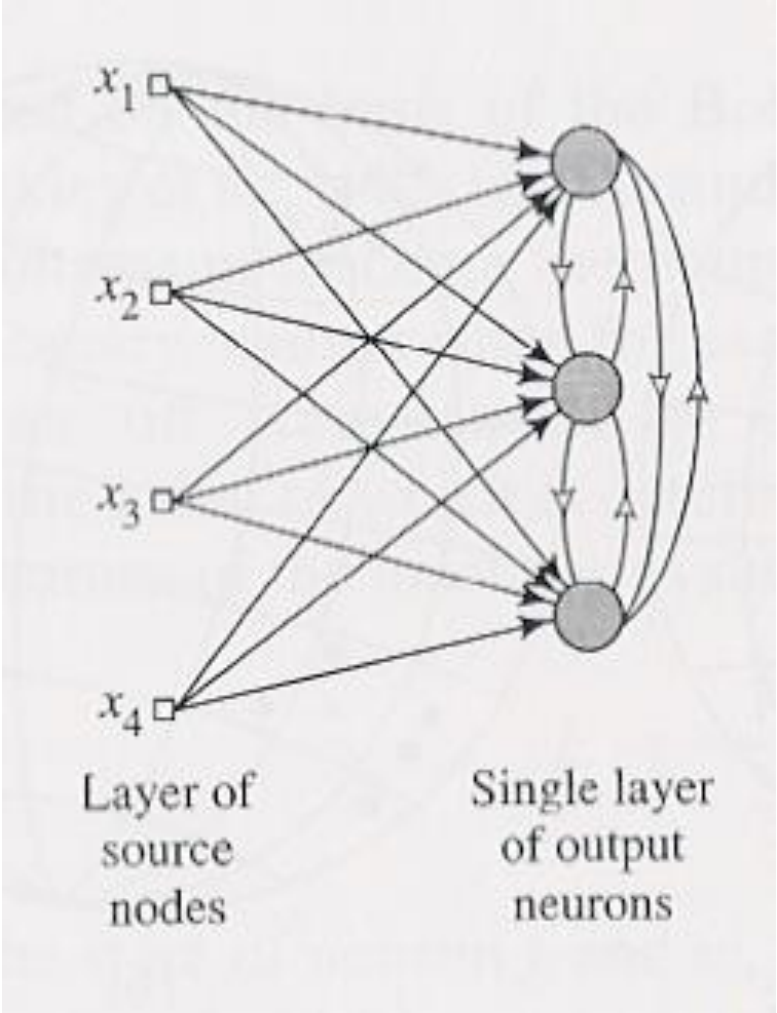
- DOS O MAS NEURONAS ACTIVADAS **SIMULTANEAMENTE**

=> REFORZAR LA CONEXIÓN ENTRE ELLAS

$$\Delta W_{ij} = \alpha Y_i Y_j$$

Algoritmos de Aprendizaje Competitivo

- Las neuronas de salida compiten para activarse.
- Solo una neurona de salida estará activa a la vez.
- Usadas en clasificación.



COMPETITIVO

A. *NEURONAS COMPITEN (COOPERAN PARA HACER UNA TAREA)*

B. *APRENDIZAJE*

=> NEURONAS QUE COMPITEN SE INHIBEN ,

NEURONAS QUE COOPERAN SE EXCITAN

=> Wij REFORZADO SI NEURONA j NO ES EXCITADA POR i Y VICEVERSA

$$\Delta Wij = \begin{cases} \alpha\,(Xi - Wij) \text{ SI NEURONA j GANA} \\ 0 \text{ SI NEURONA j PIERDE} \end{cases}$$

Algoritmos de Aprendizaje Competitivo

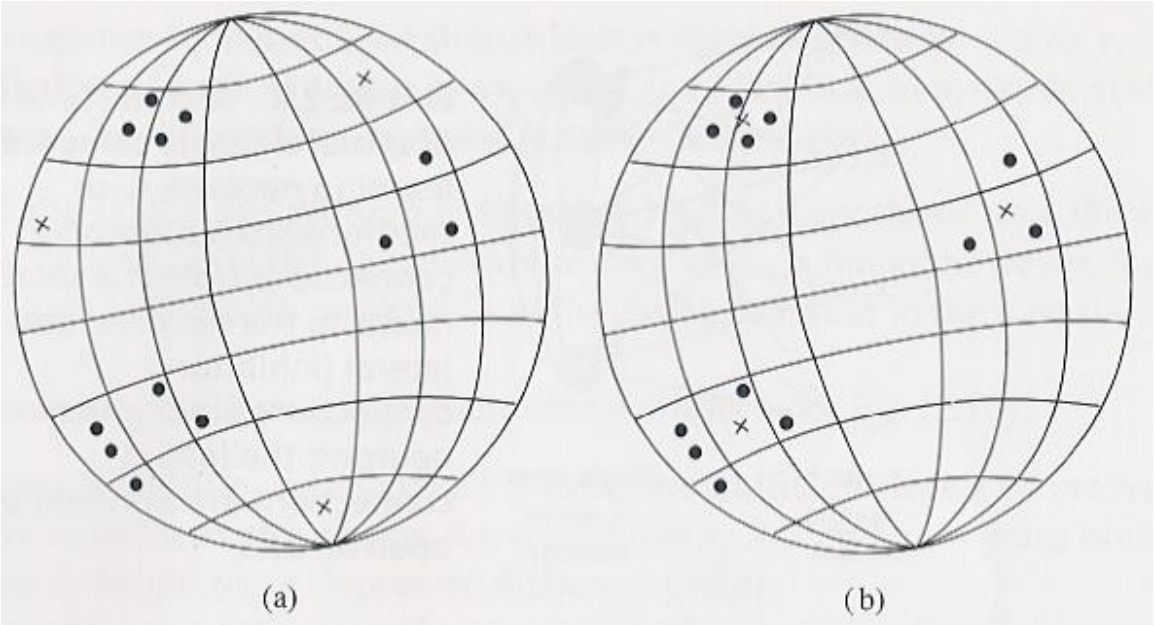
$y_k = 1$ si $v_k > v_j \quad \forall j \neq k$

$y_k = 0$ otro caso

$\Sigma_i w_{kj} = 1 \ \forall k$

$\Delta w_{kj} = \eta (x_j - w_{kj})$

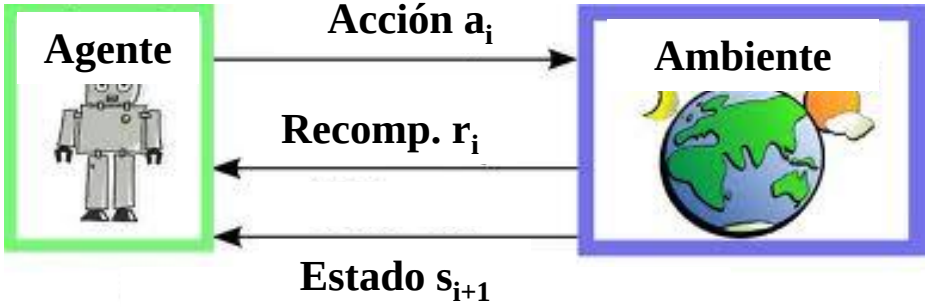
$\Delta w_{kj} = 0$



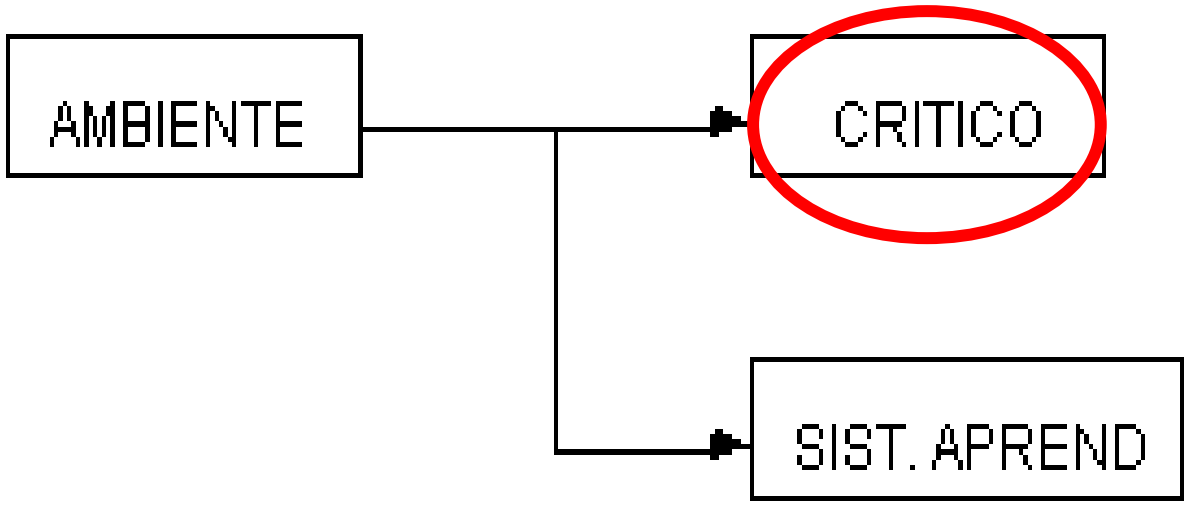
REFORZADO

Recompensa ocasional

- **SUPERVISOR INDICA SI SALIDA SE AJUSTA A LO DESEADO O NO** (que bien o mal se esta haciendo, no si es la salida deseada!!)
- **SUPERVISOR HACE PAPEL DE CRÍTICO MÁS QUE DE MAESTRO** (premio-castigo)

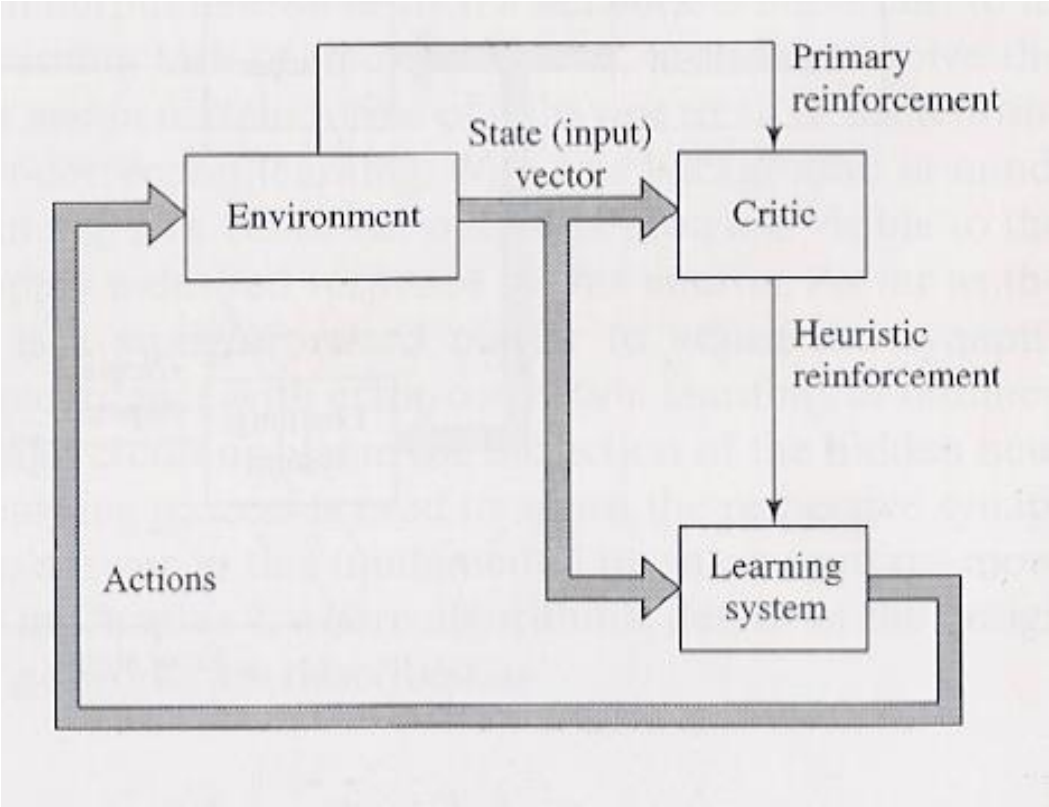


REFORZADO



Paradigmas de Aprendizaje
Por Refuerzos

1. Minimizar un **índice escalar de aptitud**
2. **Refuerzo retardado**
3. **Asignación de Crédito y Culpa** a acciones
4. **Aprende a realizar tareas** basado solamente en el resultado de **sus experiencias**



REFORZADO

- Particularmente útiles en los ámbitos en los que exista información de reforzamiento (expresado como penalizaciones o recompensas) proporcionada después de una secuencia de acciones realizadas en el ambiente.
- Métodos comunes: Q-Learning y diferencia temporal-(TD)
  - **Q-Learning:** aprende la utilidad de llevar a cabo **acciones que me lleven a ciertos estados**,
  - **TD** aprender la utilidad de **estar en ciertos estados**.

REFORZADO

- Todos los métodos de aprendizaje por refuerzo están inspirados en
  - Fórmulas de actualización de la utilidades esperadas
  - Exploración del espacio de estados.
- La actualización es a menudo una suma ponderada de:
  - Valor actual utilidad,
  - Refuerzo obtenido al realizar una acción y
  - Utilidad esperada por el siguiente estado alcanzado, después se realiza la acción.

Aprendizaje Reforzado

- Retroalimentación:
  - Premio o penalización
- Ejemplo:
  - Animales reconocen dolor, placer,

    $\Rightarrow$ forma de entrenamiento de los animales

$\Rightarrow$ Diseño:

Agente considera una esperada utilidad (acción-valor) de tomar una acción en un estado dado (aprendizaje reforzado)

- Tres casos:
  - **Se aprende una función de utilidad** y se usa para seleccionar la mejor acción
  - **Se aprende una función acción-utilidad:** esperada utilidad por la acción
  - **Se aprende mapeo estado-acción**

REFORZADO

Señal de refuerzo

$$W_{ji}(t)=W_{ji}(t-1)+\alpha(r-\sigma_j)e_{ji})$$

donde

$$e_{ji}=\partial n(g_i)/\partial W_{ji} \qquad g_i=Pr(y_j=b_{kj}/W_{ji}, A_k)$$

Ejemplos de Reforzamiento

$$R_t = r_{t+1} + \mu R_{t+1}$$

función de utilidad

$$\theta_{t+1} = \theta_t + \eta \left[r + \mu \left(P(x_{t+1}, \theta_t) - P(x_t, \theta_t) \right) \right] \frac{\partial P(x_t, \theta_t)}{\partial \theta}$$

Donde:

θ_t es un vector de parámetros en el tiempo t.
x_t es un vector de datos de entrada en el tiempo t.
r es la señal de refuerzo.
$P(x_t, \theta_t)$ es una función de predicción.
$0 <= \mu < 1$ es un parámetro de peso.
η es la tasa de aprendizaje.

Aprendizaje Reforzado

Esqueleto de un sistema basado en aprendizaje reforzado (e)

Aprende tabla de utilidades (estado-acción)

- añadir *e* a la percepción
- incrementar N[estado[e]]
- M=f(percepción, N)
- Si [e] es terminal entonces
  - U=actualizar (U, percepción)
- de lo contrario
  - Ir inicio

N: frecuencia de estados

U: tabla de utilidades estimadas

M: transición entre estados

percepción: secuencia percibida

(1,1), (1,2), (1,3), (1,4), (2,4) -> -1

(1,1), (1,2), (1,3),(2,3), (3,3), (3,4) ->+1

(1,1), (2,1), (3,1),(3,2), (3,3), (3,4) ->+1

Movimientos de un robot en un cuadrado

Aprendizaje Reforzado

Aprendizaje Activo: considera que acciones tomar y como ellas afectan a la premiación

(función acción-utilidad)

$$U(i)= R(i)+Max_a [\sum_j M^a(i, j)U(j)]$$

Toma decisión!!

- Algoritmo
  - añadir *e* a percepción
  - $M[i, j]=R(i)+ Max_a [\sum_j M^a(i, j)U(j)]$
  - Si [e] es terminal entonces
    - U=actualizar (U, M, percepción)
  - de lo contrario
    - Ir inicio

R premios

a posible desde e

M: transición entre estados

percepción: secuencia percibida

Aprendizaje Reforzado

Algoritmo Q-Learning automático

función de utilidad

- Inicializa Q(s,a) arbitrariamente
- Repetir
  - Inicializa s
  - Repetir
    - Seleccione a' para s' tal que

$$Q(s,a)=Q(s,a)+ ß[r + \alpha \max_{a'}[Q(s',a')-Q(s,a)]]$$

    - *s = s';*
- hasta que s es estado terminal

Tareas de Aprendizaje

- Aproximación
- Asociación
  - Autoasociativa
  - Heteroasociativa
- Clasificación
- Predicción
- Control
- Filtraje

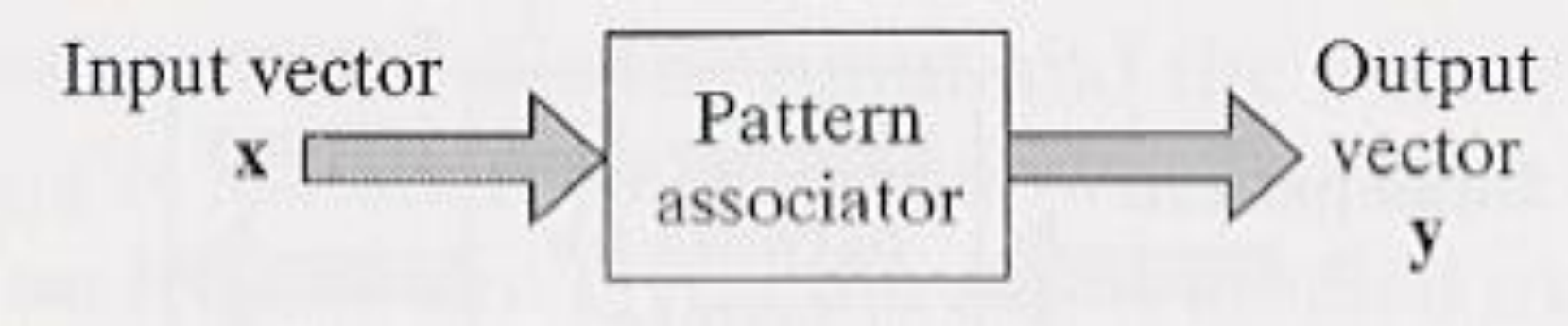
planta:u(t),y(t) modelo:r(t),d(t) lim|d(t)-y(t)|=0

Tareas de Aprendizaje
Asociación de Patrones

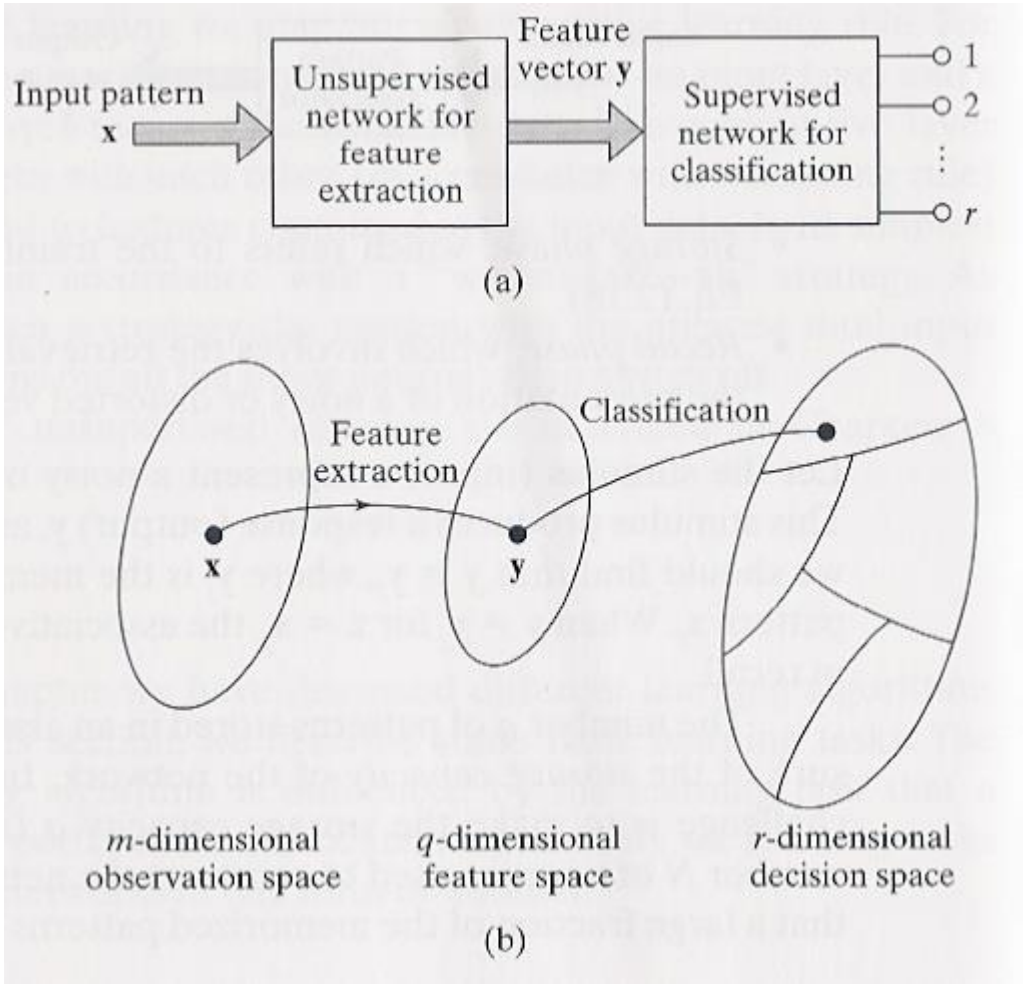
- Memoria Asociativa
- Almacenar Patrones
- Patrones con ruido
- Recordar patrones

$x_k \rightarrow y_k$, k=1, 2, ... , q

- Autoasociativo: $x_k = y_k$
- Heteroasociativo: $x_k \neq y_k$
- Meta de Diseño:
  Recordar + patrones con - neuronas



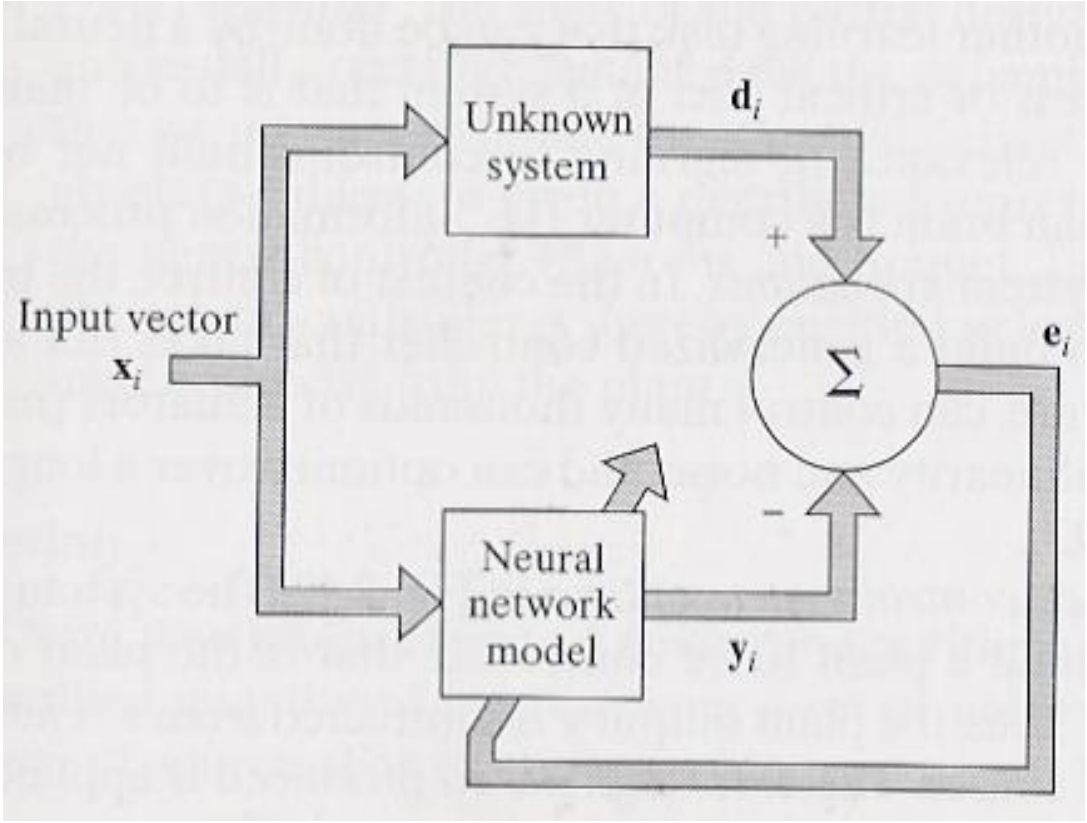
Tareas de Aprendizaje
Reconocimiento de Patrones



Clasificación

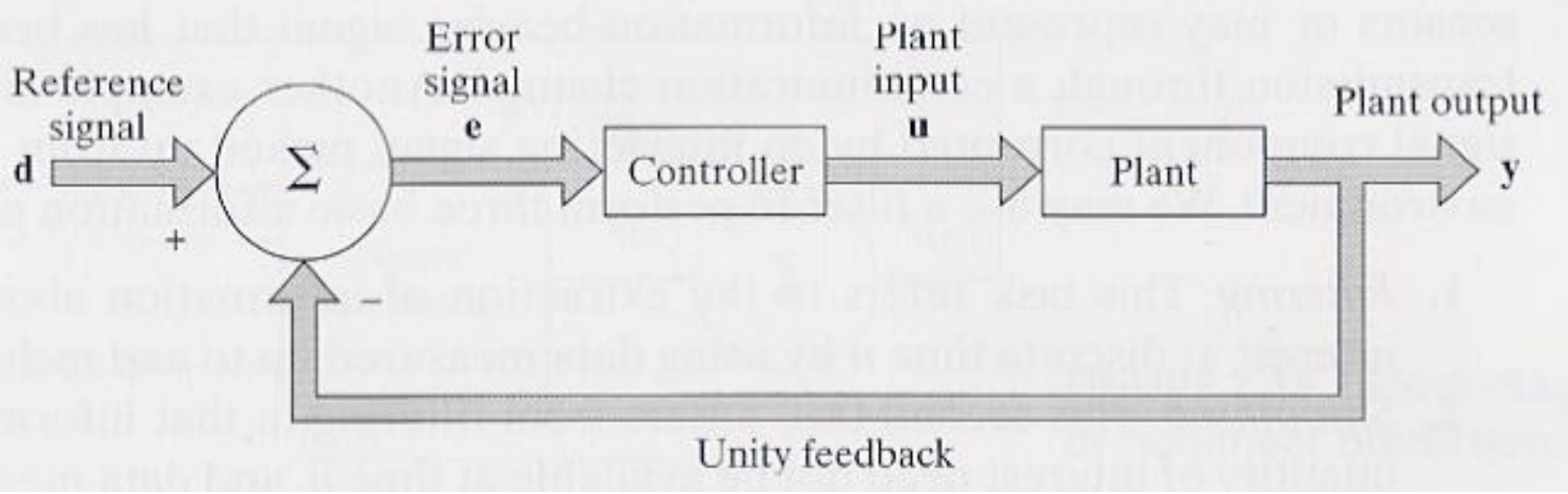
Tareas de Aprendizaje
Aproximación Funcional

- d = f(x)
- T = {(xi, di)}, i=1, ..., N
- F aproximación de f
- Meta: $|F(x) - f(x)| < \varepsilon, \forall x$



Tareas de Aprendizaje
Control

- d = f(x)
- T = {(xi, di)}, i=1, ..., N
- F aproximación de f
- Meta: $|F(x) - f(x)| < \varepsilon, \forall x$

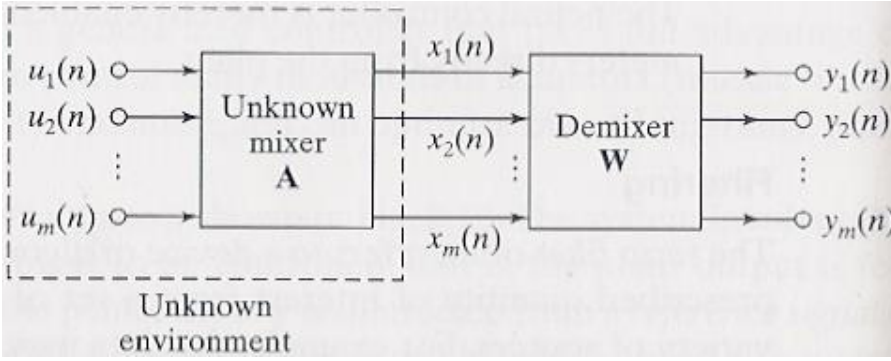


Tareas de Aprendizaje
Filtrado

- Filtrado. Extracción de información de una variable en el tiempo discreto n, usando mediciones hasta n.

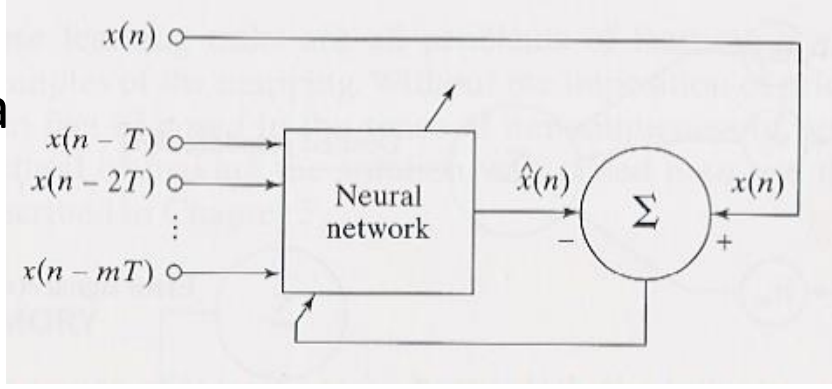
  Smoothing. Filtrado con retardo.

Problema del Cocktail



Predicción no Lineal.

- Predicción. Derivar información acerca del futuro de una variable, usando información pasada.



Aprendizaje profundo

algoritmos en aprendizaje automático para RNA multicapas.

- Los algoritmos pueden utilizar aprendizaje supervisado o no supervisado.

- Basados en el **aprendizaje de múltiples niveles de características**. Las características de más alto nivel se derivan de las características de nivel inferior para formar una representación jerárquica.

- Aprenden múltiples niveles de representación que corresponden a **diferentes niveles de abstracción**, que forman una jerarquía de conceptos.

Aprendizaje profundo

¿por qué es generalmente mejor?

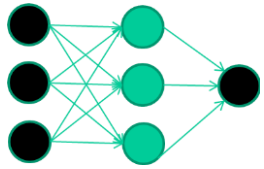
- Utilizan una red neuronal con **varias capas de nodos** entre la entrada y salida
- La serie de capas entre la entrada y salida hace una **identificación de característica y procesamiento en una serie de etapas**, al igual que nuestro cerebro .

Aprendizaje profundo

- Redes neuronales multicapas tienen muchísimos años.

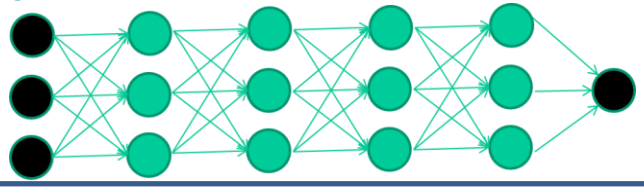
Qué es lo nuevo?

siempre han habido buenos algoritmos para el aprendizaje de la pesos en redes con 1 capa oculta

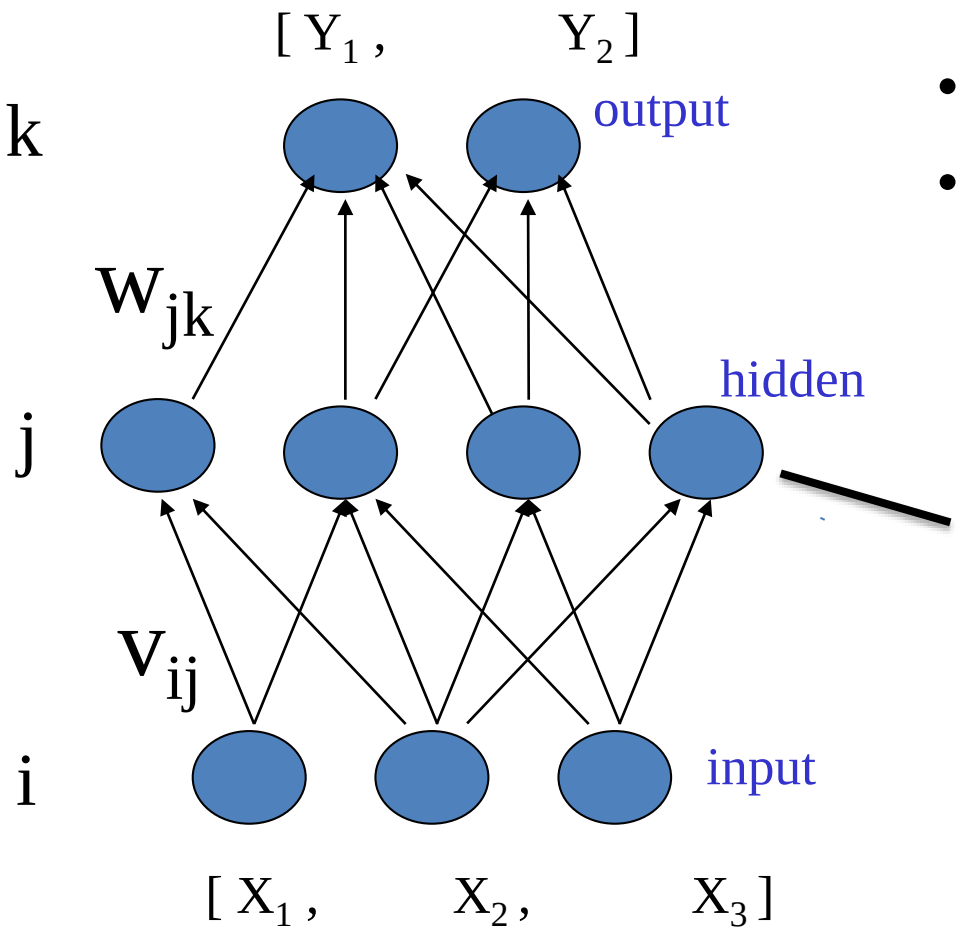


pero estos algoritmos no son buenos para aprender las ponderaciones de redes con más capas ocultas

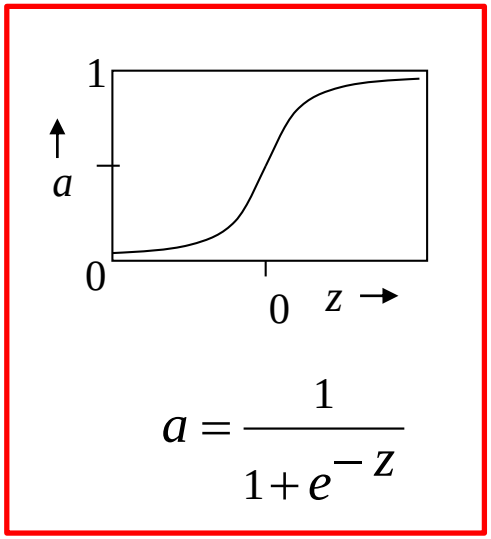
Lo nuevo es: algoritmos para entrenar redes "mucho-más tarde"



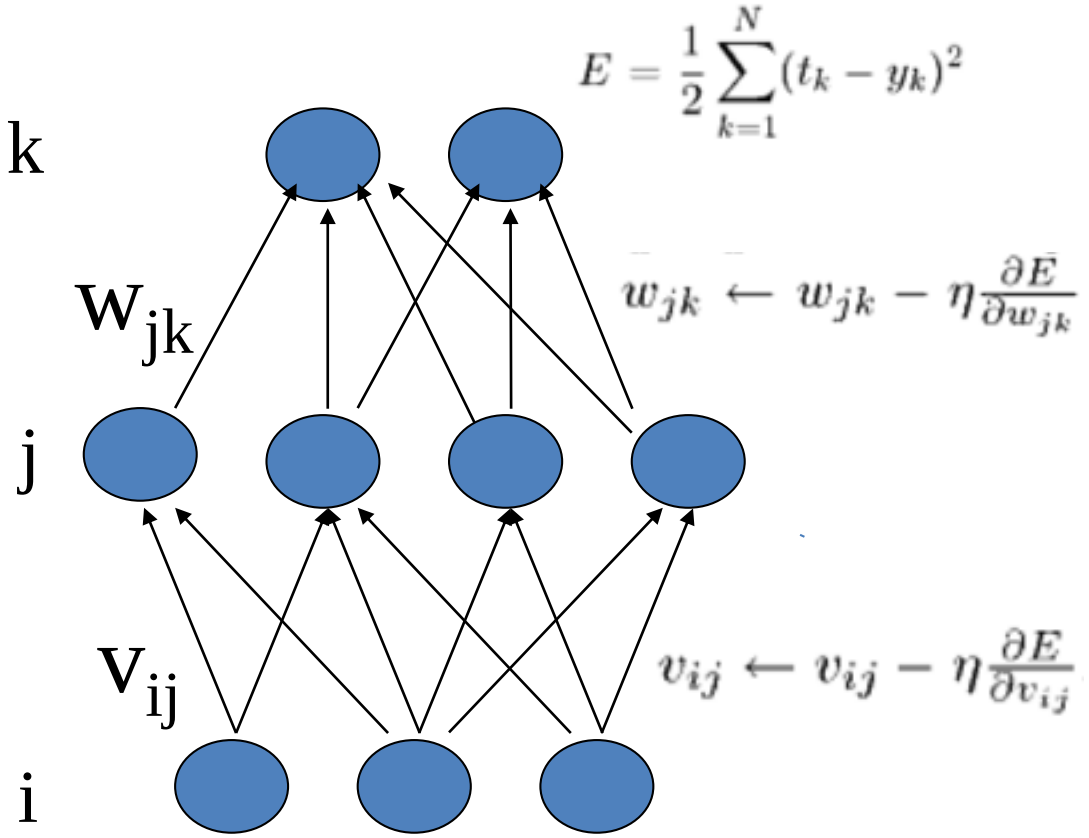
Perceptron Multicapa



- Multiples capas
- Hacia adelante
- Pesos de Conexión $z_j = \sum_i x_i w_{ij}$



Backpropagation



$$E = \frac{1}{2}\sum_{k=1}^{N}(t_k - y_k)^2$$

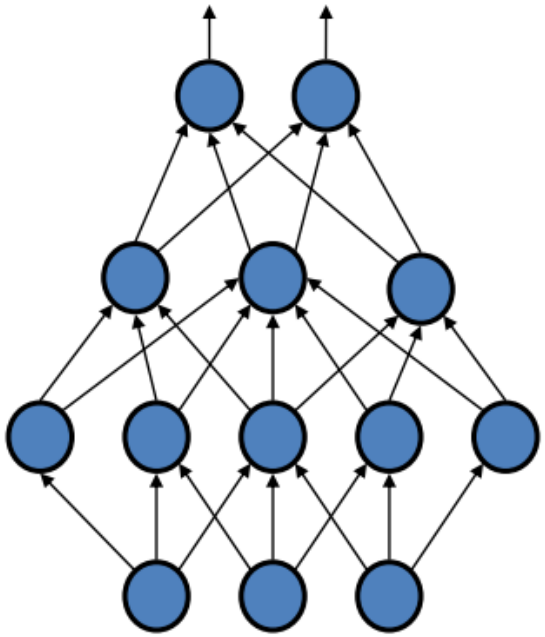
$$w_{jk} \leftarrow w_{jk} - \eta \frac{\partial E}{\partial w_{jk}}$$

$$v_{ij} \leftarrow v_{ij} - \eta \frac{\partial E}{\partial v_{ij}}$$

- Minimiza error de salida
- Ajuste de pesos
  - Descenso de Gradiente
- Pasos
  - Hacia adelante
  - Retropropagación del error
- Para cada ejemplo, Varias corridas

Problemas con Backpropagation

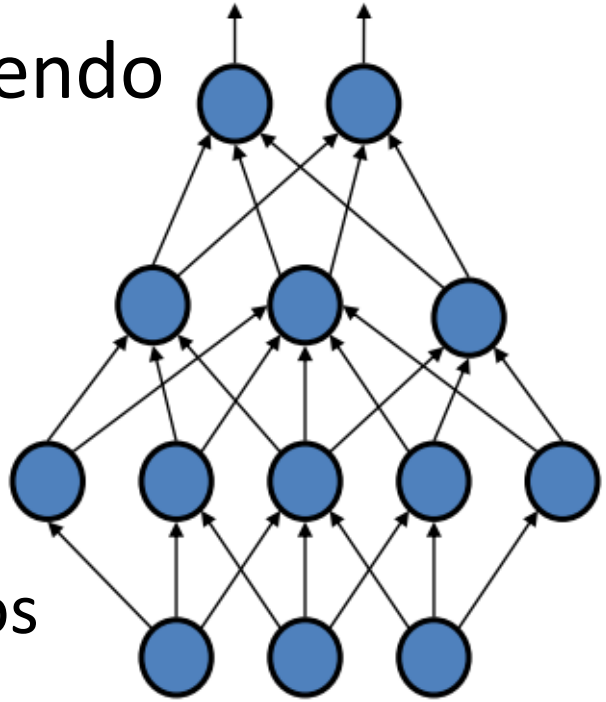
- **Multiples capas escondidas**
- **Cae en optimos locales**
  - Segun donde se inician pesos
- **Converge lento al optimo**
  - Necesita mucho entrenamiento
- **Solo usa datos etiquetados**
  - La mayoria de los datos son no etiquetados



Otro enfoque

Arquitectura Deep

- Multiples capas escondidas
- Backpropagation, como construyendo bloques
- Motivación
  - aprendizaje jerárquico
    - funciones cada vez más complejas
  - trabajar bien en diferentes dominios
    - Visión, audio, ...



Aprendizaje supervisado






Carros





Motocicletas

Aprendizaje semi-supervisado








Imágenes sin etiquetas (solo carros/motocicletas)

Carro

Motocicleta

Aprendizaje autodidacta





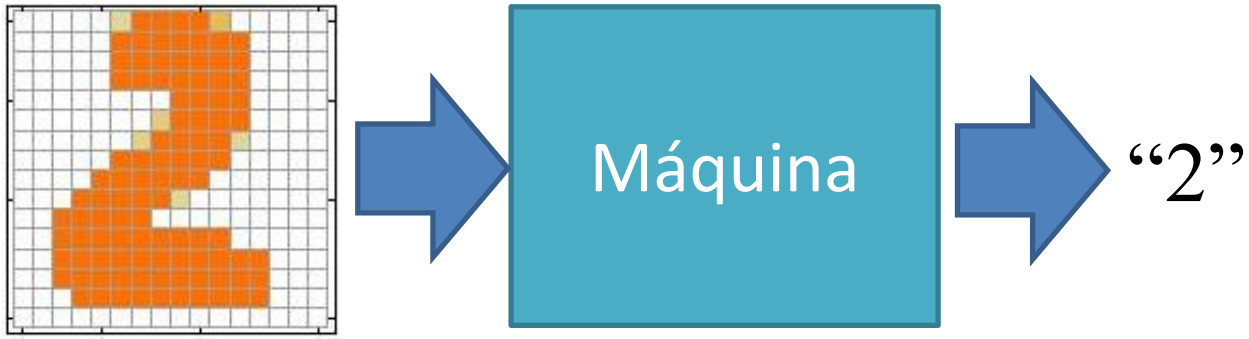

Imágenes sin etiquetas (p.e., tomadas de internet)

Carro

Motocicleta

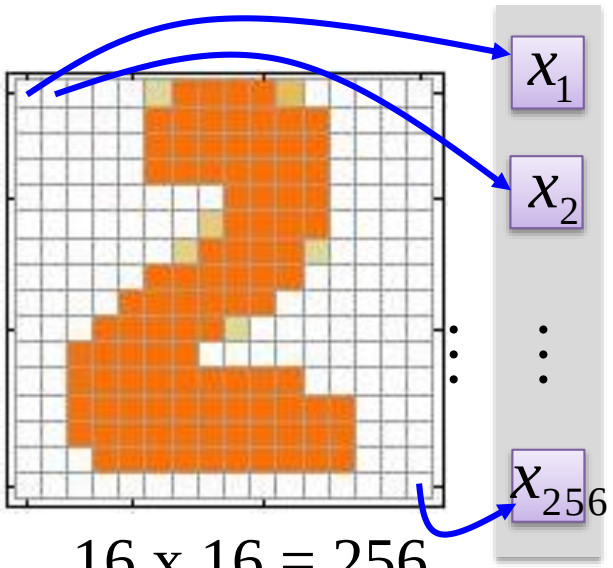
Ejemplo

- Reconocimiento de dígitos escritos



Reconocimiento de dígitos escritos

Entrada

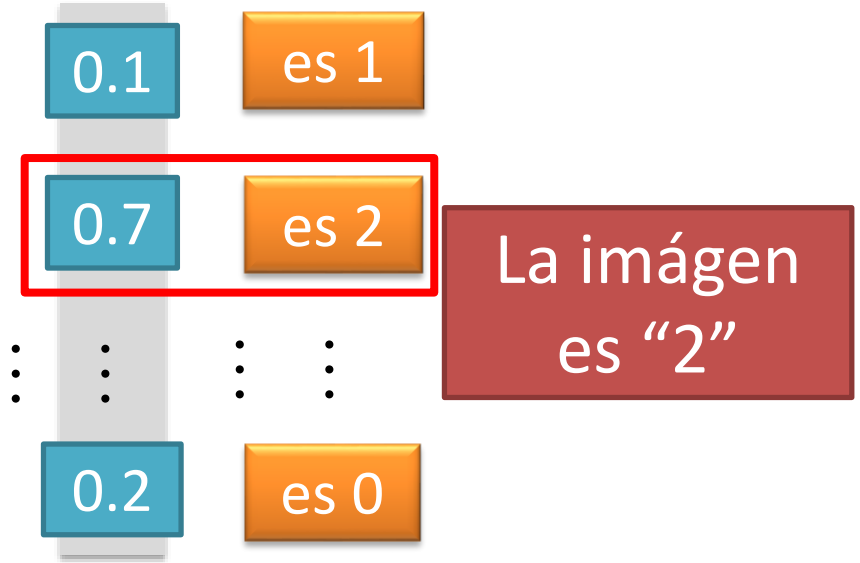


16 x 16 = 256

Ink → 1

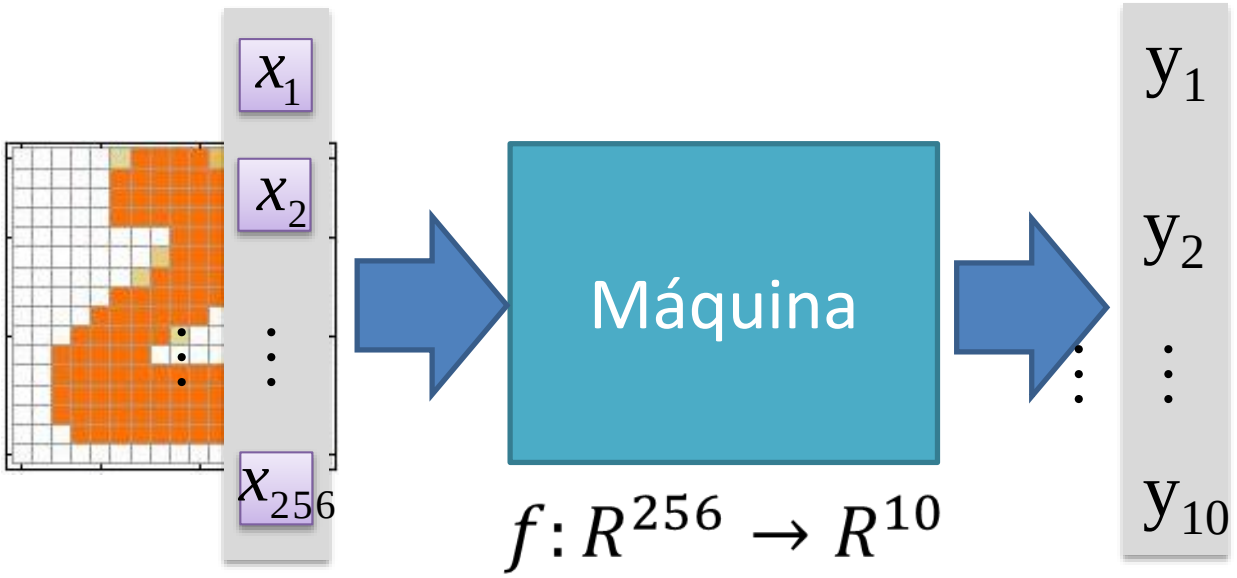
No ink → 0

Salida



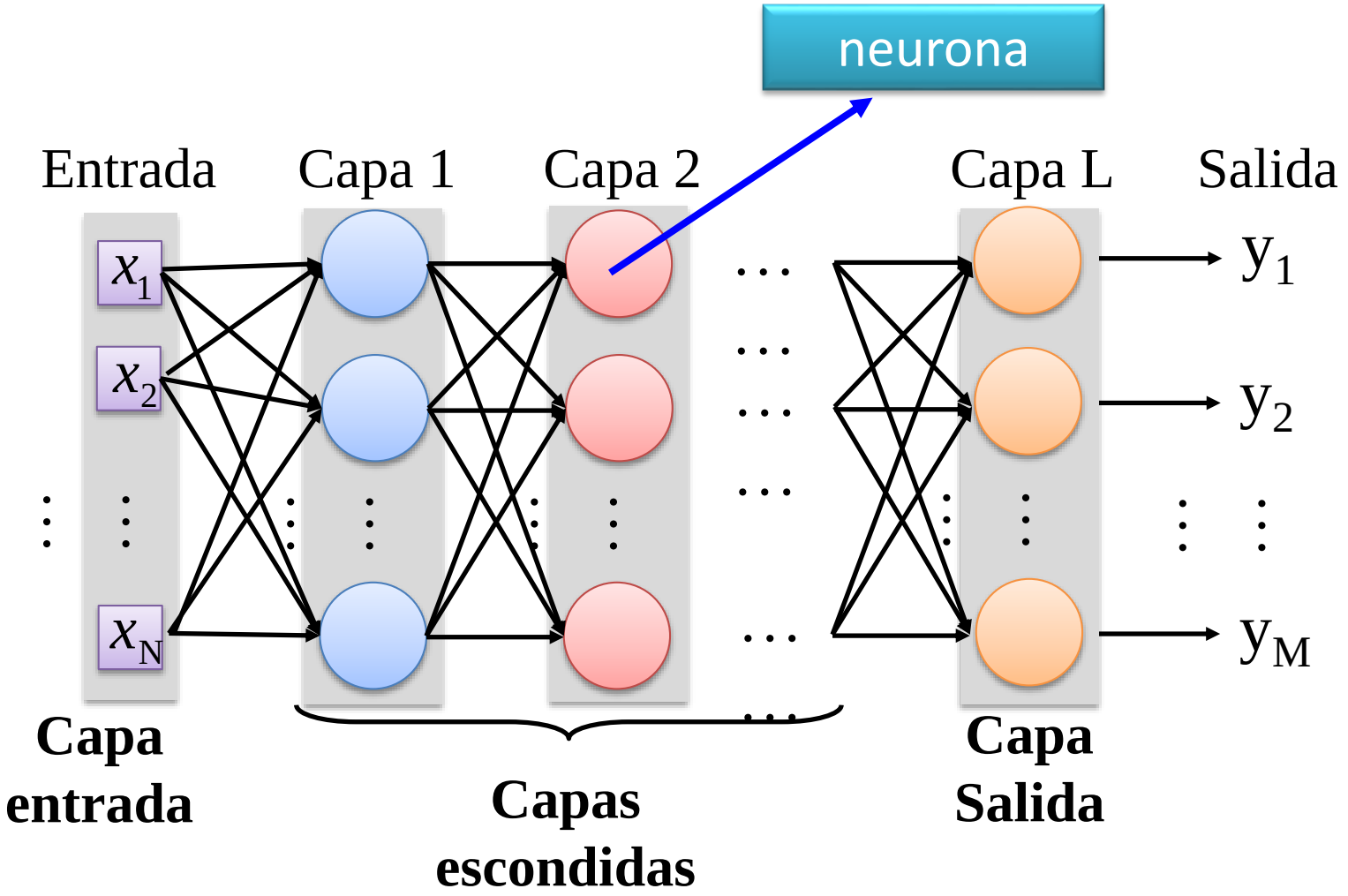
Cada dimensión representa la confianza de un dígito.

Reconocimiento de dígitos escritos



En aprendizaje profundo, la función f es una RNA

RNA



Deep significa muchas capas ocultas

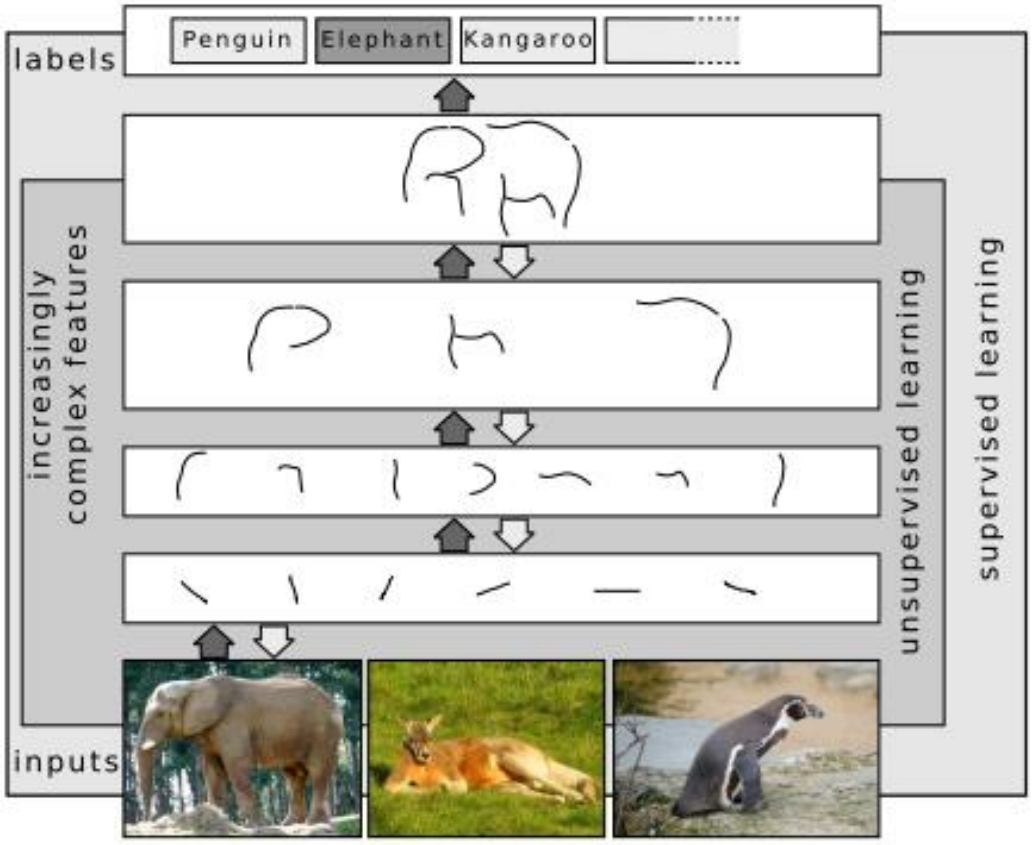
¿Más profunda es mejor?

Capa x tamaños	Tasa error palabra (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

No sorprende, más parámetros, un mejor rendimiento

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

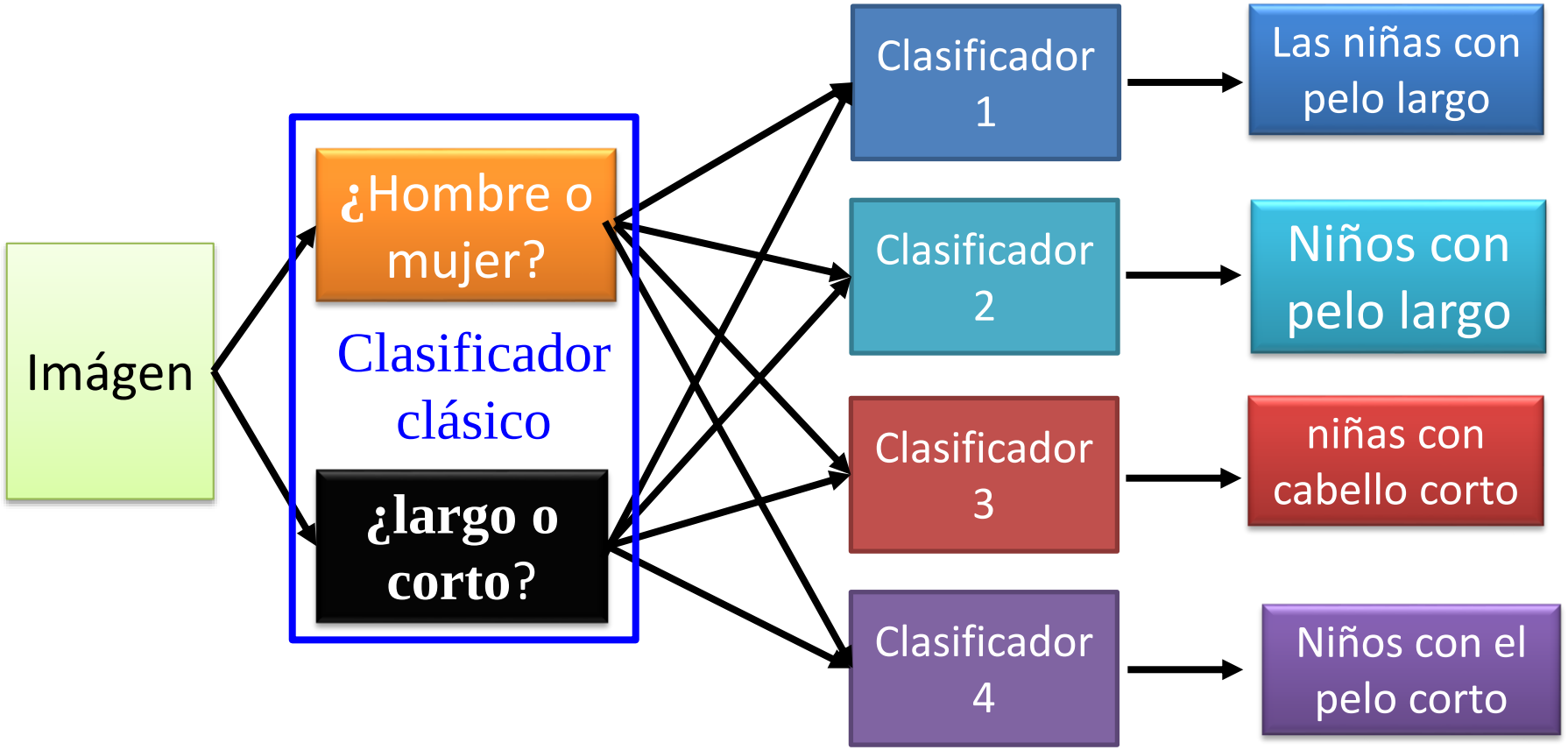
Aprendizaje jerárquica



- progresión natural desde el bajo nivel a la estructura de alto nivel, como se ve en la complejidad natural
- Más fácil de controlar lo que se está aprendiendo y guiar la máquina a mejores subespacios

¿Por qué Deep?

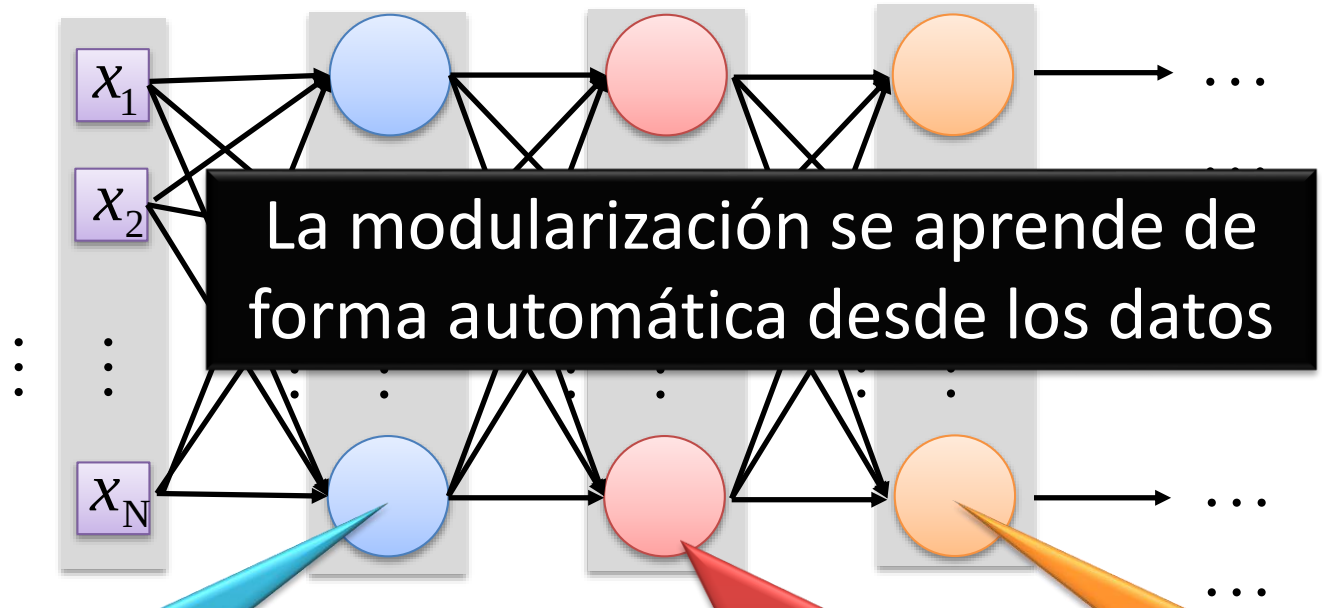
- Modularidad

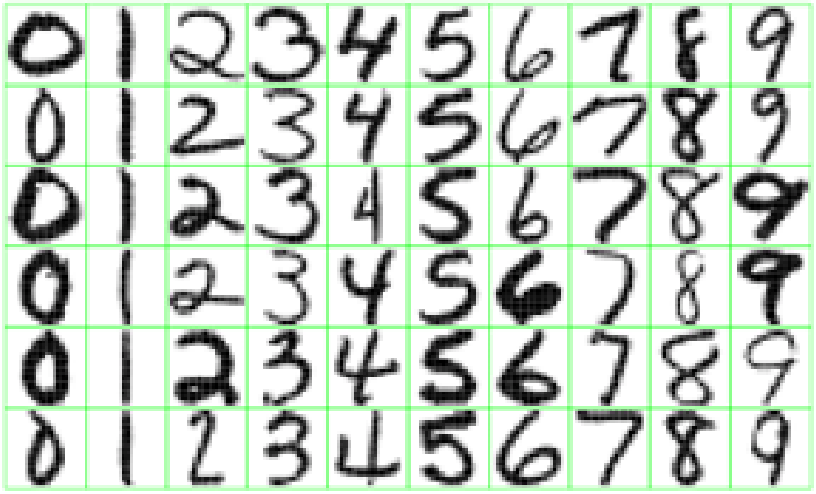


¿Por q

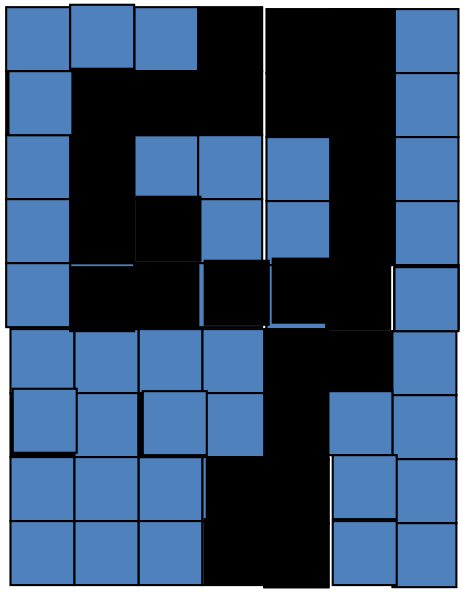
trabaja con conjuntos pequeños de datos

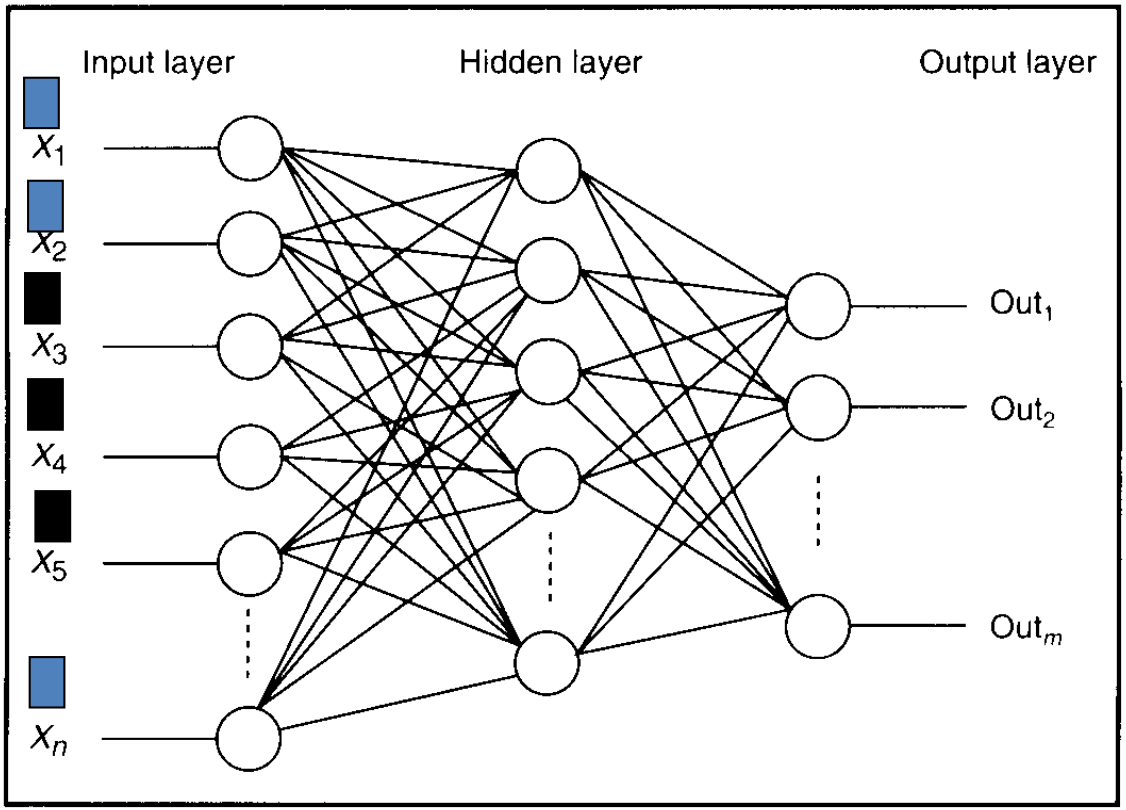
- Modularidad

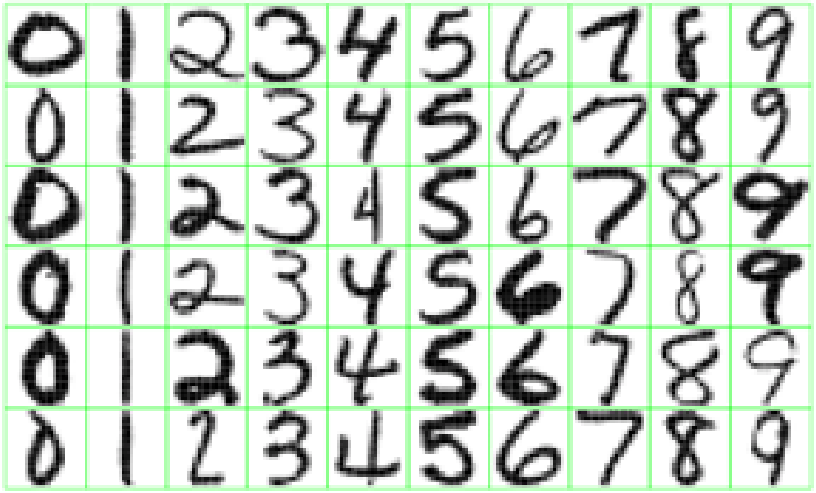




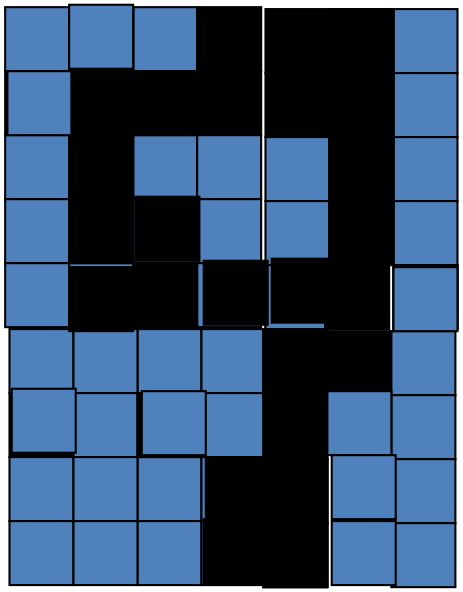
detectores de características

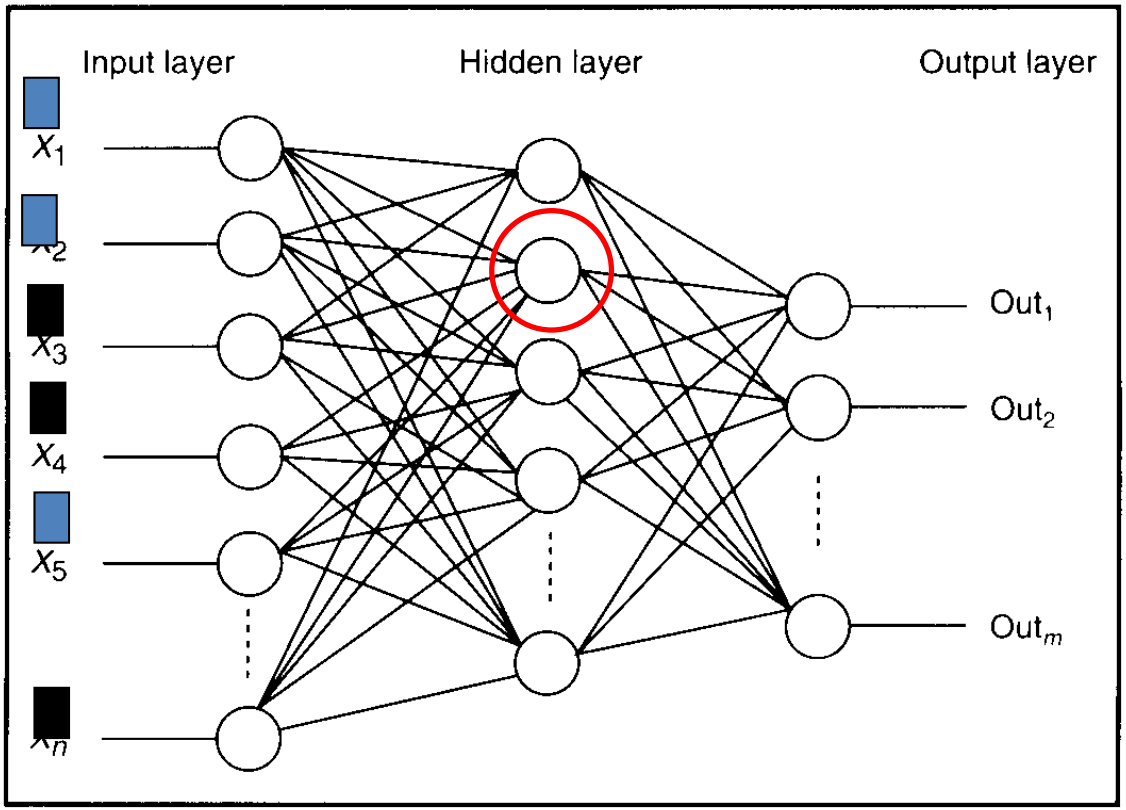




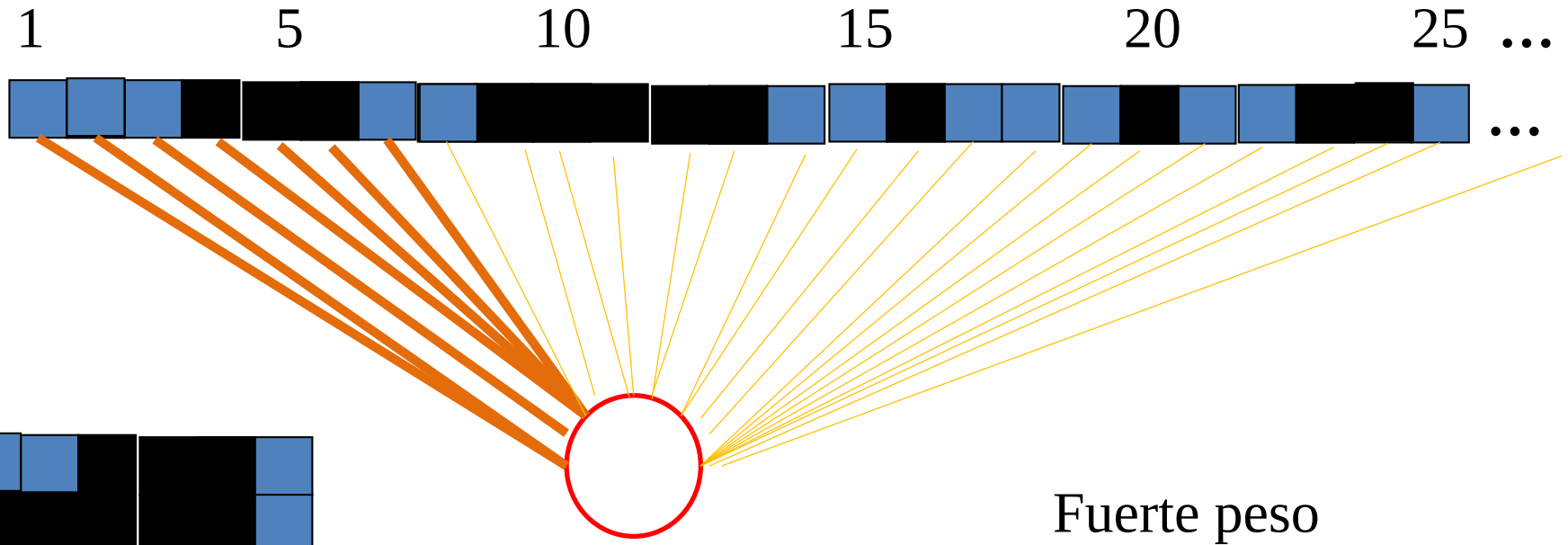


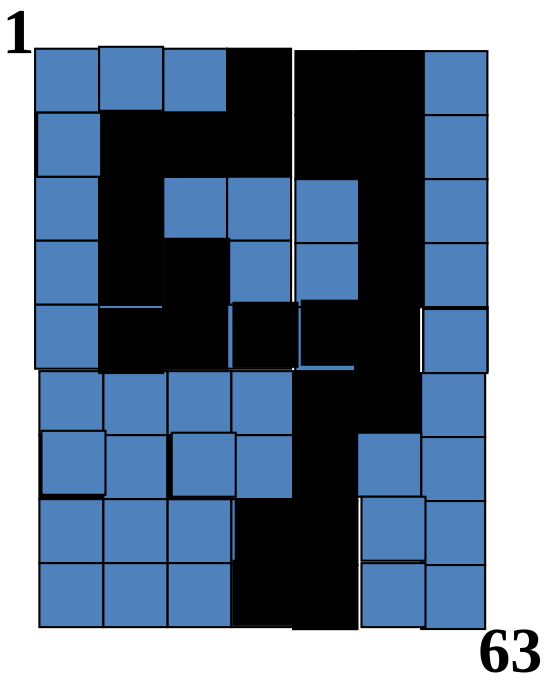
¿Qué hace?




Input layer
Hidden layer
Output layer
X_1
X_2
X_3
X_4
X_5
X_n
Out_1
Out_2
Out_m

Redes Capas ocultas son detectores auto-organizadas de características

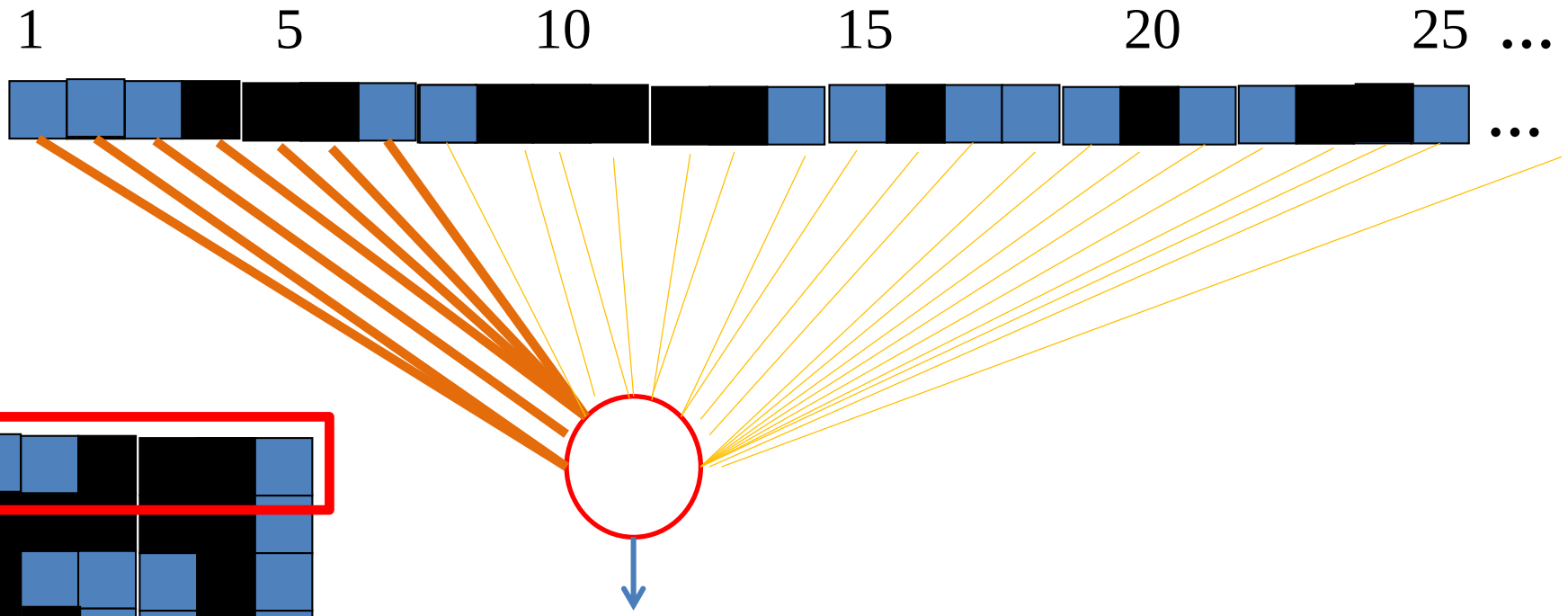


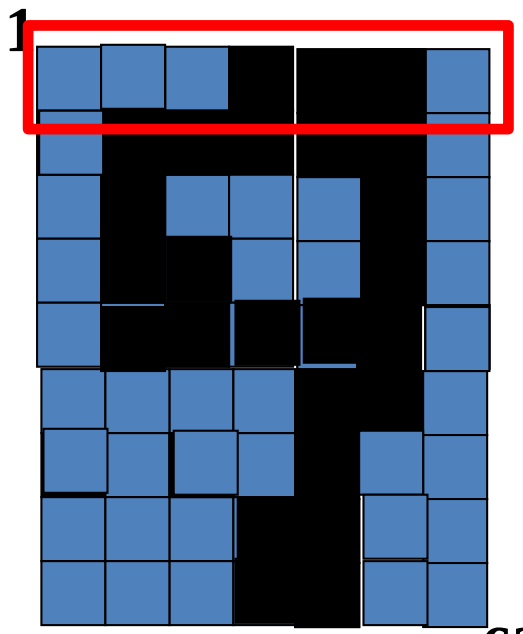


Fuerte peso

Bajo peso

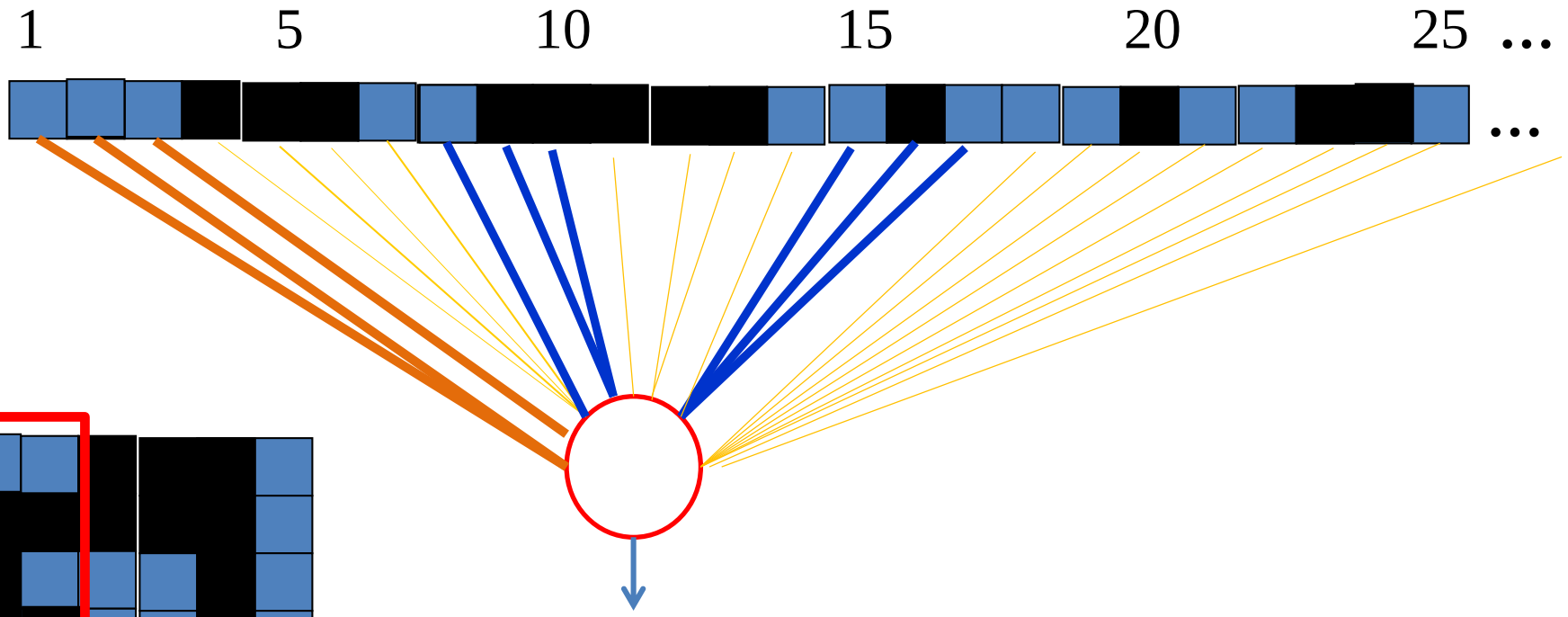
¿Qué detecta?

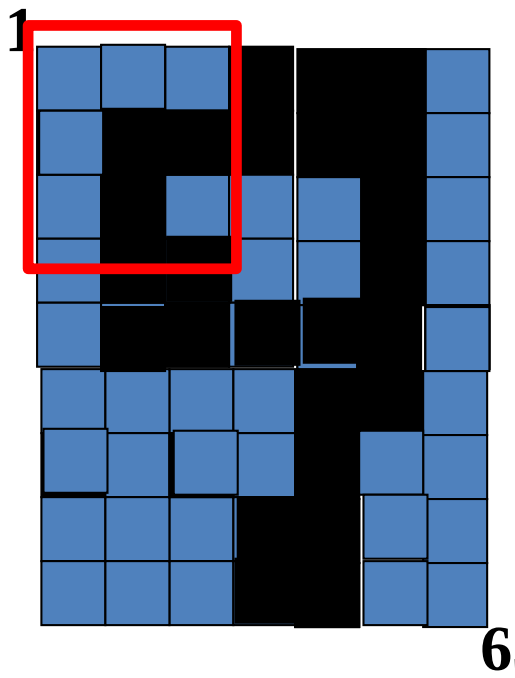




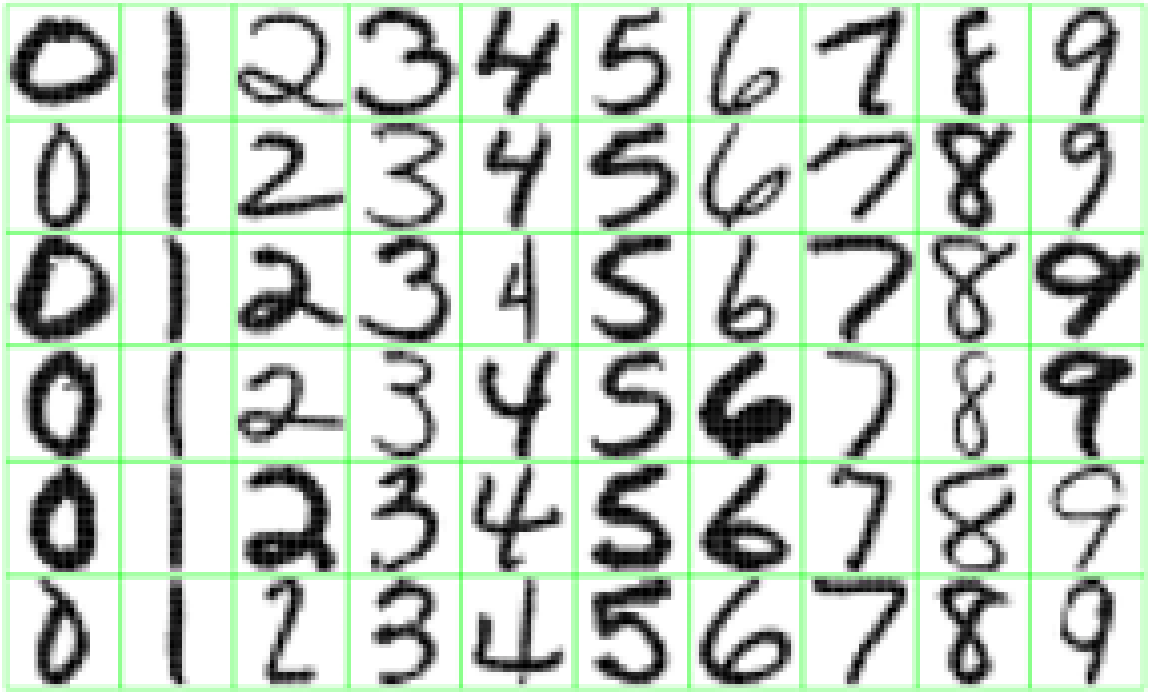
Enviará la señal fuerte para una línea horizontal en la fila superior, haciendo caso omiso de todos los demás sitios

¿Qué detecta?

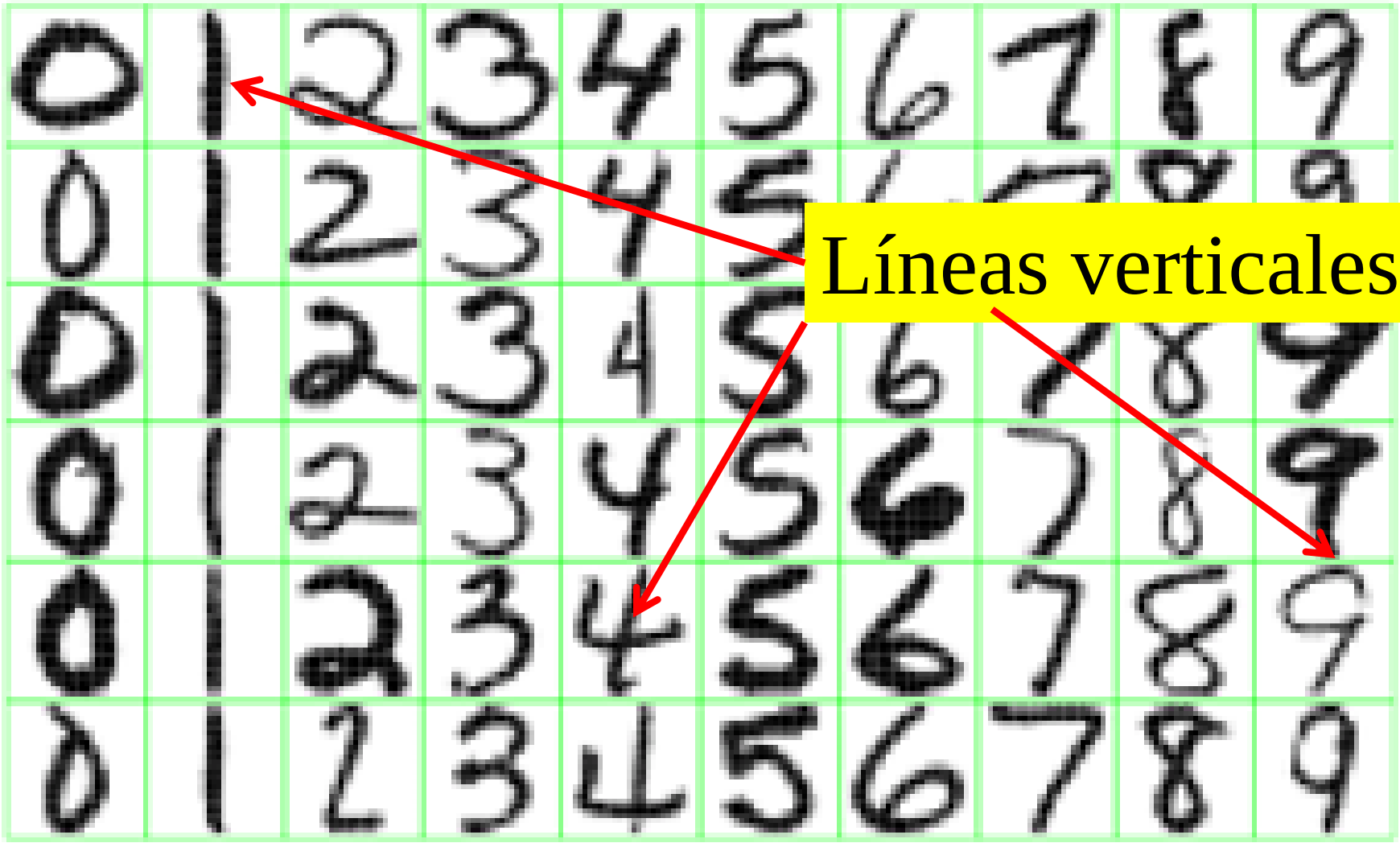




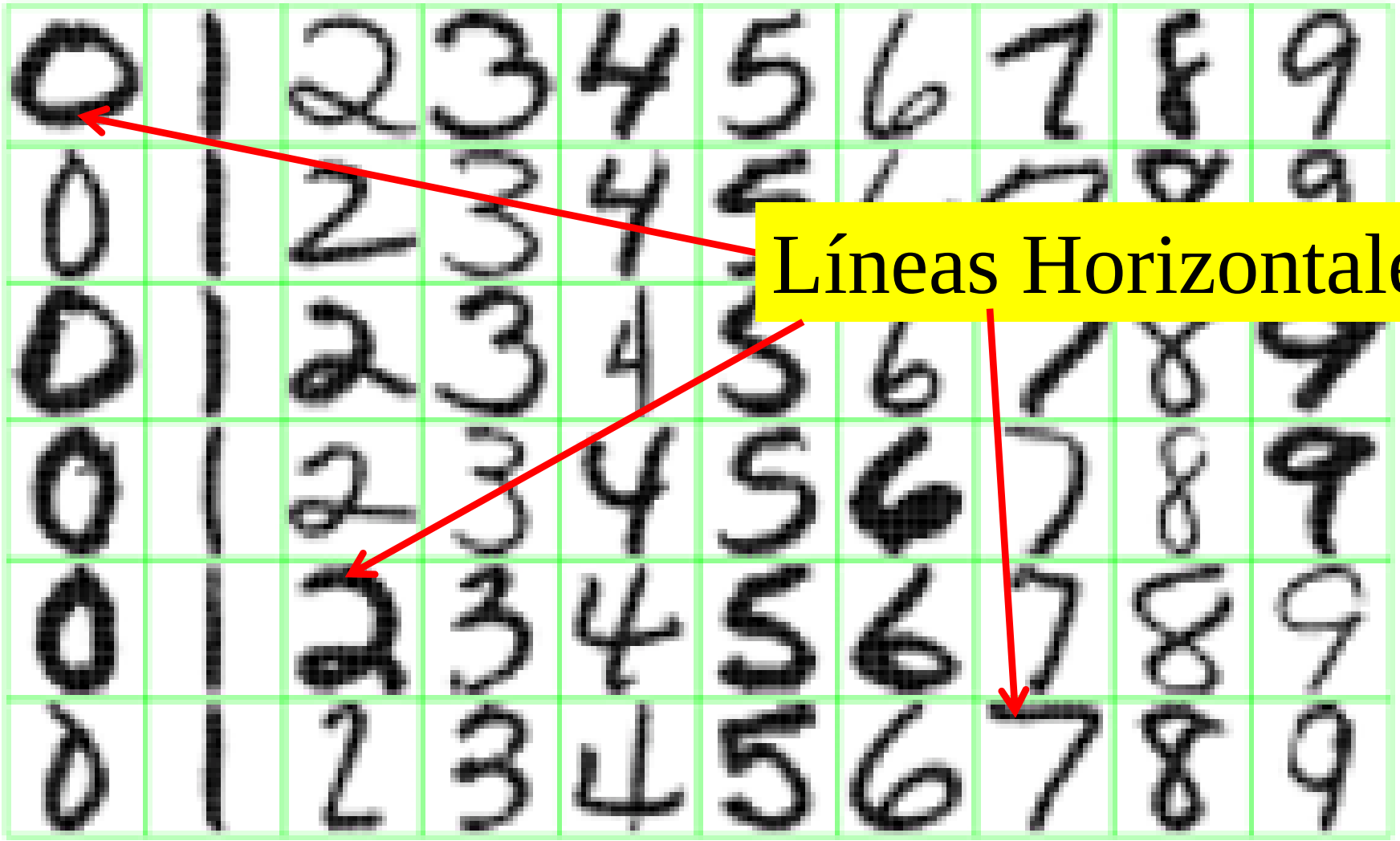
señal fuerte para un área oscura en la esquina superior izquierda



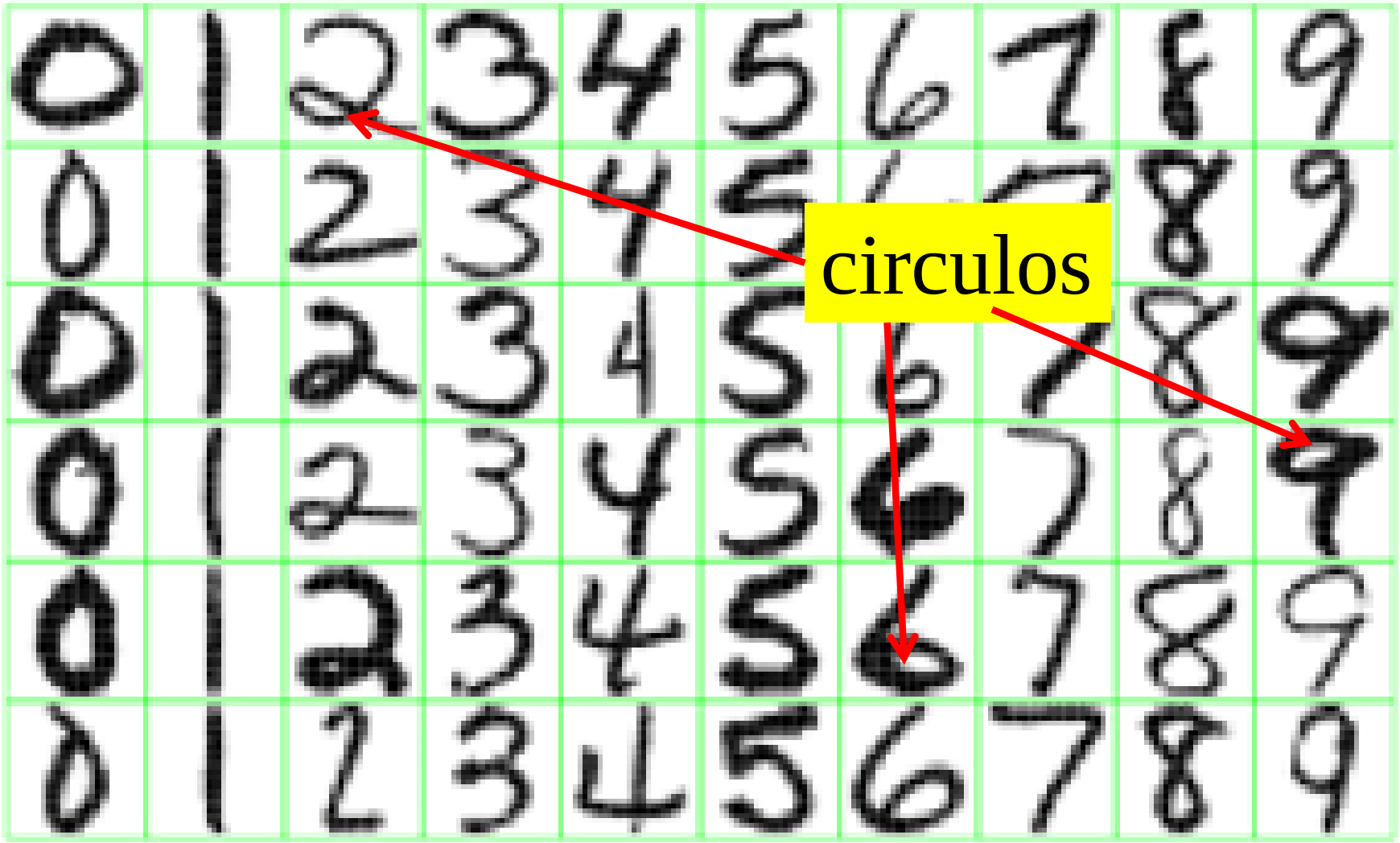
¿Qué características podría esperar una buena NN aprender, entrenados con datos como este?



1

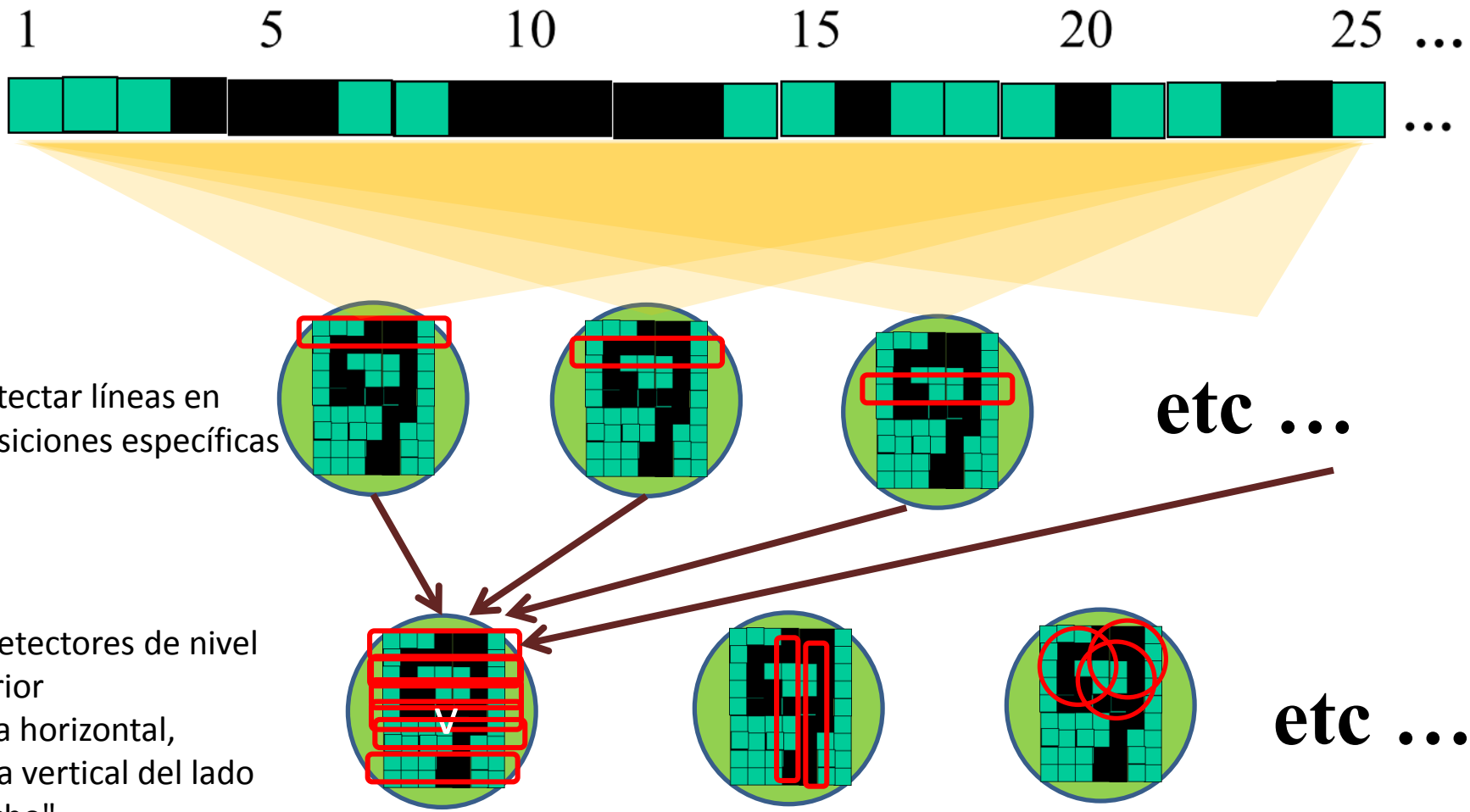


1

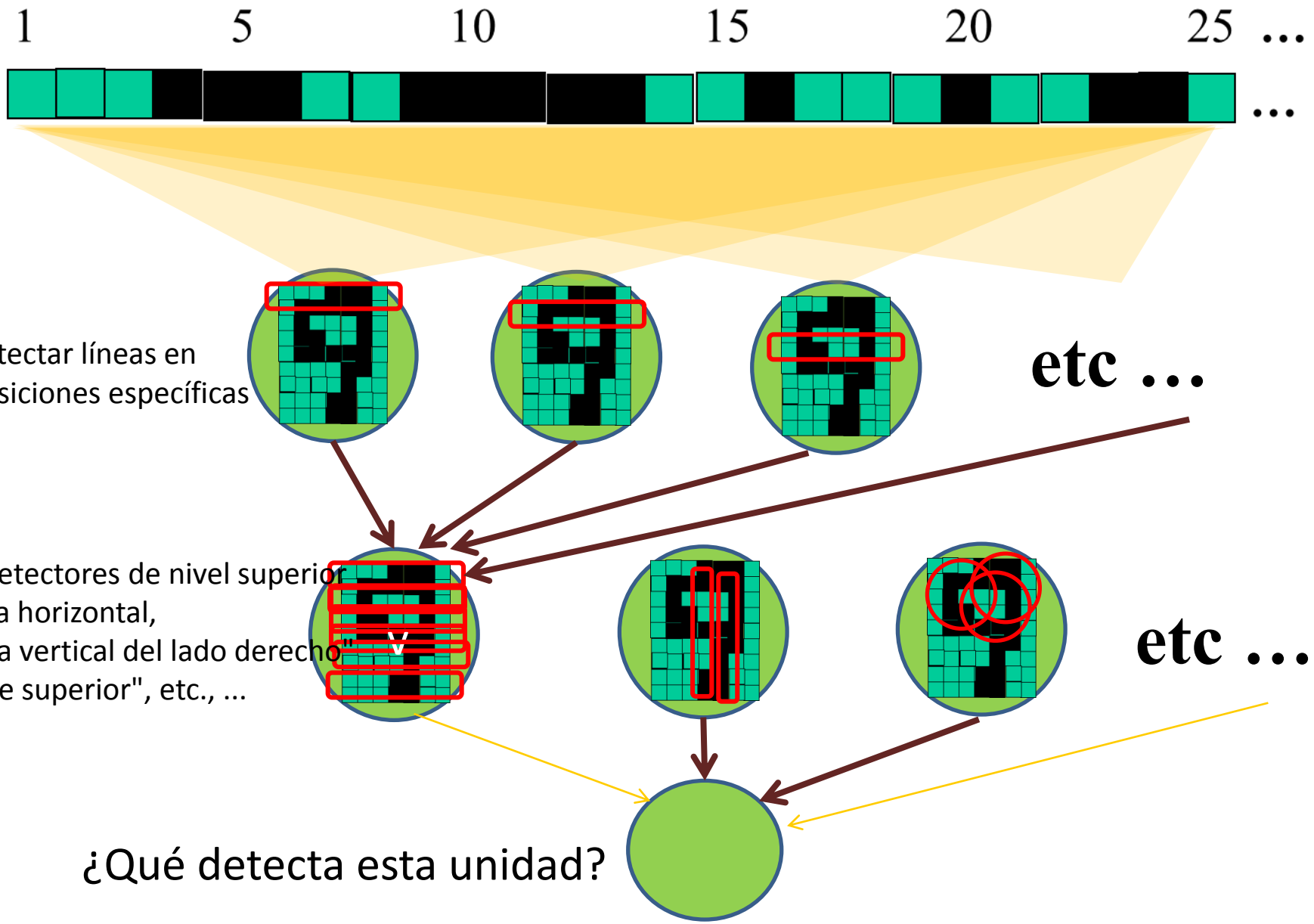


1

capas sucesivas pueden aprender características de nivel superior …



capas sucesivas pueden aprender características de nivel superior …



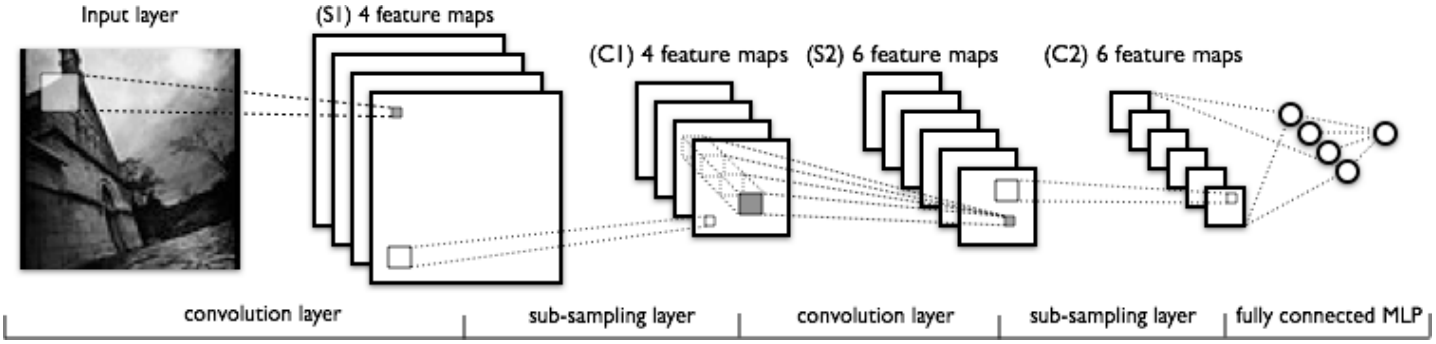
Algunas Deep Learning NN

- Convolutional neural networks
- Recursive neural networks
- Deep belief networks
- Deep Boltzmann machines
- Deep Q-networks

Convolutional Neural Networks

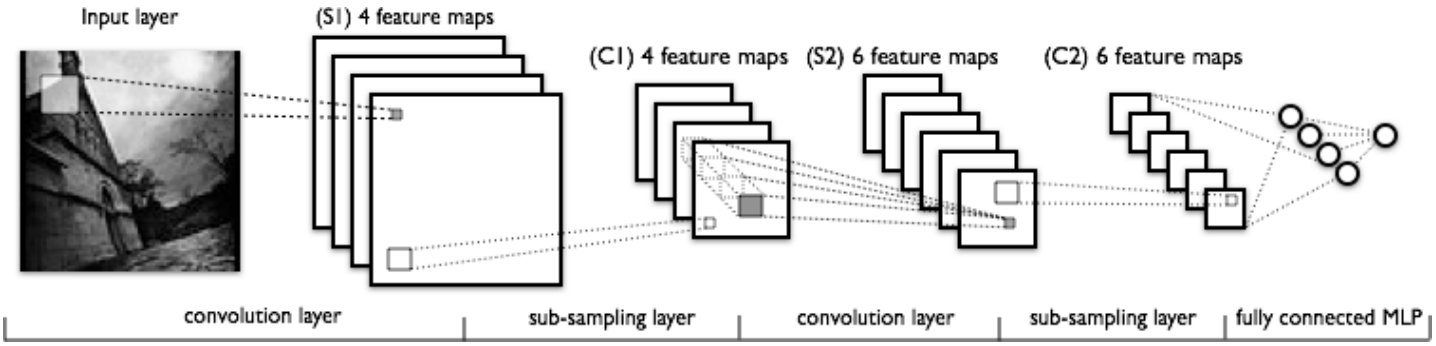
Mientras que los nodos en una estándar RNA obtienen información de todos los nodos de la capa anterior, **en CNN un nodo recibe sólo un pequeño conjunto de características que son espacial o temporalmente cercanas unos de otros**, llamados **campos receptivos** de una capa a la siguiente (por ejemplo, 3x3, 5x5) , haciendo cumplir así la capacidad de manejar la estructura local 2-D.

Puede definir bordes, esquinas, puntos finales,



CNN

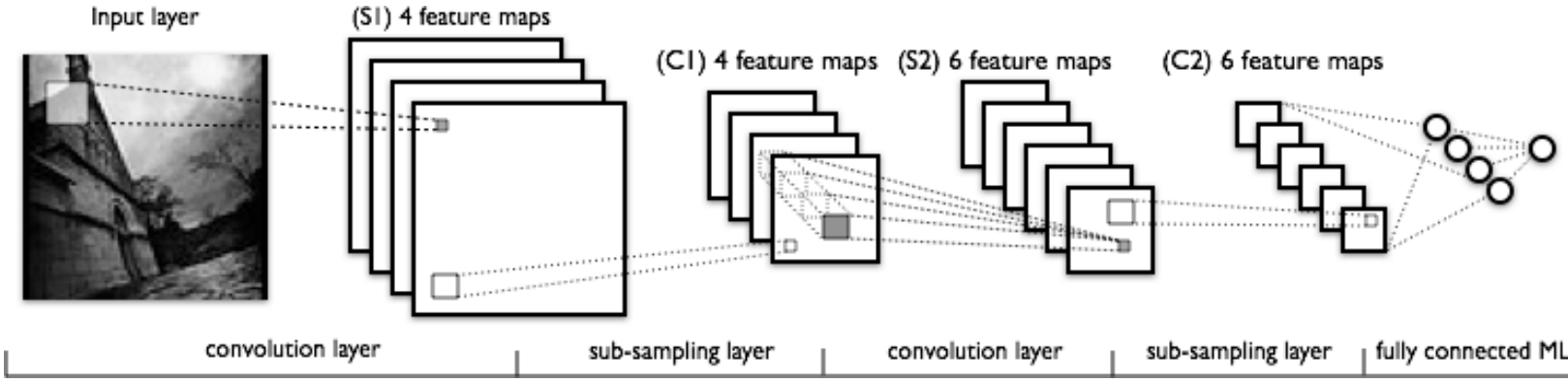
- En la capa de submuestreo, cada nodo es calculado para cada campo receptivo en la capa anterior
  - Durante el entrenamiento, así, un número relativamente pequeño de parámetros se aprenden,
  - Cada nodo de salida de CNN es sigmoide ($\Sigma xw + b$)
  - mapas de características múltiples en cada capa
  - Cada mapa de características aprende una característica invariable diferente traducción
- Capa de convolución mantiene el total de características



Submuestreo (agrupaciones)

- Capas de convolución y sub-muestreo se intercalan

- Submuestreo agrupa varias características
  - Reduce la resolución espacial, y por lo tanto, disminuye naturalmente importancia de una característica,
  - la agrupación de 2x2 haría compresión 4: 1, 3x3 9: 1, etc.
  - El agrupamiento suaviza los datos



Representación de características

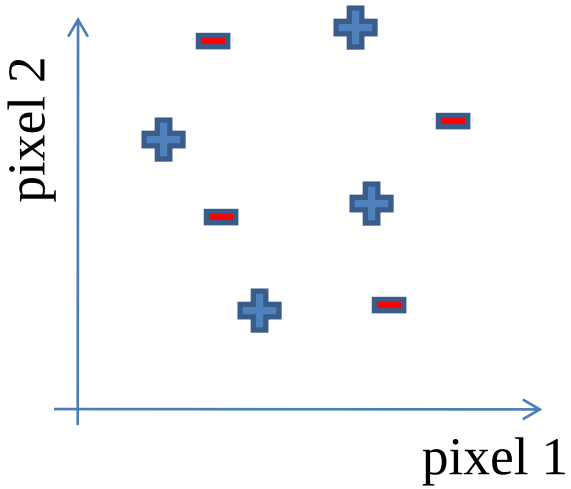
Algoritmo aprendizaje

Entrada

Espacio entrada

Motos

"No"-Motos



Representación de características

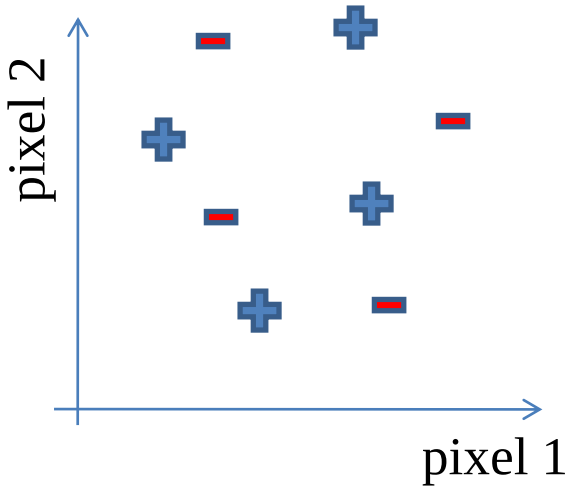
Entrada

representación de características

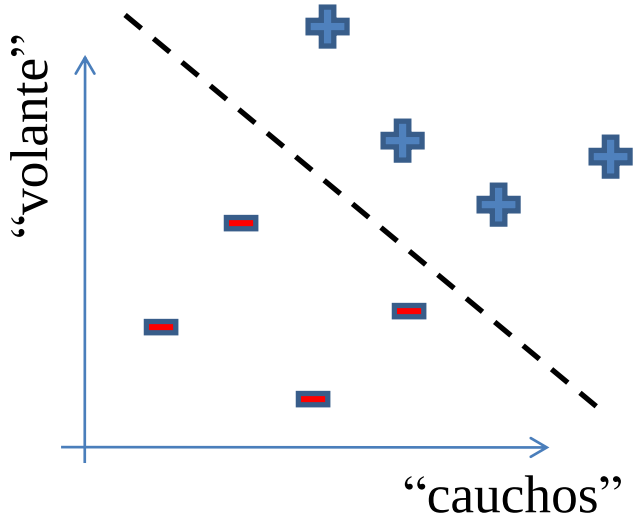
Algoritmo aprendizaje

Espacio entradas

Motos
Non"-Motorbikes

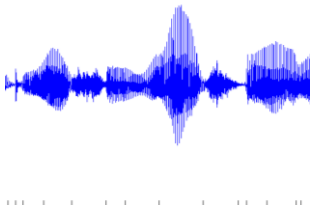


Espacio Características

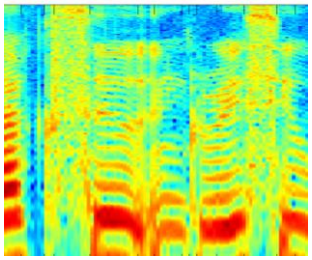


Representación de características

Clasificación audio



Audio

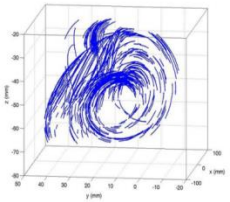


características de audio de bajo nivel

Identificación de quien habla

Control helicóptero

Helicóptero



características de estado de bajo nivel

Acción

Aplicaciones Deep Learning

Ingeniería:

- Visión por Computador (por ejemplo, la clasificación de imágenes, segmentación)
- Reconocimiento de voz
- Procesamiento del Lenguaje Natural (por ejemplo, el análisis de opiniones, la traducción)
- Sistemas de recomendación
- bioinformática

Ciencia:

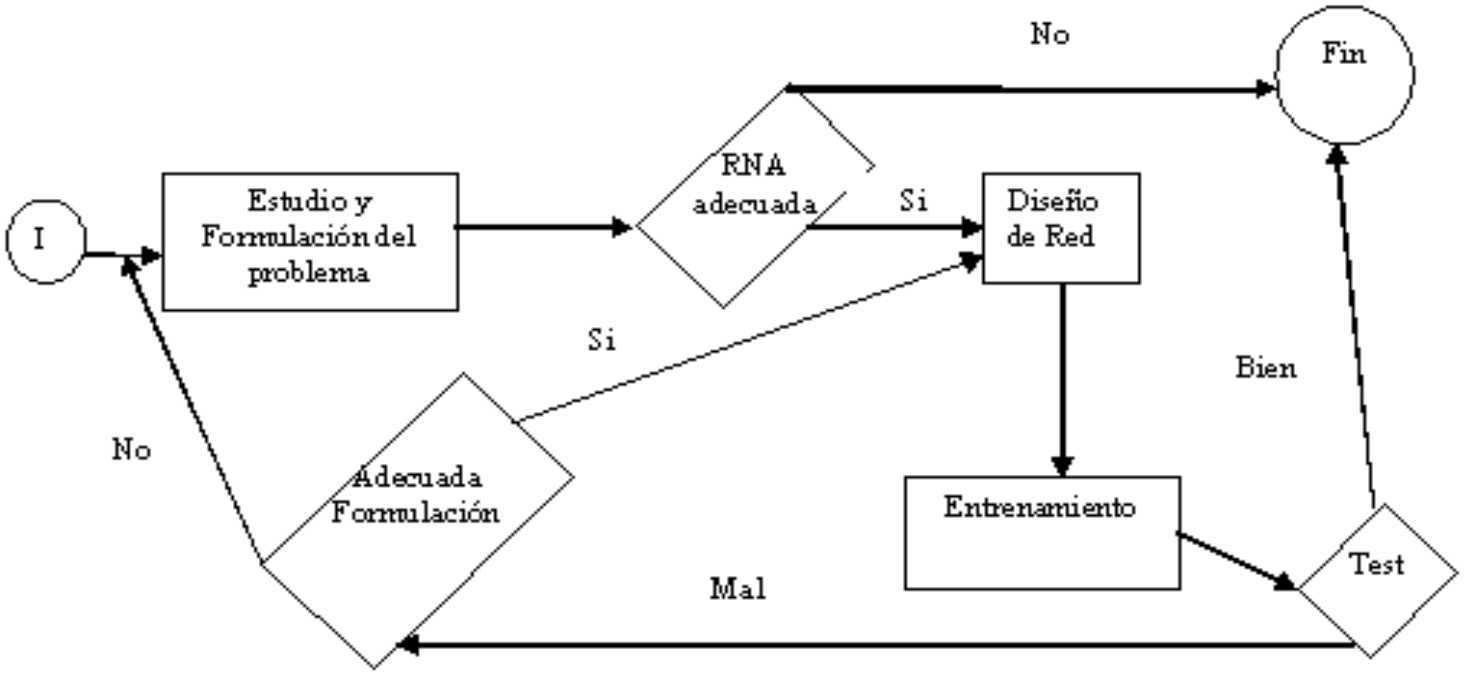
- Biología (por ejemplo, predicción de estructura de la proteína, análisis de datos genómicos)
- Química (por ejemplo, la predicción de las reacciones químicas)
- Física (por ejemplo, la detección de partículas exóticas)

y muchos más

Cómo aplicar una RNA a un problema?

- ✓ Análisis y definición del problema
- ✓ Análisis y procesamiento de los datos
- ✓ Definir el modelo de la RNA mas adecuado al problema
- ✓ Diseñar la estructura y construir la red
- ✓ Selección del conjunto de entrenamiento
- ✓ Validación del modelo
- ✓ Uso de la red => incorporación a un software de toma de decisiones

Cómo aplicar una RNA a un problema?



WCCI 2016

IEEE World
Congress on Computational Intelligence (IEEE WCCI)
Vancouver, Canada, 24–29 July 2016.

- **Hierarchical Dynamic Neural Networks** for Cascade System Modeling with Application to Wastewater Treatment
- **Decentralized Adaptive Neural Network Sliding Mode Control** for Reconfigurable Manipulators with Data-based Modeling
- A Note on **Fractional-Order Non-Linear Controller:** Possible Neural Network Approach to Design
- **Robust Neural Decentralized Control** for a Quadrotor UAV
- Induction Motor Torque Control via **Discrete-Time Neural Sliding Mode**
- **System Identification Models and Using Neural Networks** for Ground Source Heat Pump with Ground Temperature Modeling
- **An Emotional Controller PLC** Implementation for an Industrial Fan System

Trending topics: Deep Learning (54), Extreme Learning Machines (30), Feed forwardNNs and Supervised Learning (24),Online and Incremental Learning (24), Spiking Neural Networks (18), Unsupervised Learning and Clustering (18), ADP and Reinforcement Learning (18),

The main topics: Feature Selection and Extraction (18), Data Mining and Knowledge Discovery (12), Data Analysis and Pattern Recognition (12), Time Series Analysis and Forecasting (12),ML for Computer Vision (12), Power Systems Applications (12), Biomedical and Bioinformatics Applications (12), Applications to Complicated Data Environment (12), and Robotics (12).