Aprendizaje Conexionista: Redes Neuronales Artificiales y otras técnicas

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?

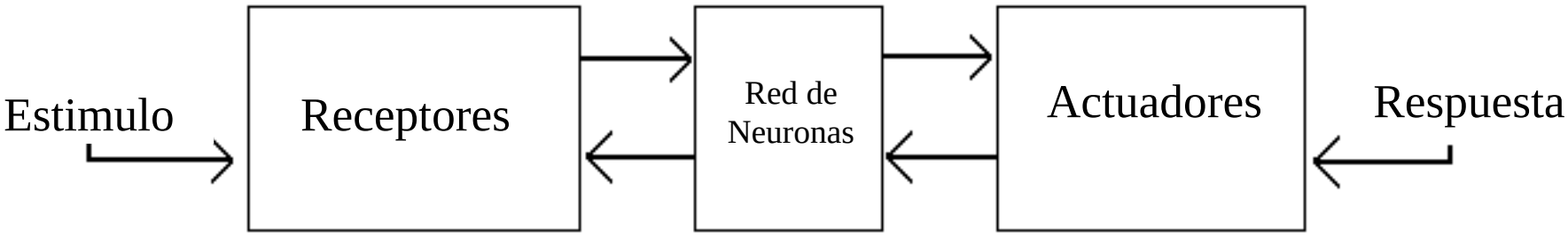
¿CÓMO PUEDE EL CEREBRO IMAGINAR?

¿CÓMO PUEDE EL CEREBRO INVENTAR?

¿QUÉ ES PENSAR?

¿QUÉ ES SENTIR?

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

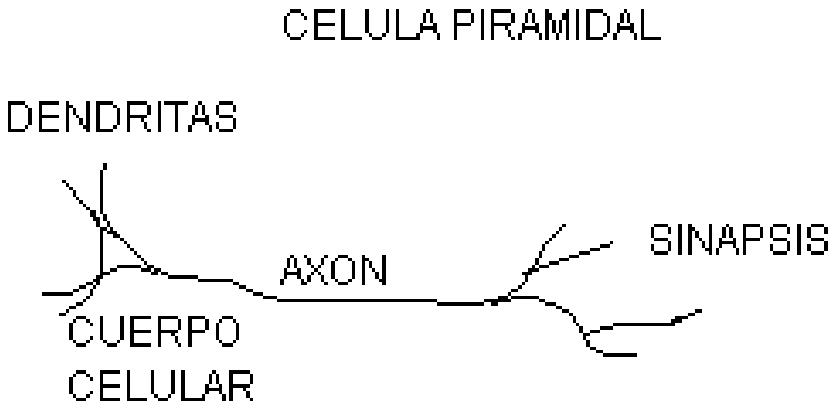
NEURONA

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

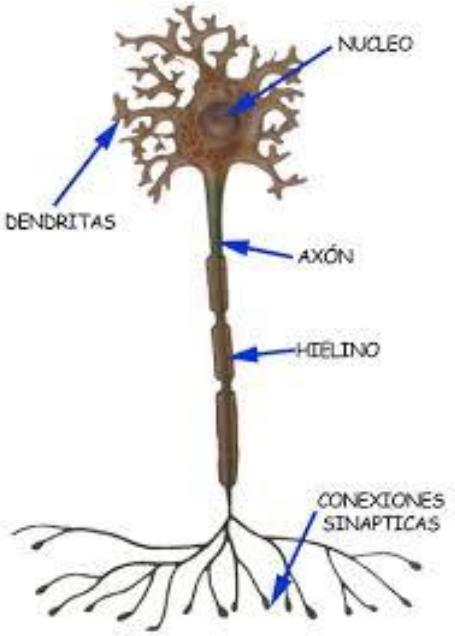
- **PROCESADOR DE SEÑALES ELÉCTRICAS** (DESCARGAS EN EL CUERPO CELULAR) **Y BIOQUÍMICAS** (NEUROTRANSMISORES)

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

NEURONA



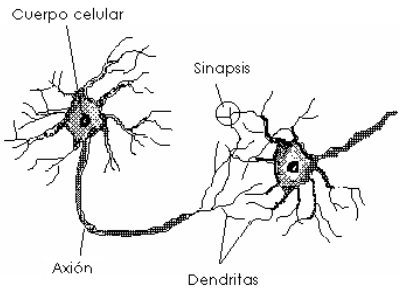
- **AXÓN:** LINEA DE TRANSMISIÓN
- **DENDRITAS:** ZONAS RECEPTORAS
- **SINAPSIS:** EXCITADORAS E INHIBIDORAS
- SEÑALES ELECTRICAS Y QUIMICAS



SINAPSIS

UNIDAD FUNCIONAL QUE INTERRELACIONA LAS NEURONAS

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



REDES NEURONALES

- MUCHAS **CONEXIONES PARALELAS** ENTRE NEURONAS
- MUCHAS CONEXIONES PROVEEN **MECANISMOS DE RETROALIMENTACIÓN** PARA LAS NEURONAS
- ALGUNAS NEURONAS PUEDEN **EXCITAR** UNAS NEURONAS MIENTRAS **INHIBEN** A OTRAS

REDES NEURONALES

- 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

HISTORIA REDES NEURONALES ARTIFICIALES

- El primer modelo matemático sobre el funcionamiento del cerebro es del años **1940**. **McCulloch y Pitts**.

- **Hebb** probó en **1949** que la conectividad del cerebro esta enteramente en evolución de la misma manera que un organismo aprende. Mas adelante, Hebb postuló que la activación repetida de una neurona por otra, a través de una sinapsis en particular, aumenta su conductancia.

HISTORIA REDES NEURONALES ARTIFICIALES

- Década 50, los investigadores se confrontaron con el problema de **consistencias del modelo** al no poder explicar el funcionamiento del cerebro a partir de este.

- Primer modelo sólido fue presentado por **Rosenblatt** en **1958** (Perceptrón).

- 1era RN Comercial (ADALINE) propuesta por **Widrow** y **Hoff** en **1959**

HISTORIA REDES NEURONALES ARTIFICIALES

- En **1969 Minsky-Papert** => determinan problemas de aprendizaje del PERCEPTRÓN

- En los años **80**, los modelos propuestos por **Hopfield, Kohonen, Grossberg** y otros han ayudado al avance en esta área.

- **Cohen y Grossberg** desarrollaron un modelo basado en la idea de plasticidad

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

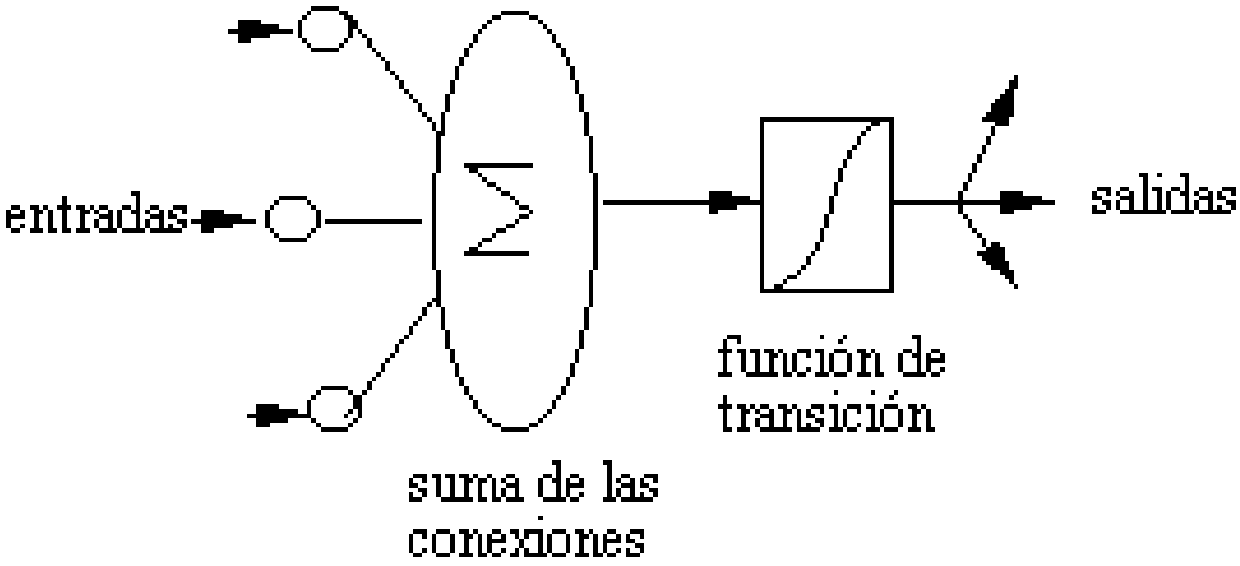
COMPARACION RED NEURONAL

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}

COMPARACION RED NEURONAL

SISTEMAS EXPERTOS	RNA
Enfoque descendente	Enfoque Ascendente
Basado en la psicología	Basado en la biología
Qué hace el cerebro	Cómo lo hace el cerebro
Reglas Si/Entonces Sentencias	Generalización a partir de ejemplo
Sistemas Programados	Sistemas Entrenados
Lógicas, reglas	Reconocimiento de Patrones
Maquina Von Newman	Computación Paralela/Distribuida

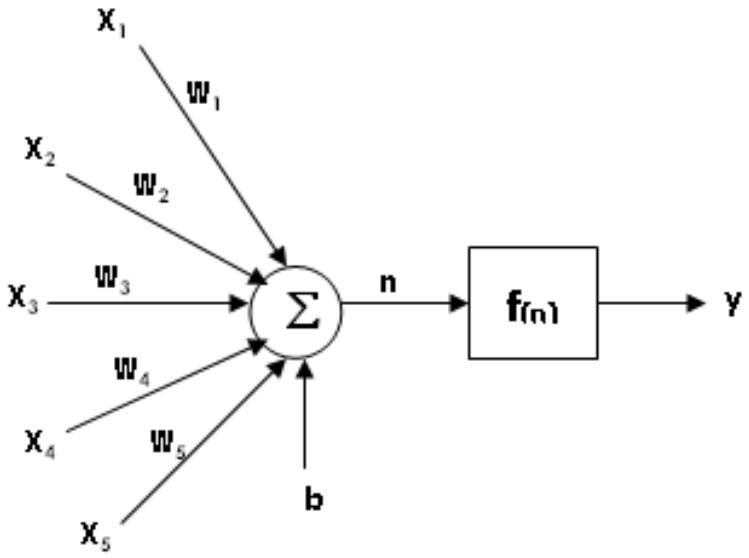
COMO TRABAJA UNA NEURONA ARTIFICIAL



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**

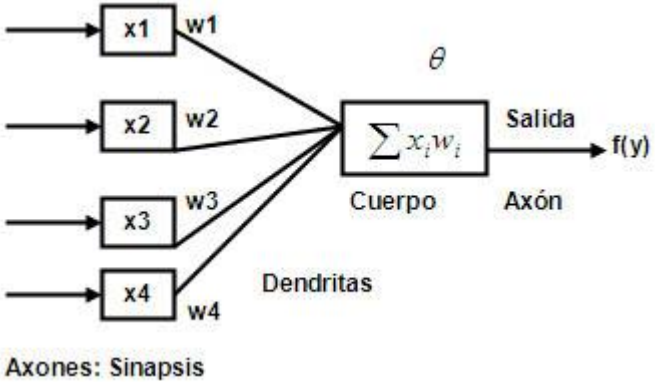
El **nodo sumatorio** acumula todas las señales de entrada multiplicadas por los pesos y las pasa a la salida a través de una **función de activación** o transferencia f(n), (b es el sesgo).

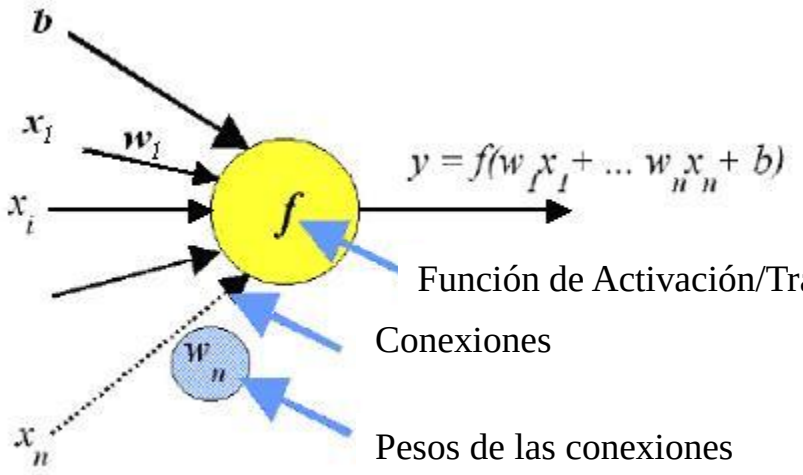


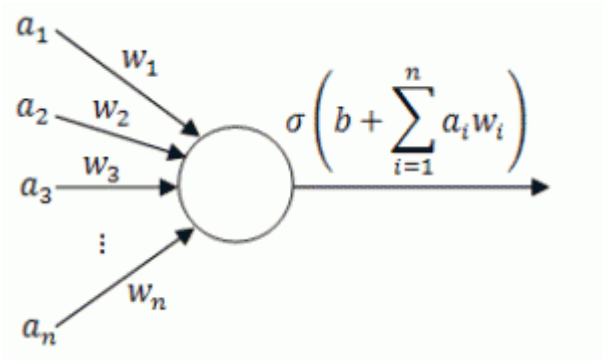
COMO TRABAJA UNA NEURONA ARTIFICIAL

- **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.

COMO TRABAJA UNA NEURONA ARTIFICIAL







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**

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

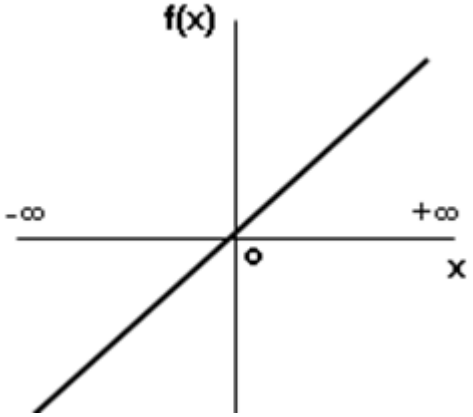
7. **La función de transición o de salida**

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

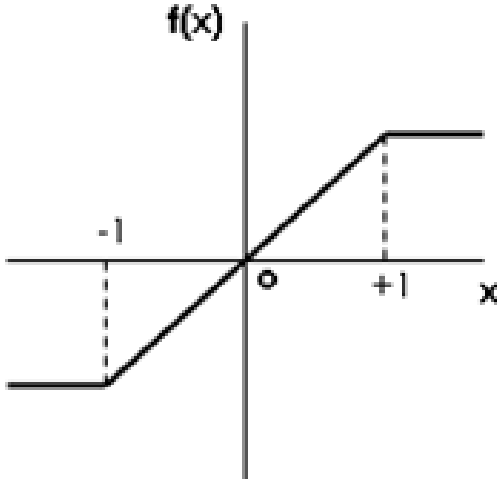
Función de activación

Función identidad o función lineal:

$$f(x) = x$$



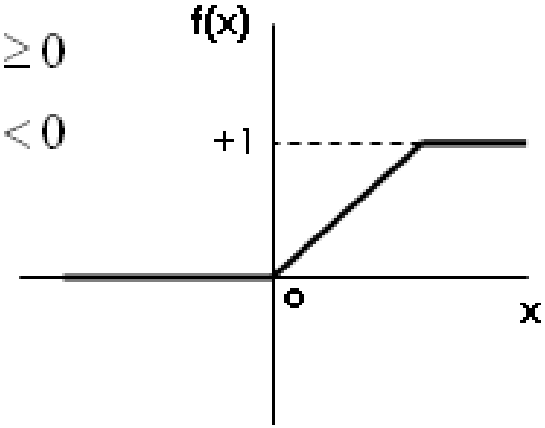
Función lineal por tramos



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

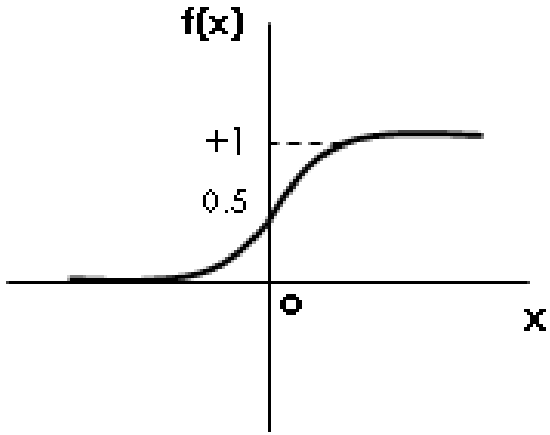
Función escalón

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



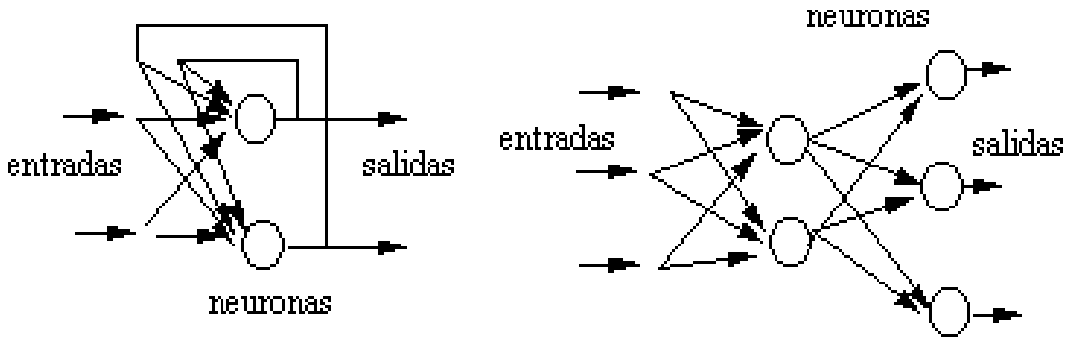
Función sigmoidal

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



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

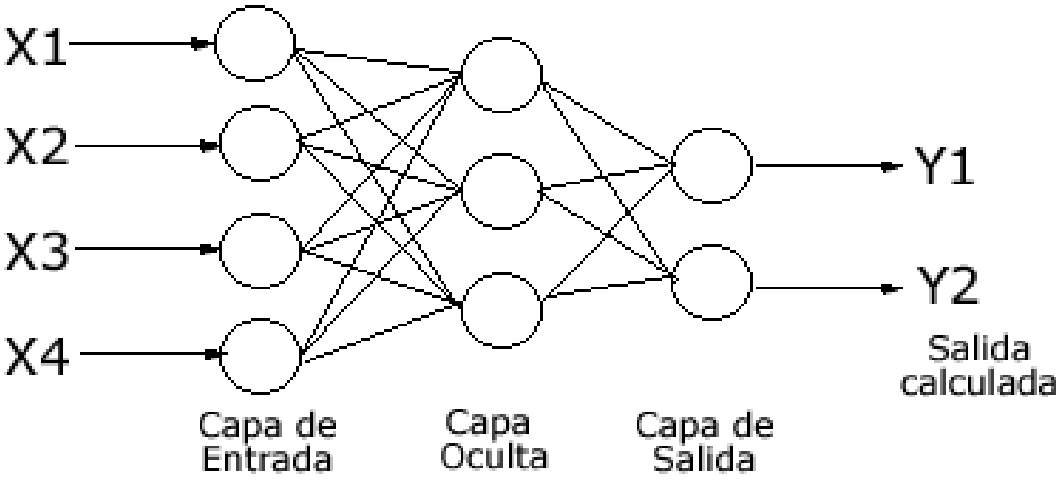
Propagación hacia atrás

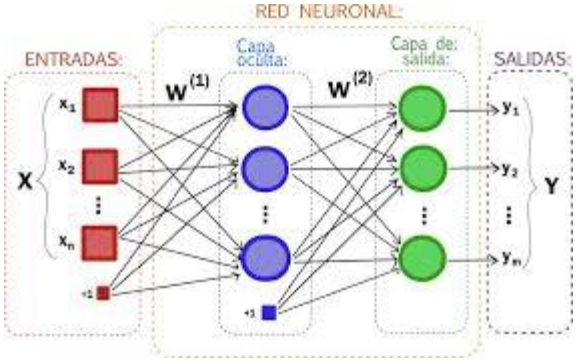
Redes recurrentes

Redes de alimentación lateral

Redes Multicapas

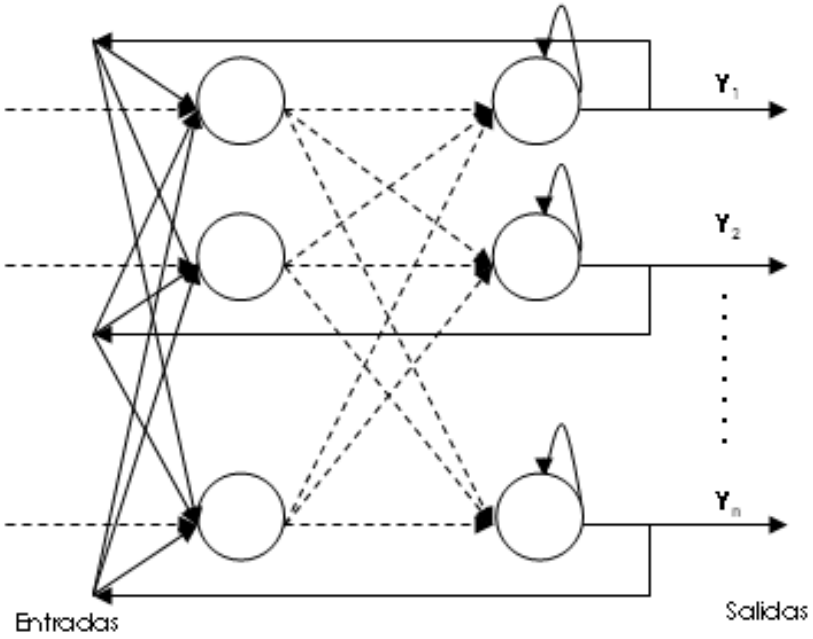
- **Capa de Entrada:** está constituida por los nodos de entrada, que reciben directamente la información de las fuentes externas a la red.
- **Capas Ocultas:** no tienen contacto con el exterior ya que se encuentran ubicadas entre la capa de entrada y la capa de salida. La cantidad de capas ocultas 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.



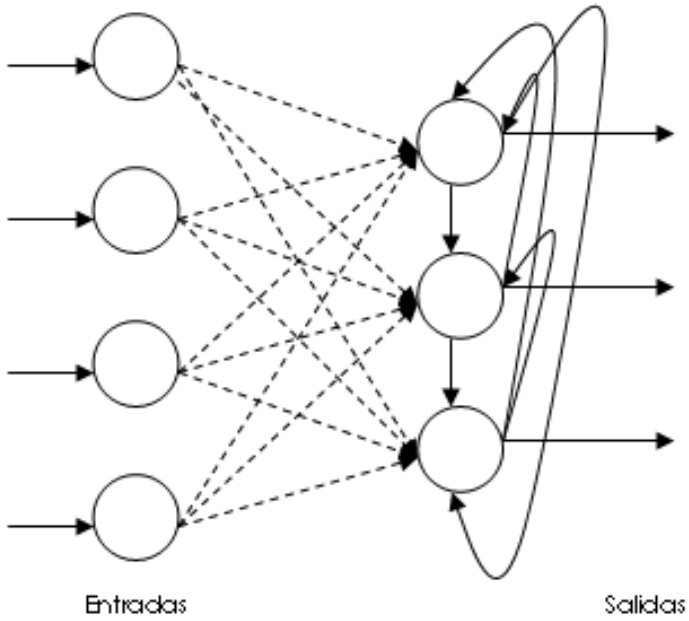


Redes Multicapas

Redes recurrentes



Redes de alimentación lateral



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.

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.

- No existe en la literatura una metodología que indique la **manera de escoger el tipo o forma de aprendizaje** de la red para obtener resultados óptimos.

- Tipo de aprendizaje viene determinado por la **forma en que los parámetros** se deben adaptar

MEMORIAS ASOCIATIVAS

- RN **ALMACENAN INFORMACIÓN APRENDIDA** REFLEJADA EN SUS PESOS

- AL APLICARLE UNA **ENTRADA LA RNA RESPONDE CON UNA SALIDA ASOCIADA** A DICHA INFORMACIÓN DE ENTRADA

ASOCIACIÓN ENTRADA/SALIDA

APRENDIZAJE

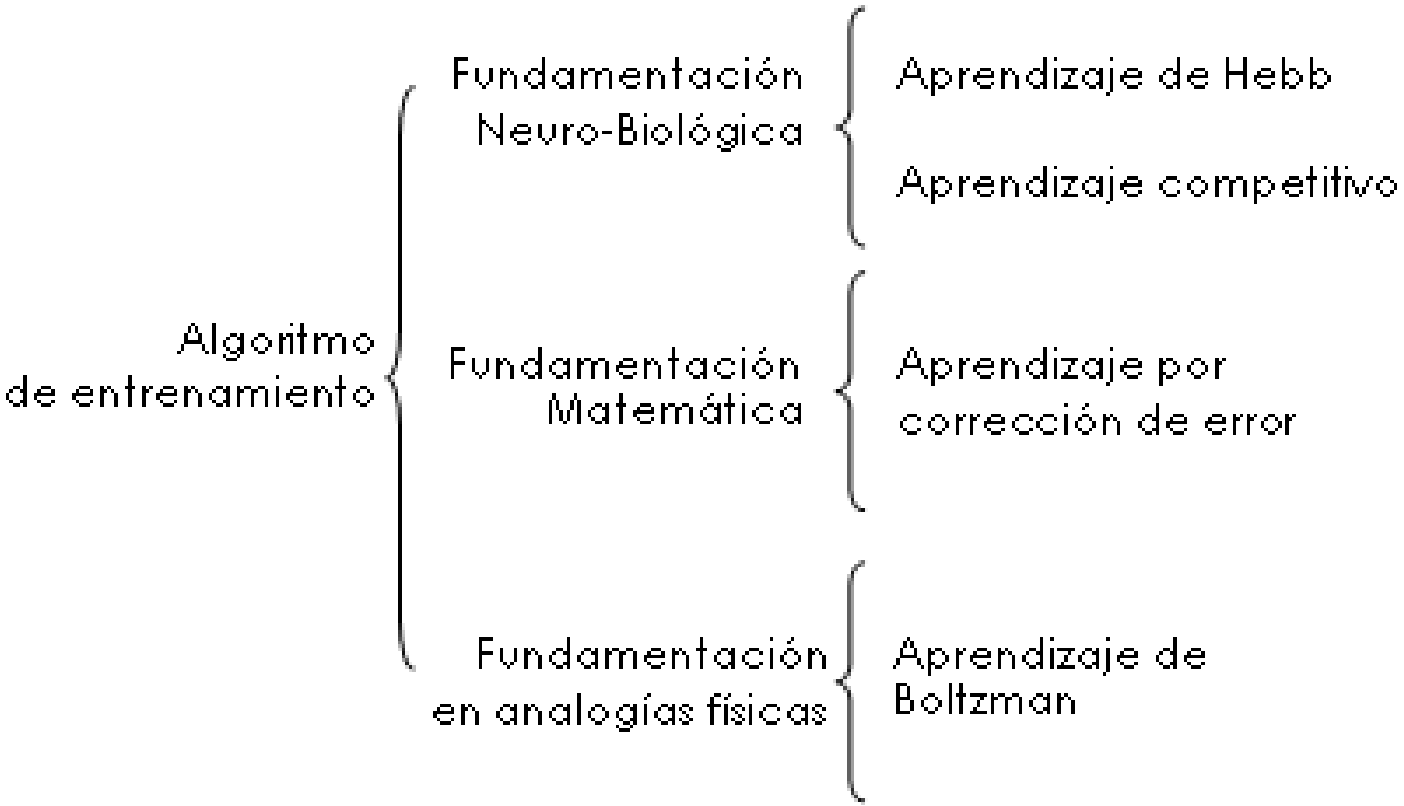
MODIFICAR PESOS DE LAS CONEXIONES DE LAS NEURONAS (CREAR, DESTRUIR, MODIFICAR)

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

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

- En **un agente** se pueden usar todas
- En el caso de **múltiples agentes**, los métodos supervisados no son fáciles de aplicar

  Mas usado los métodos de recompensa.

- Aprendizaje basado en recompensas puede ser dividido en dos subconjuntos:
  - **Métodos de aprendizaje por refuerzo**: estiman funciones de valor
  - **Métodos estocásticos** **,** tales como la computación evolutiva, recocido simulado.

APRENDIZAJE

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

CORRECCIÓN DE ERROR
BOLTZMAN
HEBBIANO
COMPETITIVO
EVOLUTIVO

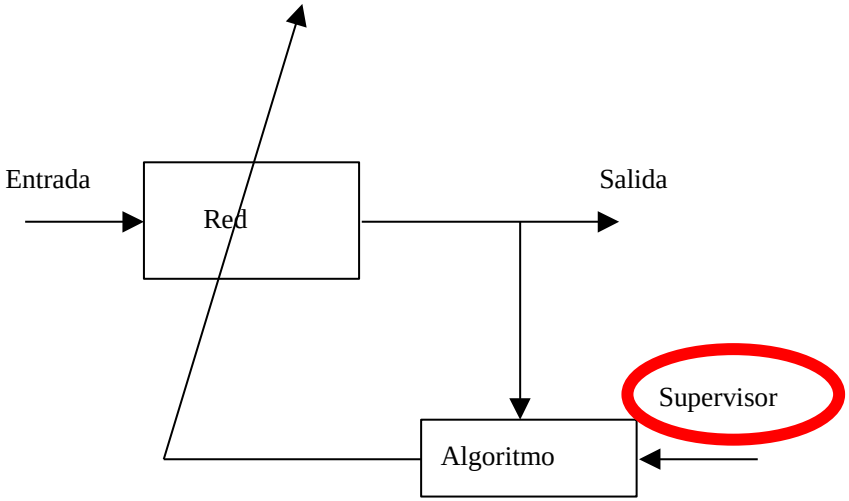
SUPERVISADO

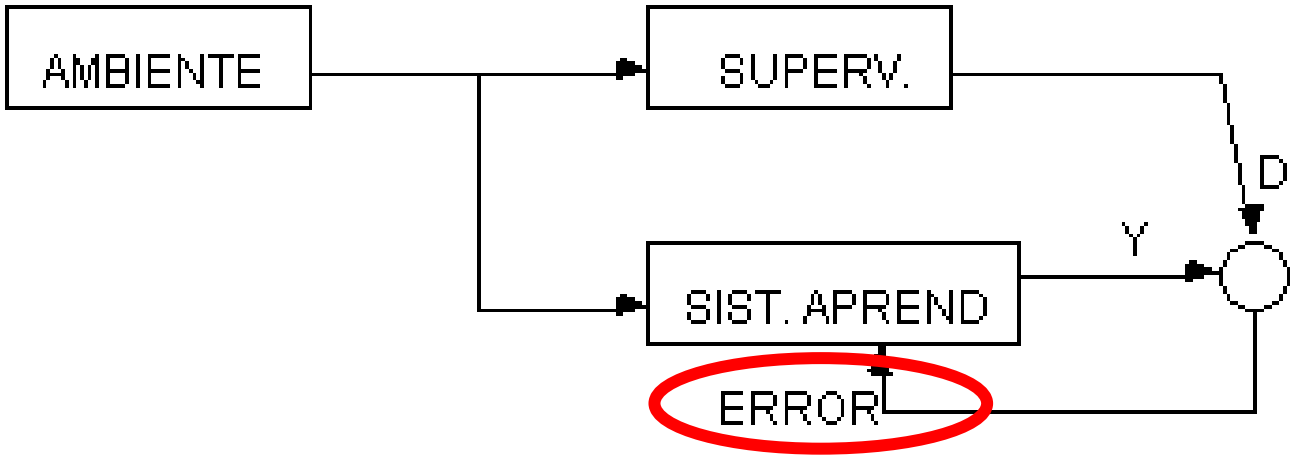
Respuesta correcta para cada ejemplo es dada

- SE DAN DATOS DE ENTRADA Y SALIDA OBJETIVO

- SALIDA RED DEBE CONCORDAR CON LA DESEADA

SUPERVISADO





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

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

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

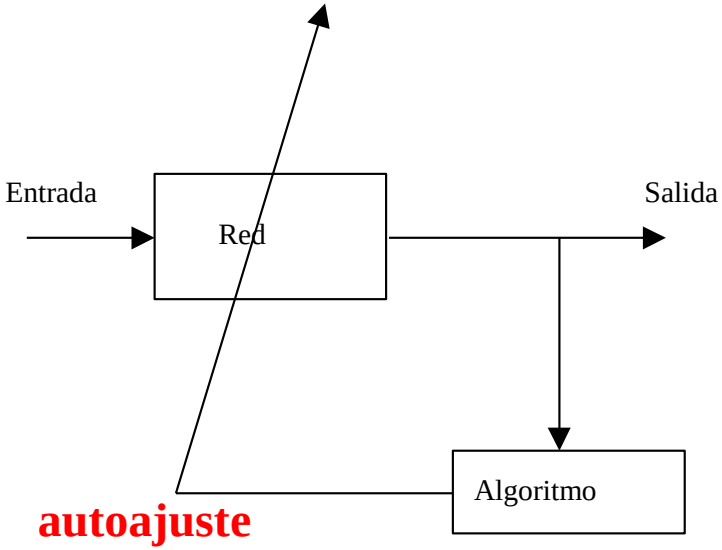
NO SUPERVISADO (AUTOORGANIZADO)

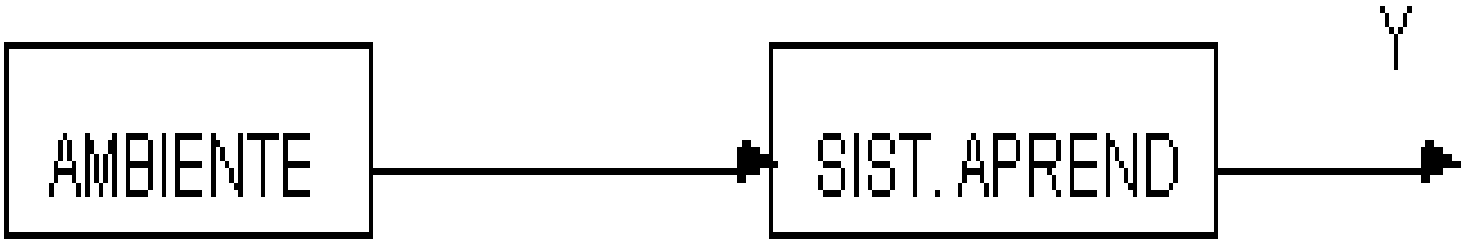
- NO RECIBE INFORMACIÓN DE SU ENTORNO (Se reciben patrones sin la respuesta deseada)

- 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

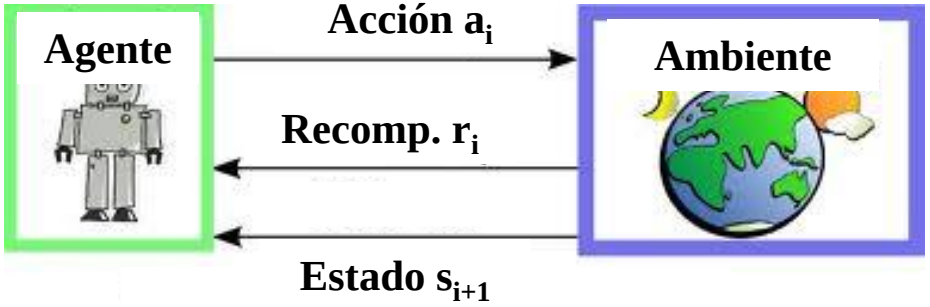
=> REFORZAR LA CONEXIÓN ENTRE ELLAS

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

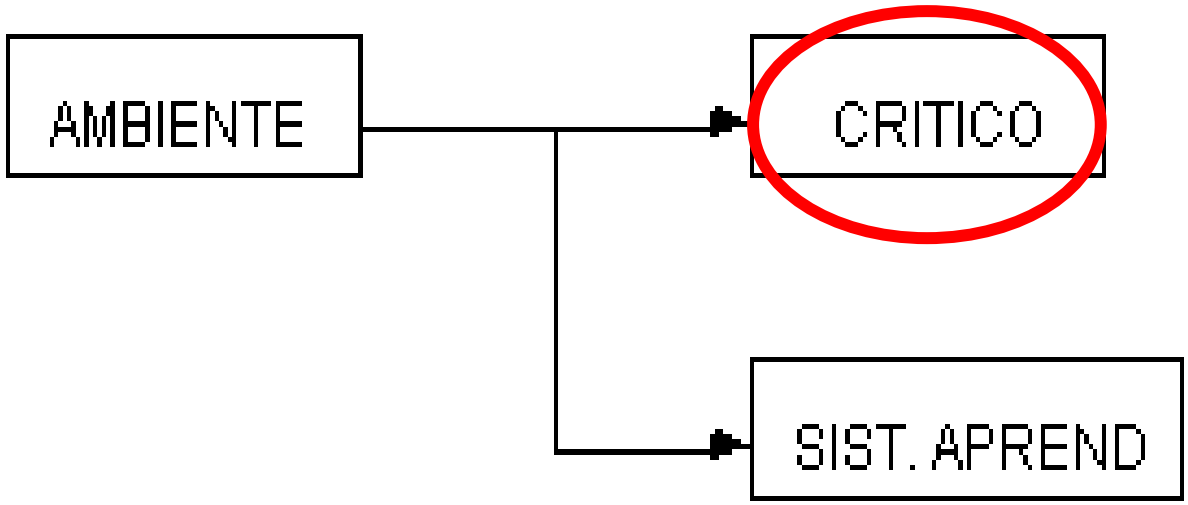
REFORZADO

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



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
  - **Agente reflexivo** aprende mapeo estado-acción

Ejemplos iniciales de Reforzamiento

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

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

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 agente 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 de la **función de utilidad** que determina comportamiento del agente

- Programación Dinámica Adaptativa

$$U(i)=R(i) + \sum_j M(i,j)U(j)$$

M(i, j): prob. de transición del estado i al j

R(i): premio por estar en estado i

Sistema observable!!

- Diferencia Temporal

$$U(i)=U(i)+\alpha(R(i) +[U(i')-U(i)])$$

Transición entre estado i e i'

Diferencia en la utilidad entre sucesivos estados

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

- Añadir Exploración sobre el futuro:
  - Ganar premio en la secuencia actual
  - Capacidad para percibir el futuro pensando en ganar premio en ese momento

$$U(i)= R(i)+Max_a\ f(\sum j\ M_a(i, j)U(j), N(a,i))$$

f: función de exploración

N: número de veces que *a* ha sido intentado en el estado i

Aprendizaje Reforzado

Algoritmo Q-Learning automático

- 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

Modelos Neuronales

Clasificación por tipo de aprendizaje y arquitectura

Híbridos: RBF (RADIAL BASIC FUNCTION)

Supervisados

Realimentados : feed-propagation

Unidireccionales PERCEPTRON, M RN, BOLTZMAN, backpropagation

No supervisados

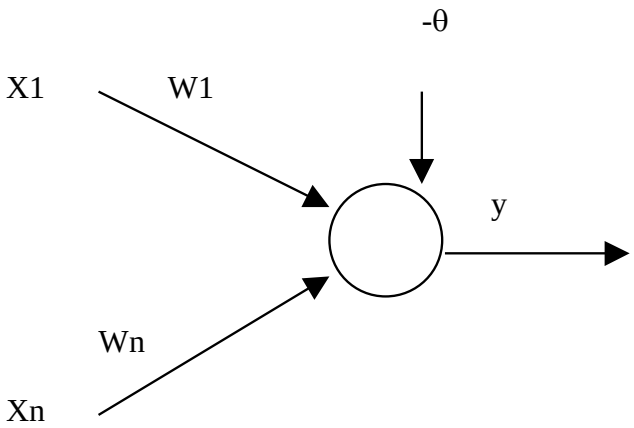
Realimentados: ART, HOPFIELD

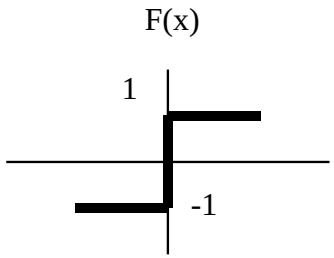
Unidireccionales: KOHONEN

Reforzados

PERCEPTRÓN

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





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

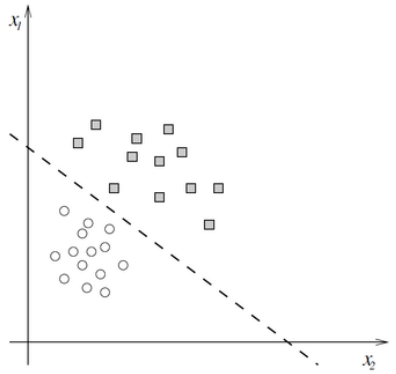
PERCEPTRÓN

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

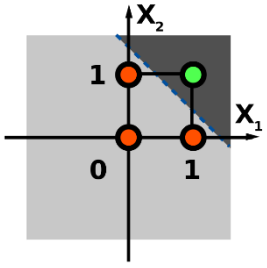
  => PATRONES SEPARABLES GEOMÉTRICAMENTE

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

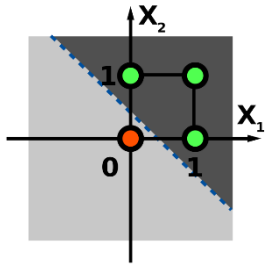
  => TRES ENTRADAS PLANO



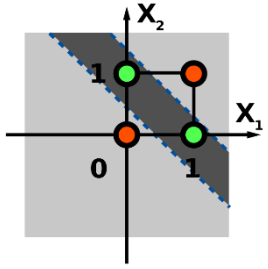
- LAS FUNCIONES **AND** Y **OR** SON LINEALMENTE SEPARABLES, POR LO TANTO LAS PUEDE APRENDER. EL **XOR** NO LO PUEDE APRENDER



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)

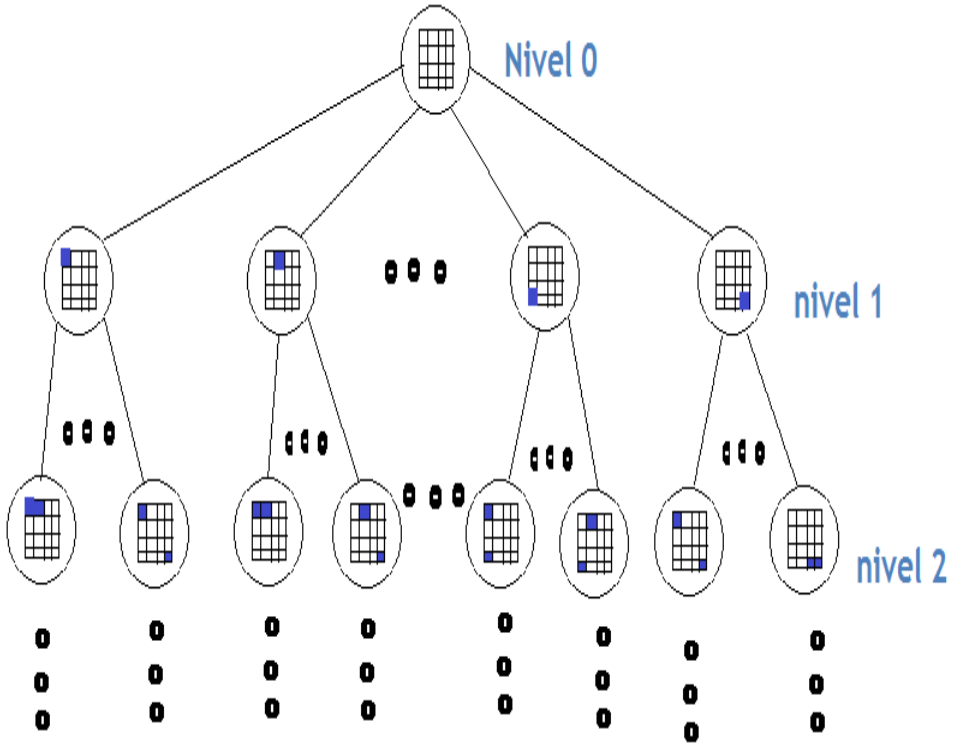
4. ADAPTAR LOS PESOS
   $W_i(t)=W_i(t)+\alpha[d(t)-Y(t)]X_i(t)$

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

5. REGRESAR AL PASO 2

Aprendizaje bayesiano

Modelo Matemático de Manejo de Incertidumbre



Red bayesiana para el manejo de incertidumbre

- $MUE = \max(U(a_i) * \sum_{i=0}^{n} P_i(^{a_i}/_{P_i}))$

Caso juego: Según la función MUE la mejor acción será aquella en la cual la razón dada entre la utilidad y la probabilidad de que el oponente obtenga una mala jugada sea máxima.

Modelo Matemático de Aprendizaje

Parametric Learning

Dada la estructura, se deben obtener las probabilidades asociadas.

- *Learning by method of maximizing the expectation*
- *Counting-Learning*

Learning by method of maximizing the expectation

Es muy utilizado en el caso de nodos ocultos. Los nodos ocultos se definen como los nodos donde s**e desconocen los valores que una variable puede tener**. El algoritmo de este método es

- Start the unknown parameters (conditional p probabilities) with random values (or estimated by experts). $\theta^{[t]} = \{\theta_1^{[t]}, \theta_2^{[t]}, \ldots, \theta_c^{[t]}\}$
- Use the known data with the current parameters $\theta_c^{[t]}$ to estimate the values of the unknown variables, where t is the iteration index, f is the auxiliary function, $x = (x_1, x_2, \ldots, x_n$ are the observed data, $z = (z_1, z_2, \ldots, z_n)$ are the unknown variables, and q is a distribution on the variables Z

$$\sum_z q(z|x) \log \frac{p(x,z|\theta)}{q(z|x)} \triangleq \mathcal{F}(q, \theta) \qquad q_c^{[t+1]} \leftarrow \underset{q}{\operatorname{argmax}}\, \mathcal{F}(q, \theta_c^{[t]})$$

- Use the values estimated in the previous step q(t+1) to complete the data table (also called STEP M by maximizing). $\theta_c^{[t+1]} \leftarrow \underset{\theta}{\operatorname{argmax}}\, \mathcal{F}(q_c^{[t+1]}, \theta_c)$
- Repeat until there are no significant changes in the odds.

Counting-Learning

es el más simple y más rápido método, cuando no hay muchos datos que falten o muchos datos inciertos. El método de recuento se puede usar para **aprender o reforzar las tablas de probabilidad condicional** (CPT).

- Red que se encuentra en un estado de total ignorancia, y todas las probabilidades son uniformes.
- Permite aprender o reforzar las probabilidades. En ambos casos:
- Para los nodos de la red en la que hay un nuevo hallazgo, se debe modificar la relación con sus padres (por ejemplo, entre un nodo dado (Ix) y sus padres (PX> z), donde z es el número de padres del nodo), tal que su experiencia y probabilidad individual se modifican (Ix) (la probabilidad condicional de sus padres a él se actualiza).

$$experiencia_{I_x}^{n} = experiencia_{I_x}^{n-1} + creencia$$

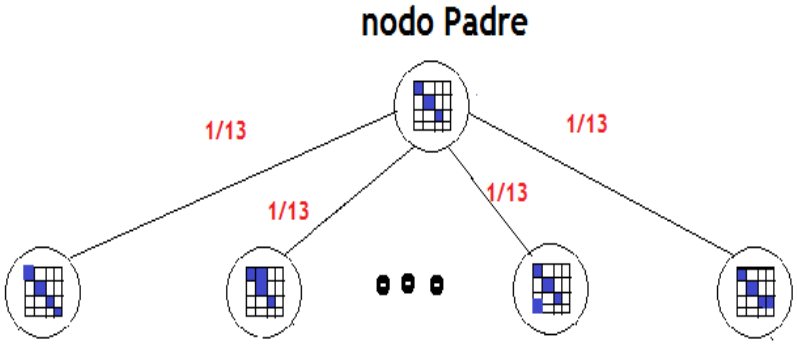
$$probabilidad_{I_x}^{n} = \frac{(probabilidad_{I_x}^{n-1} * experiencia_{I_x}^{n-1} + creencia)}{experiencia_{I_x}^{n}}$$

Las probabilidades de los hermanos de Ix también deben cambiar

$$probabilidad_{i}^{n} = \frac{probabilidad_{i}^{n-1} * experiencia_{I_x}^{n-1}}{experiencia_{I_x}^{n}}$$

Modelo Matemático de Aprendizaje

Se tiene el siguiente Árbol con 13 nodos



Red bayesiana en su estado de máxima confusión

Según acción del adversario sea buena o no, la rama debe ser premiada (o penalizada) y las del resto de hermanos inversamente modificadas (aprendizaje reforzado)

Para actualizar las ramas se pueden usar los siguientes valores:

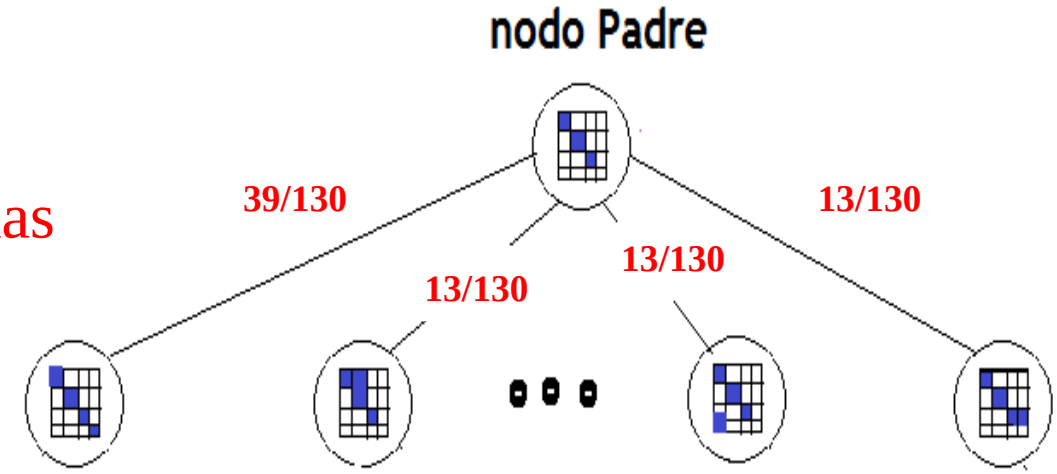
- $\mathbf{p_{obj}}$= 6/10 se suma (resta) a la rama evaluada para premiar (castigar)
- $\mathbf{p_{resto}}$=2/10 se resta (suma) al resto de ramas para penalizar (premiar)

Modelo Matemático de Aprendizaje

$$p(x_i)' = \frac{p(x_i)}{\sum_{i=1}^{n} p(x_i)}$$

Después, suponiendo que la probabilidad de cada rama es $p(x_i)$, se normalizan los niveles

red bayesiana con probabilidades actualizadas

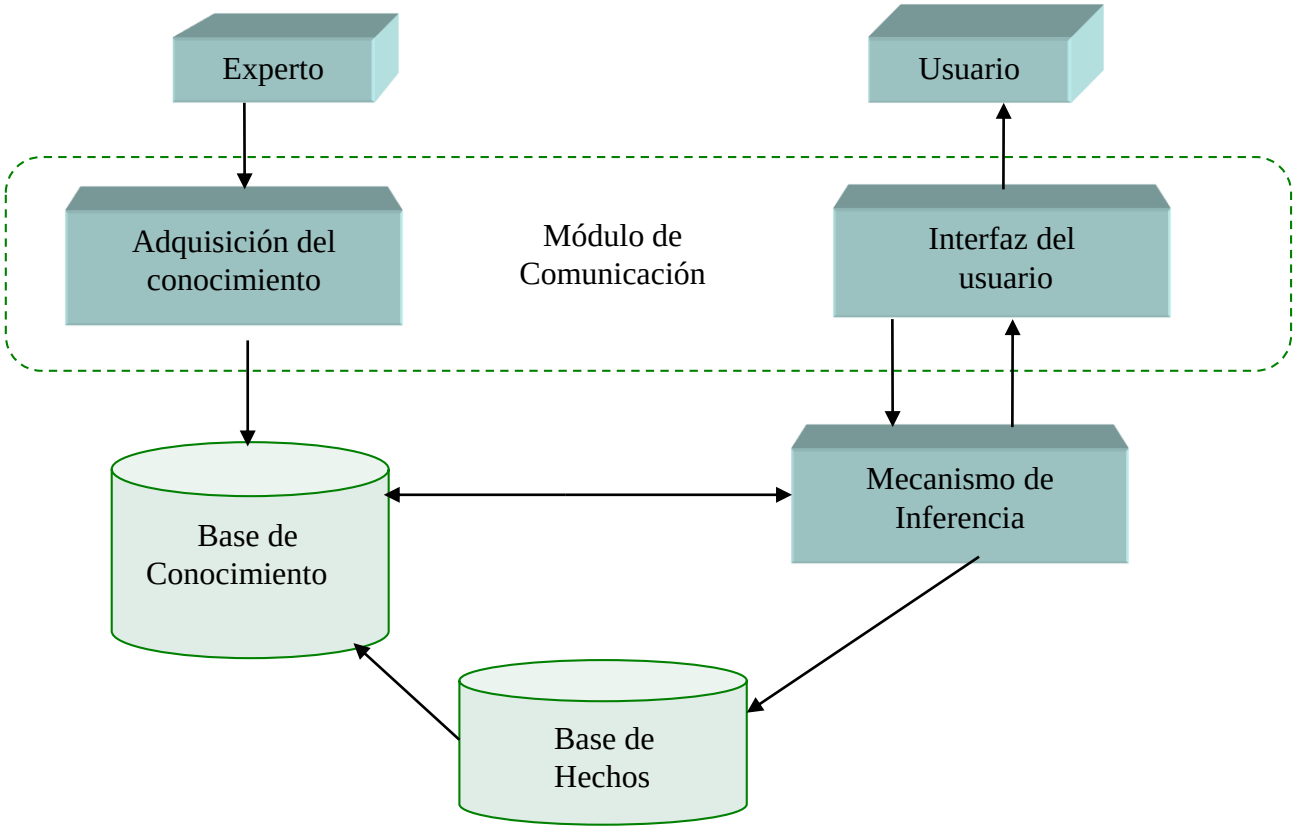


Modelo Matemático de Aprendizaje

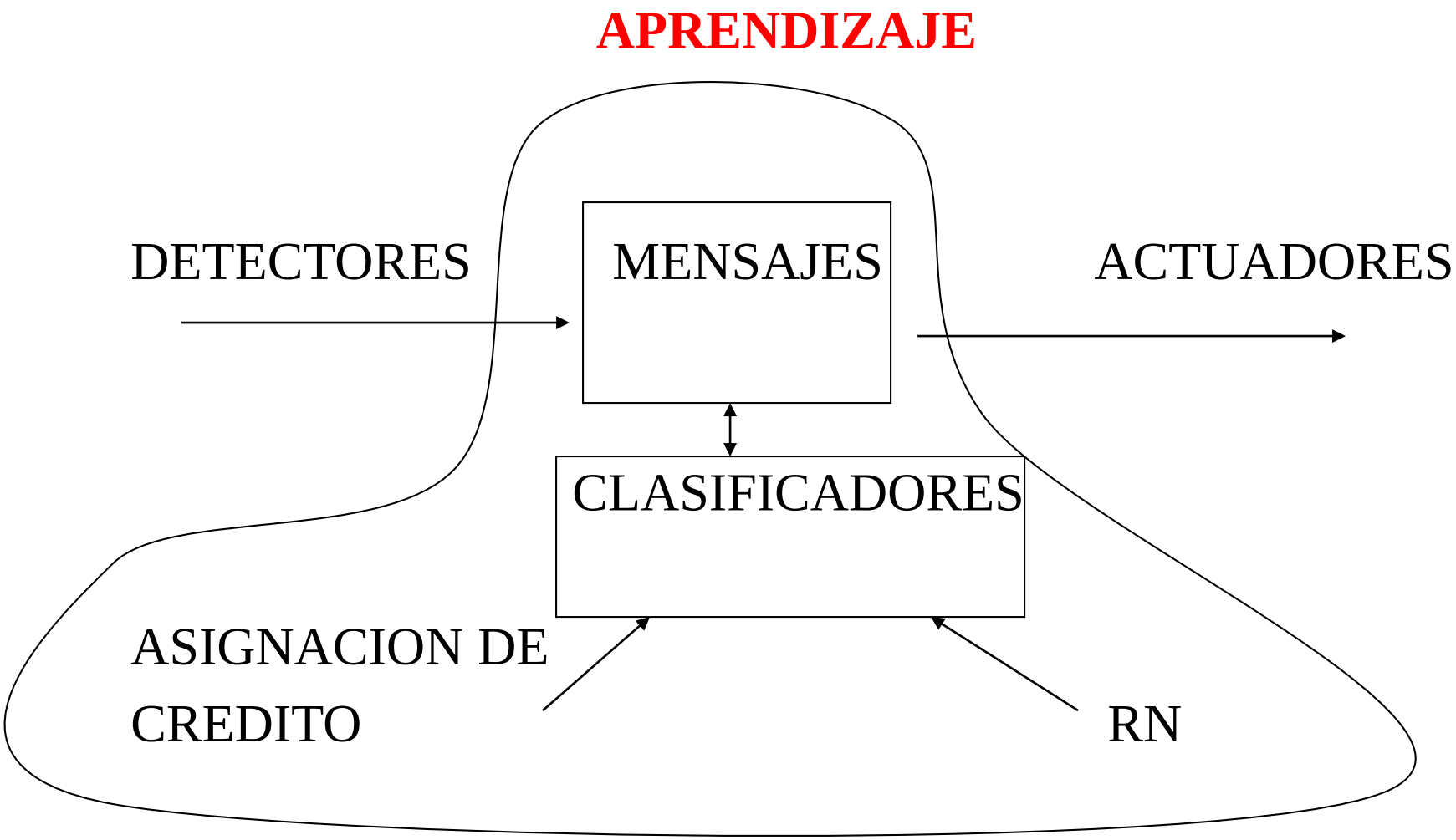
Structural Learning

- consiste en obtener la estructura o topología de la red (DAG), es decir, se encuentran las **relaciones de dependencia entre las diferentes variables**.
- Estas relaciones se pueden encontrar principalmente de dos maneras:
  - obtener la estructura de un experto,
  - usar minería de datos para obtener la estructura.

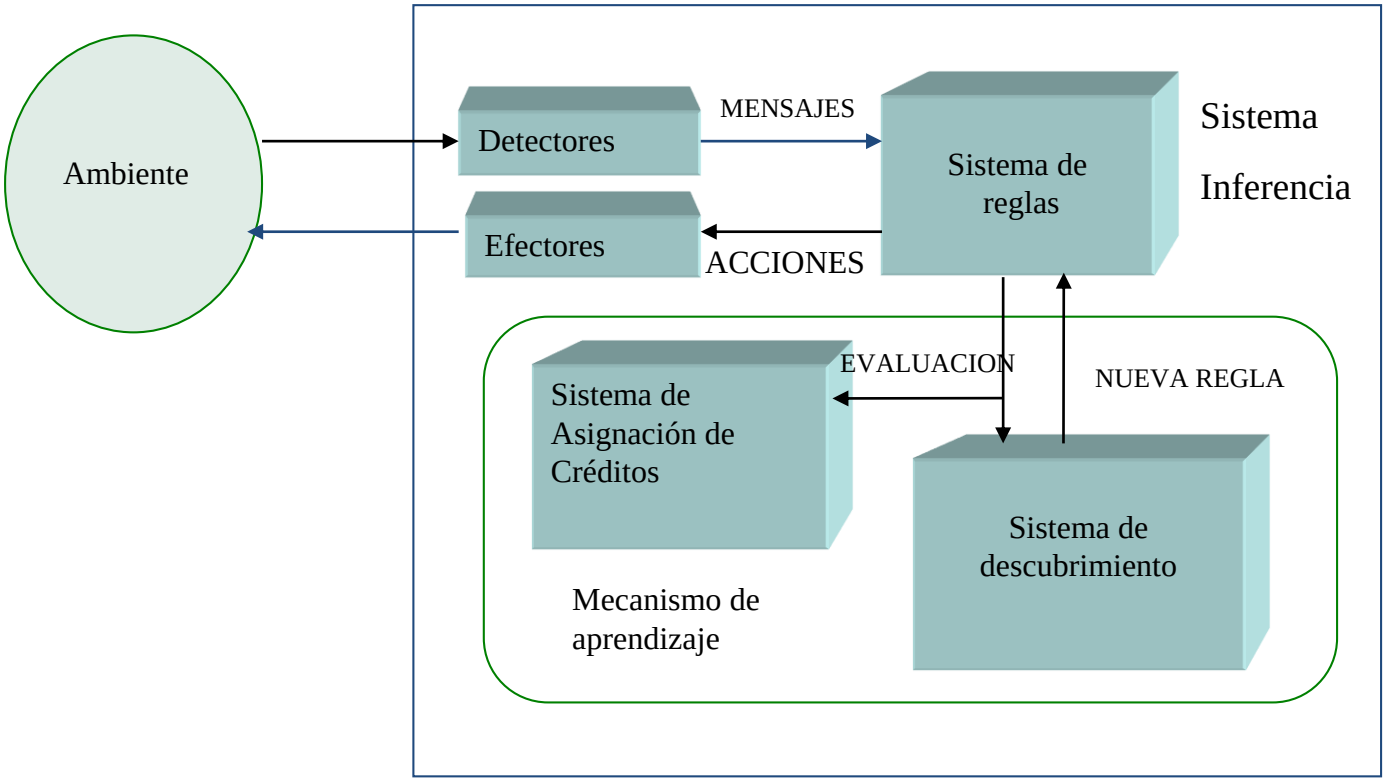
SISTEMAS EXPERTOS



Sistemas Clasificador



SISTEMAS CLASIFICADORES



REGLA i

$$S_i\ (t+1)= S_i(t)- P_i(t)+ R_i(t)+ Act_i(t)$$

Aprendizaje profundo

algoritmos en aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas compuestas de transformaciones no-lineales múltiples.

- Basados en una cascada de capas con unidades de procesamiento no lineal para extraer y transformar variables. Cada capa usa la salida de la capa anterior como entrada.
- 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

Los algoritmos de aprendizaje profundo contrastan con los otros algoritmos de aprendizaje por el **número de transformaciones aplicadas a la entrada** mientras se propaga desde la capa de entrada a la capa de salida.

Cada transformación con sus parámetros que se pueden entrenar (pesos, umbrales, etc.)

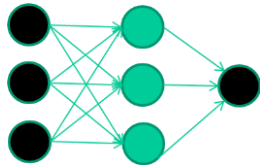
¿por qué es generalmente mejor?

- 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 nuestros cerebros parecen.

hmmm… OK, pero:

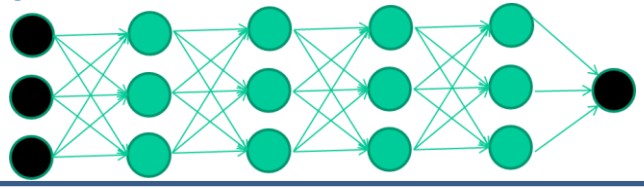
- Redes neuronales multicapas tienen muchísimos años.
- Qué es lo nuevo?

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

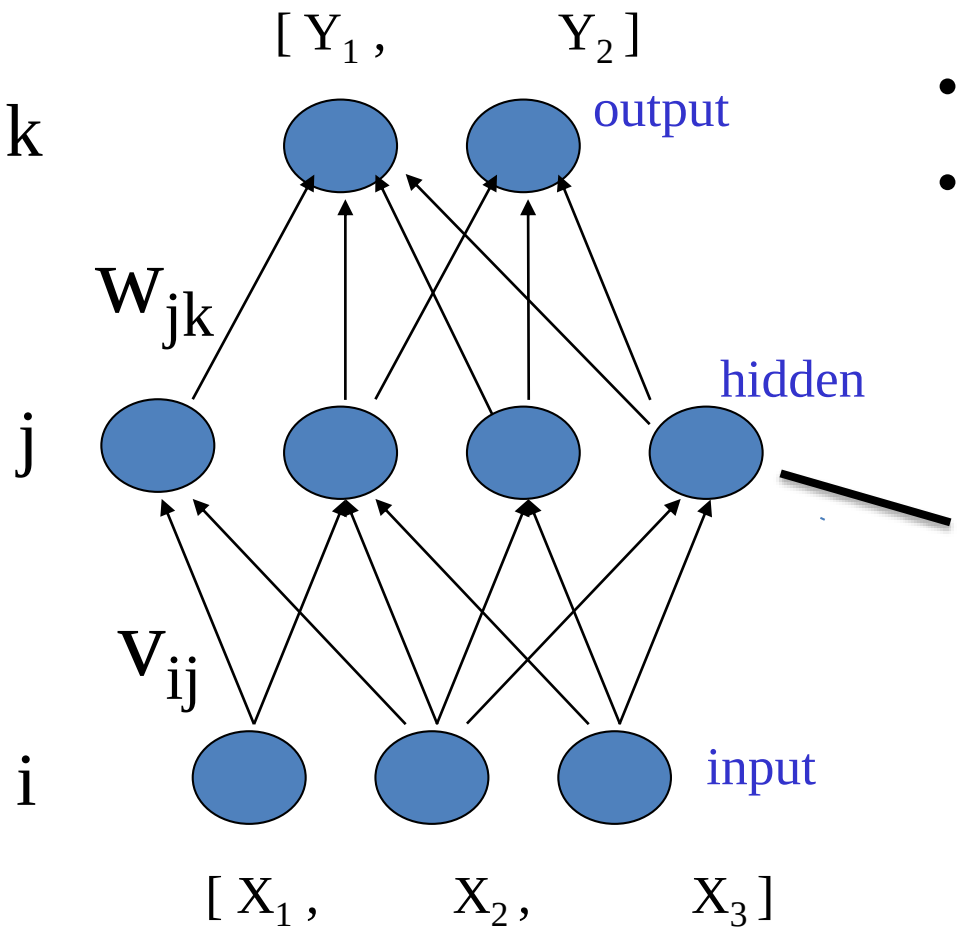


pero estos algoritmos no son buenos para aprender las ponderaciones de redes con más capas ocultas

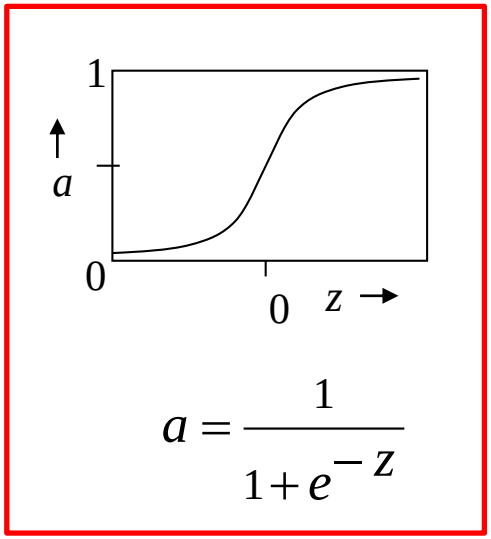
Lo nuevo es: algoritmos para entrenar redes "mucho-más tarde"



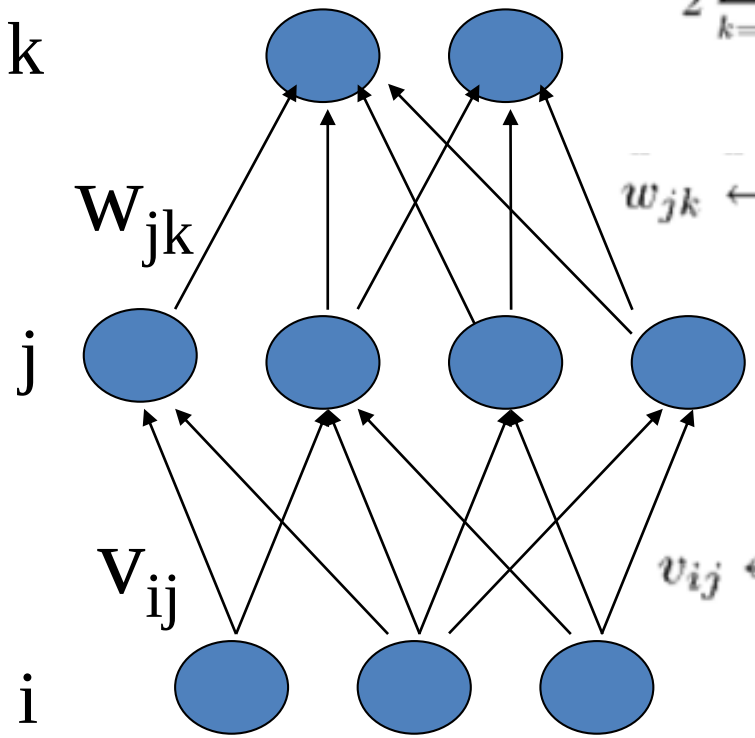
Perceptron Multicapa



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



Backpropagation



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

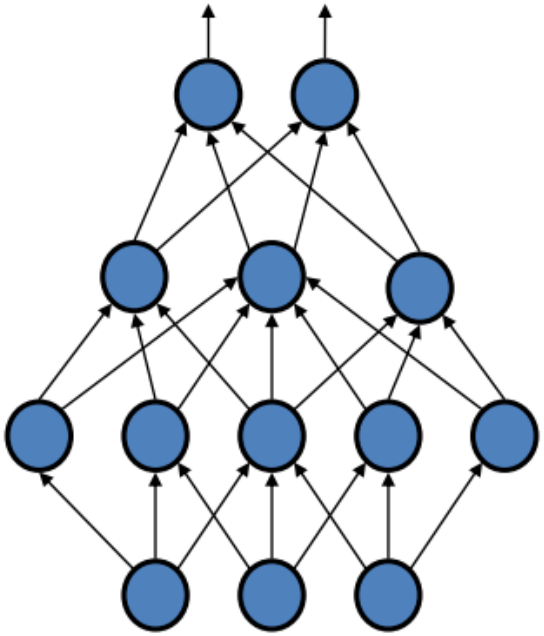
$$w_{jk} \leftarrow w_{jk} - \eta \frac{\partial E}{\partial w_{jk}}$$

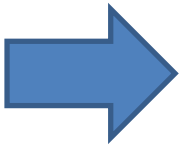
$$v_{ij} \leftarrow v_{ij} - \eta \frac{\partial E}{\partial v_{ij}}$$

- 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

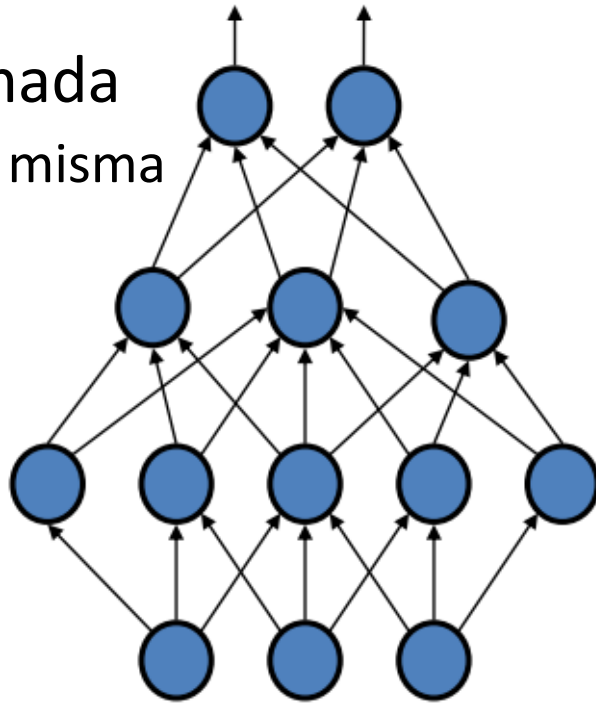




Otro enfoque

Arquitectura Deep

- Backpropagation, como construyendo bloques
- Multiples capas escondidas
- Motivación
  - Frontera de decisión compleja aproximada
    - Menos unidades computacionales para la misma tarea funcional
  - aprendizaje jerárquica
    - 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





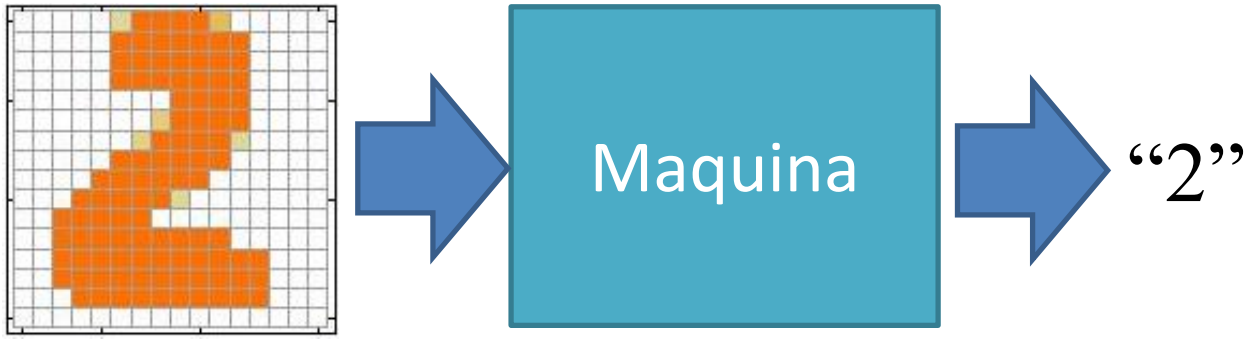

Imágenes sin etiquetas (p.e., tomadas de internet)

Carro

Motocicleta

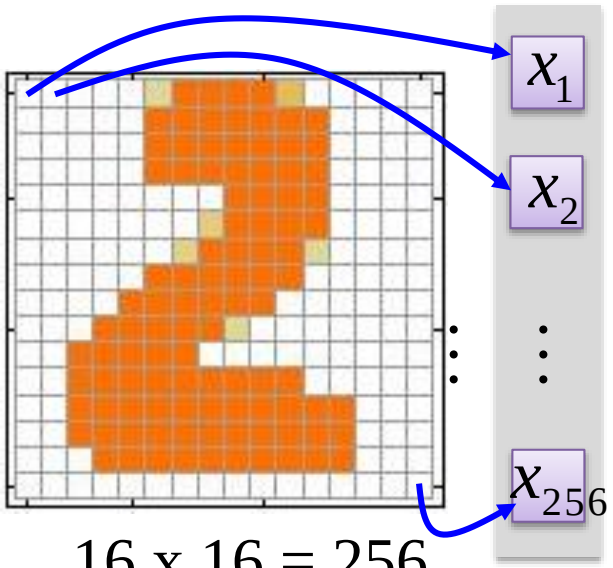
Ejemplo

- Reconocimiento de dígitos escritos



Reconocimiento de dígitos escritos

Entrada

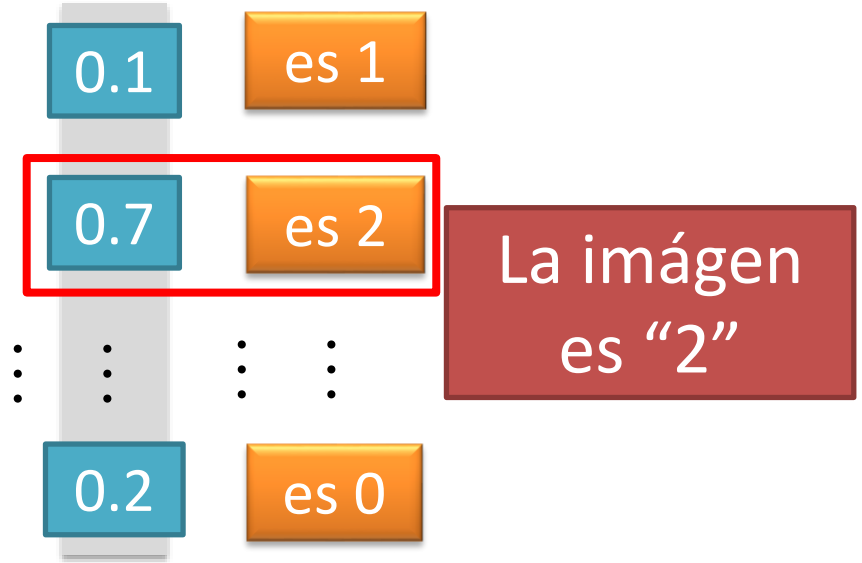


16 x 16 = 256

Ink → 1

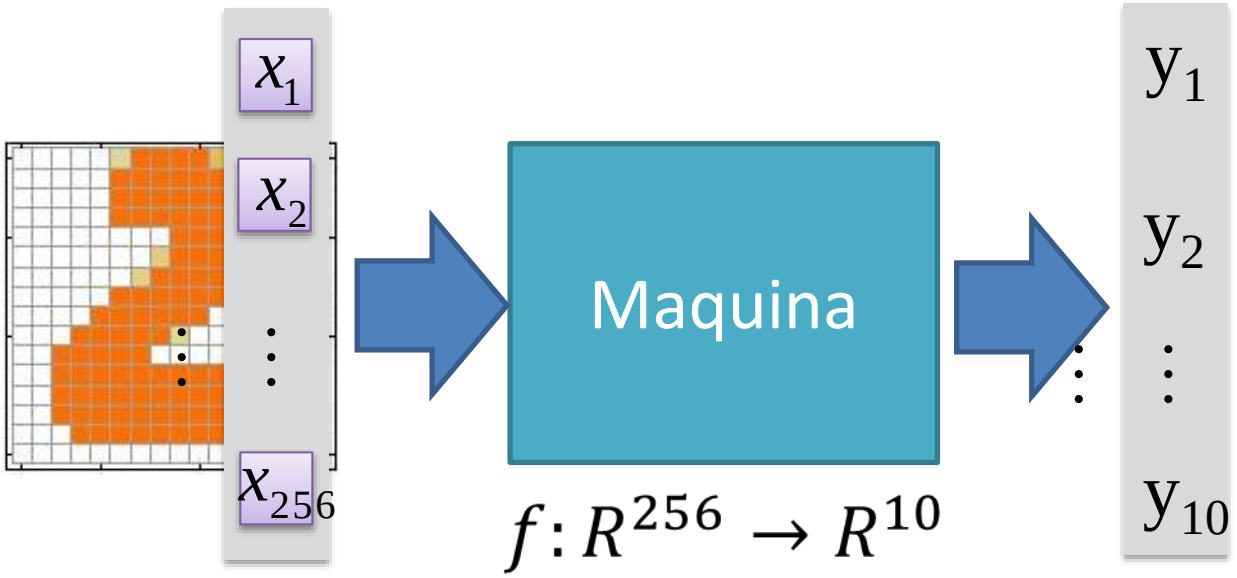
No ink → 0

Salida



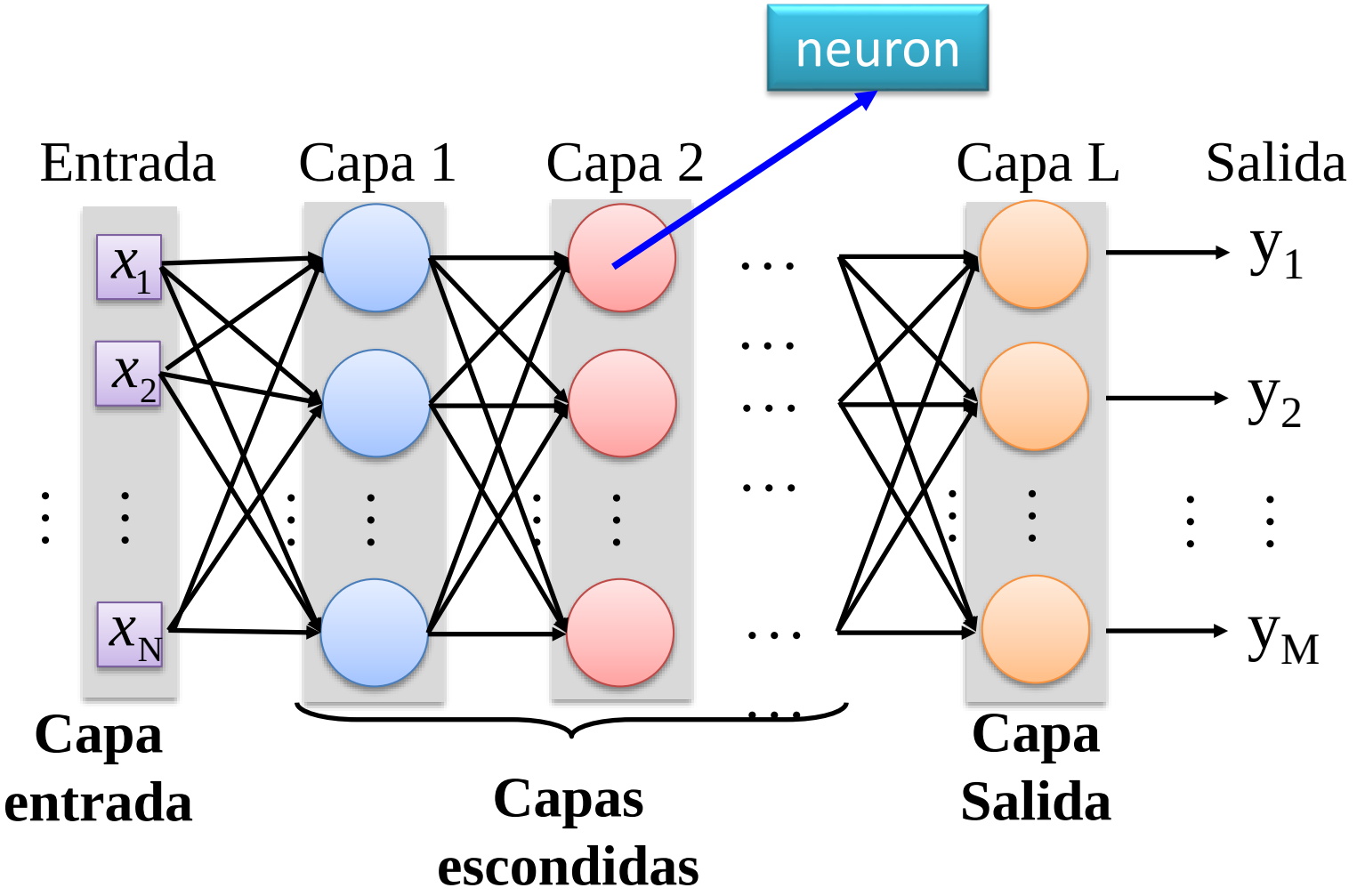
Cada dimensión representa la confianza de un dígito.

Reconocimiento de dígitos escritos



En aprendizaje profundo, la función f es una RNA

RNA



Deep significa muchas capas ocultas

¿Más profunda es mejor?

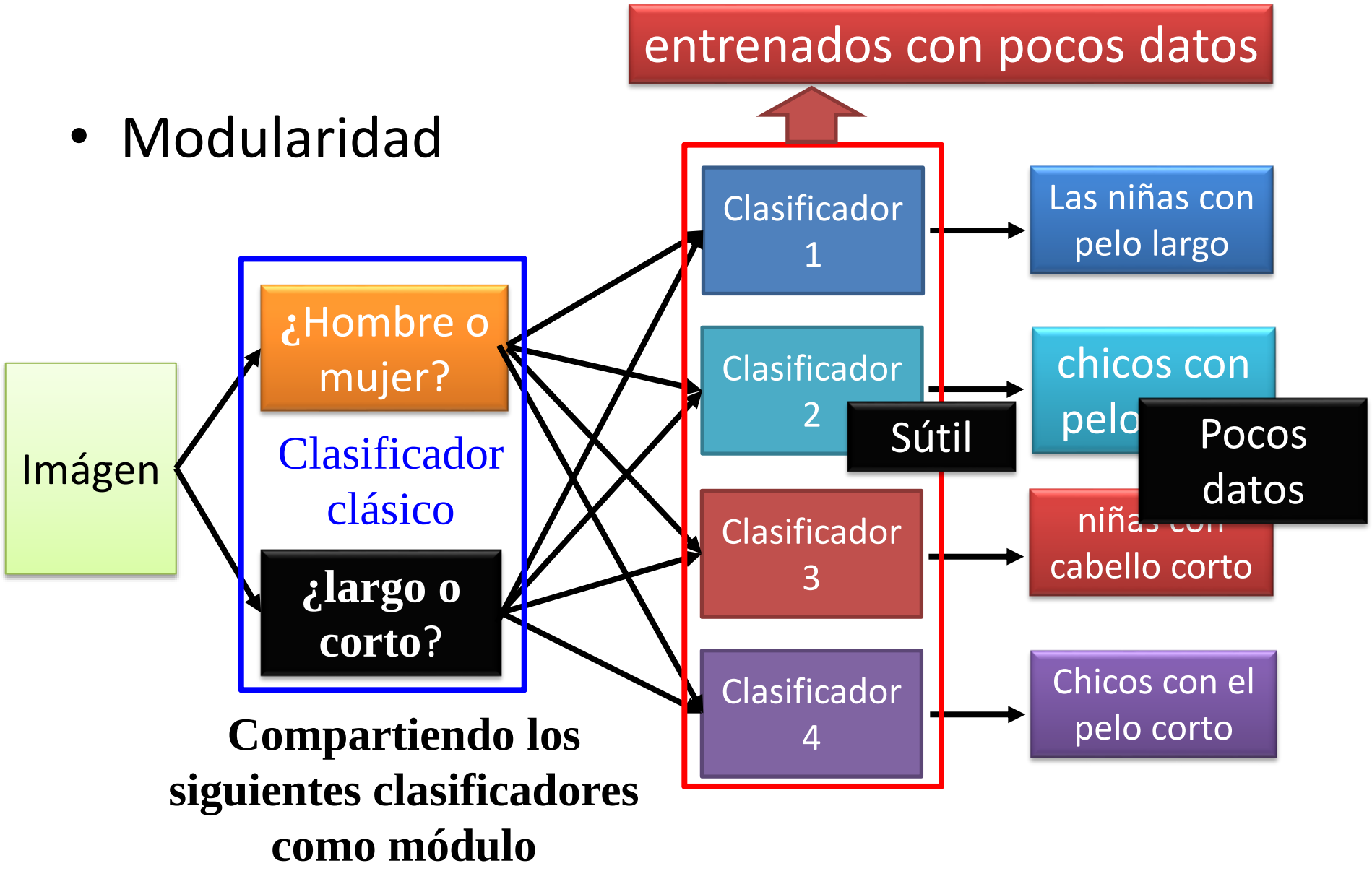
Capa x tamaños	Tasa error palabra (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

No sorprende, más parámetros, un mejor rendimiento

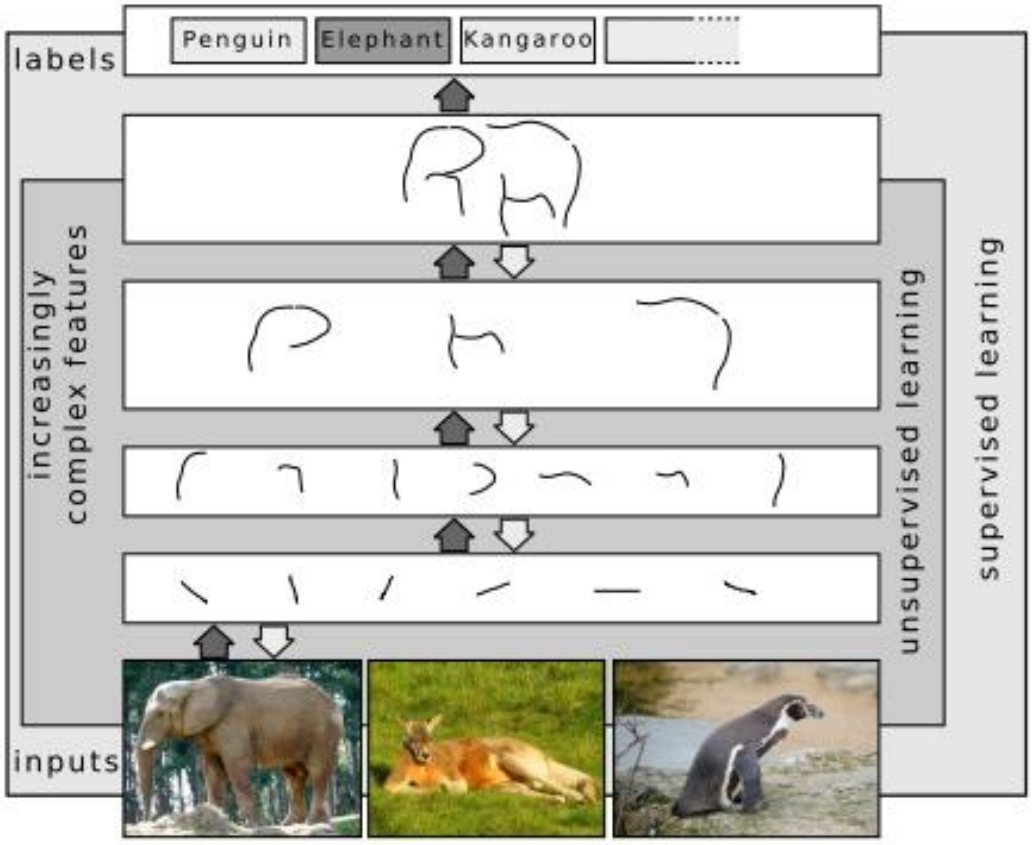
Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

¿Por qué Deep?

- Modularidad



Aprendizaje jerárquica

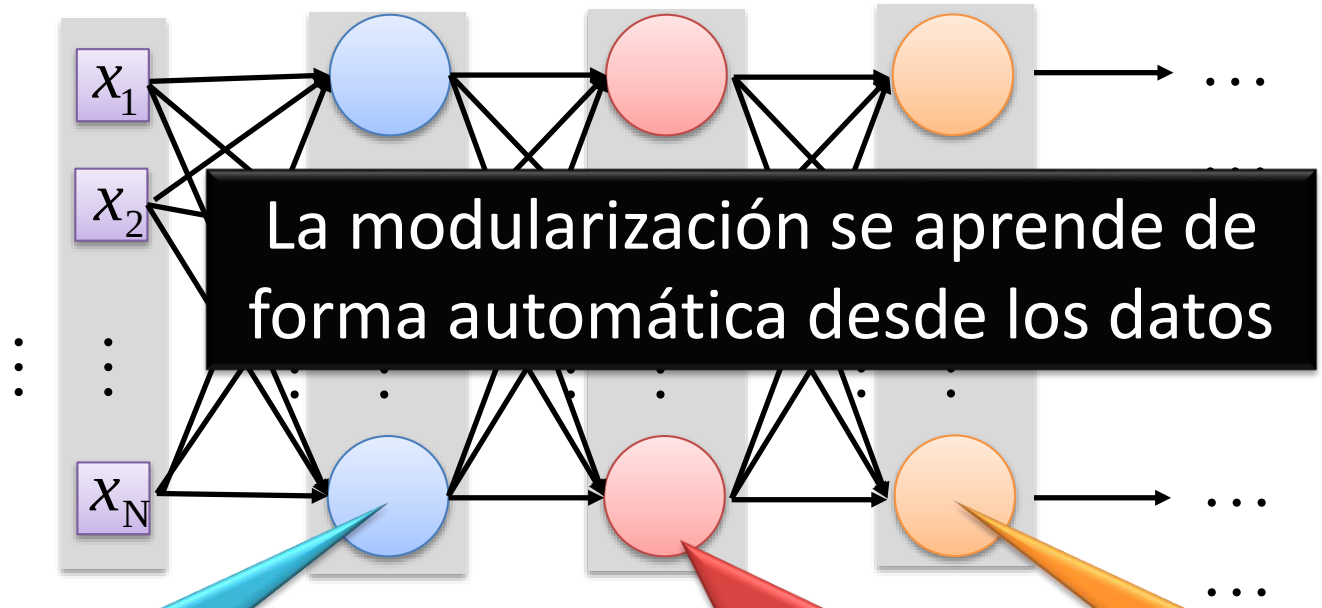


- progresión natural desde el bajo nivel a la estructura de alto nivel, como se ve en la complejidad natural
- Más fácil de controlar lo que se está aprendiendo y guiar la máquina a mejores subespacios

¿Por q

trabaja con conjuntos pequeños de datos

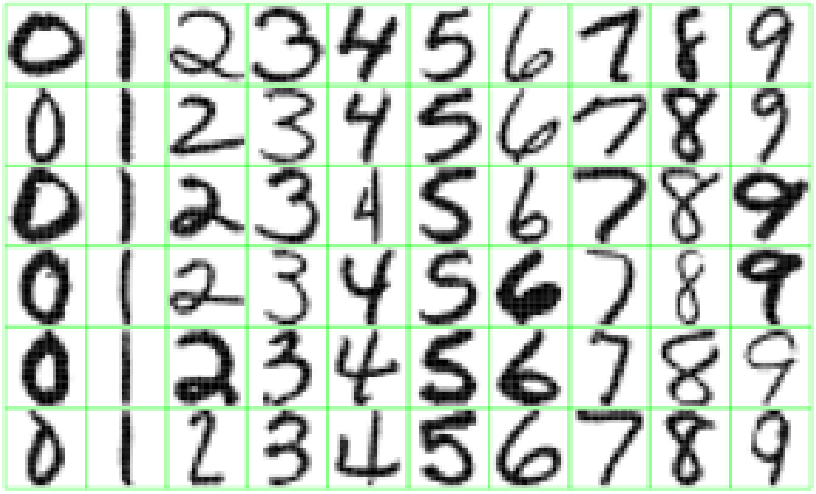
- Modularidad



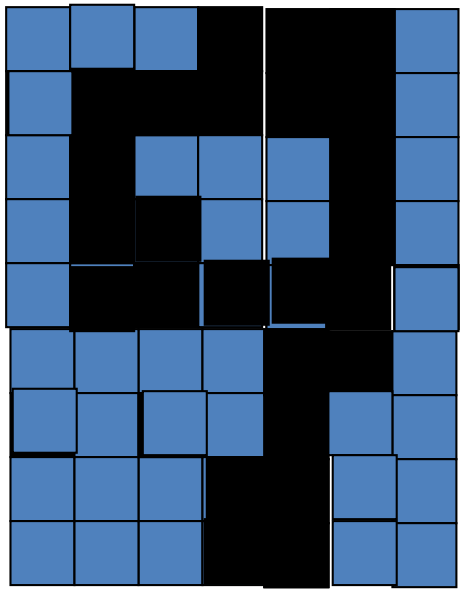
Los clasificadores más básicas

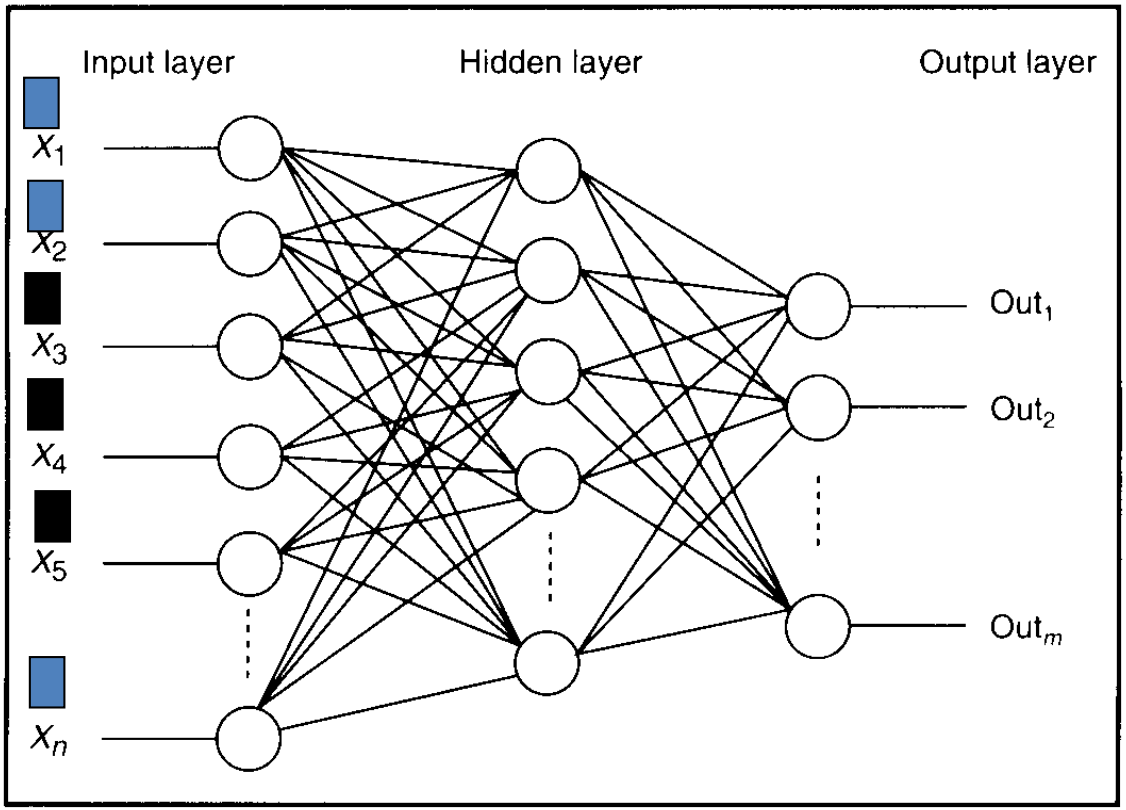
Usa primera capa como módulo para construir clasificadores

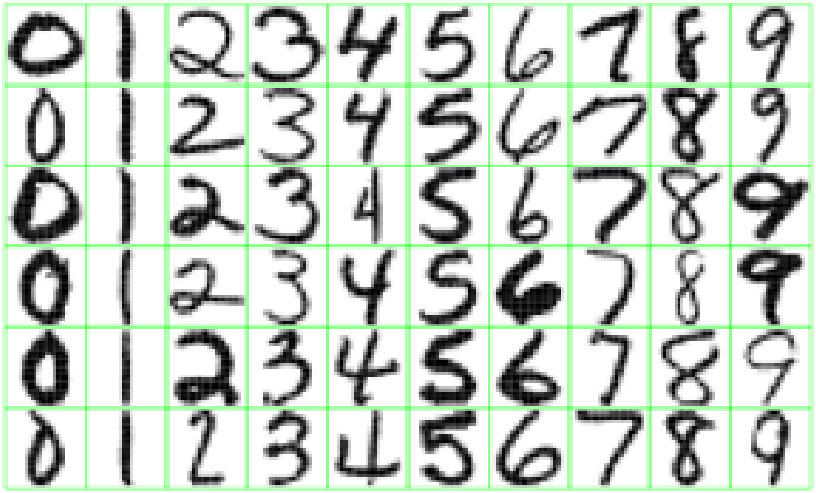
Usa segunda capa como módulo .....



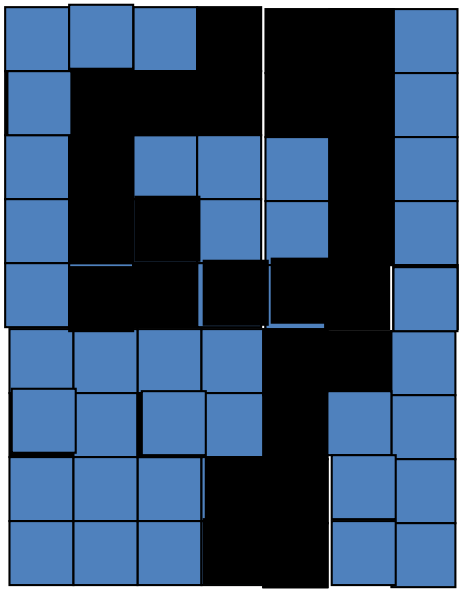
detectores de características

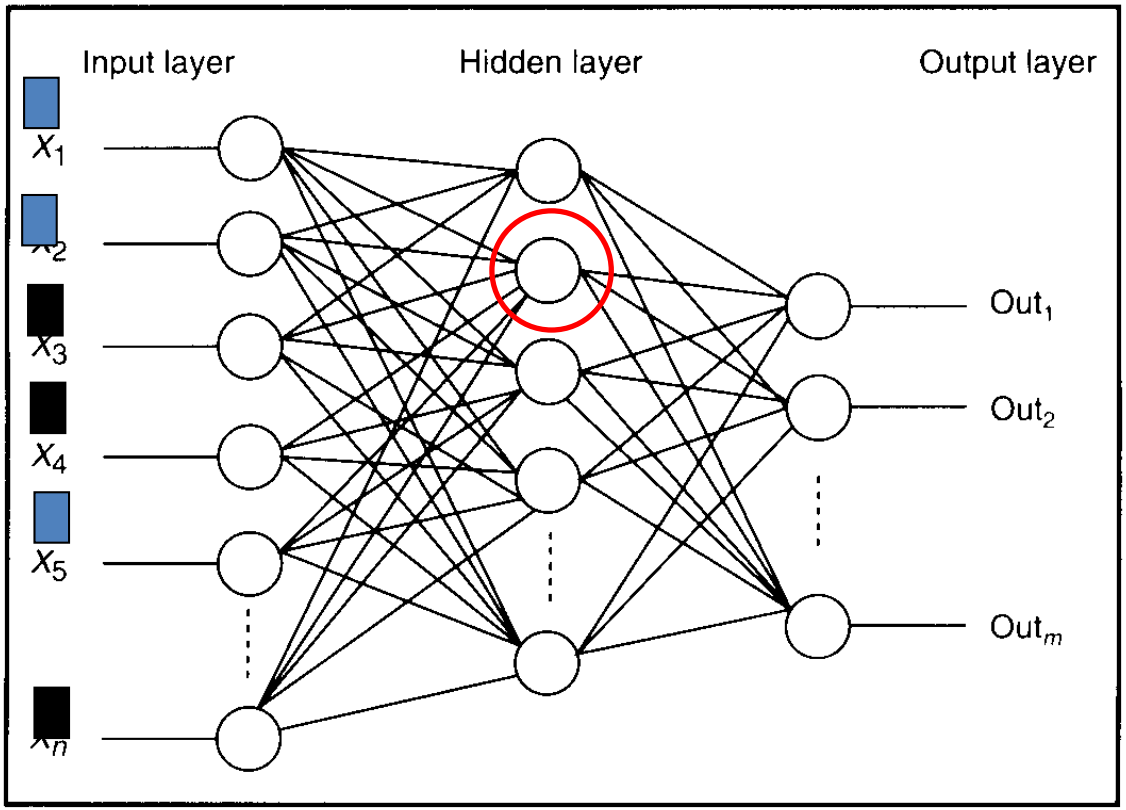




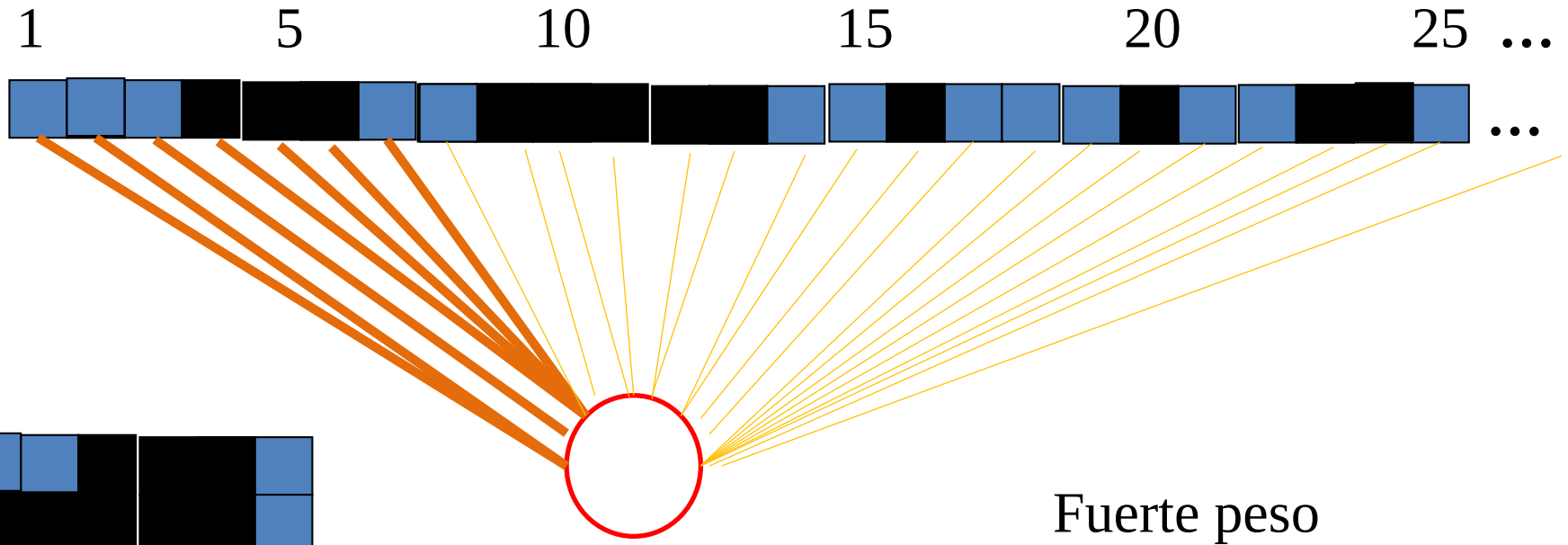


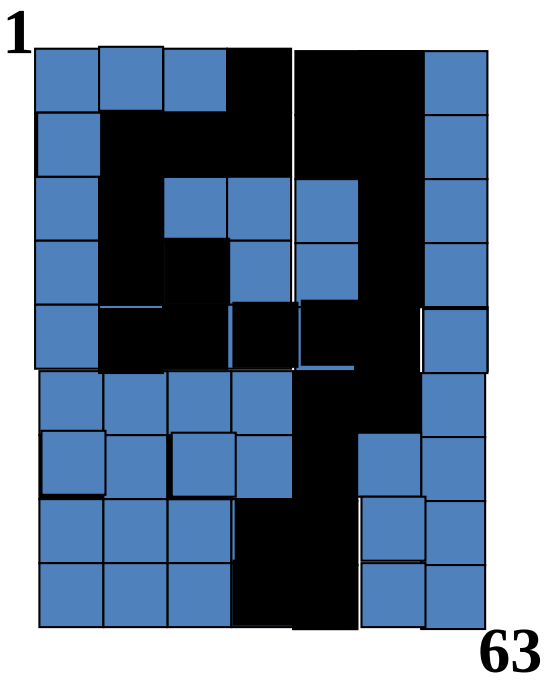
¿Qué hace?




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

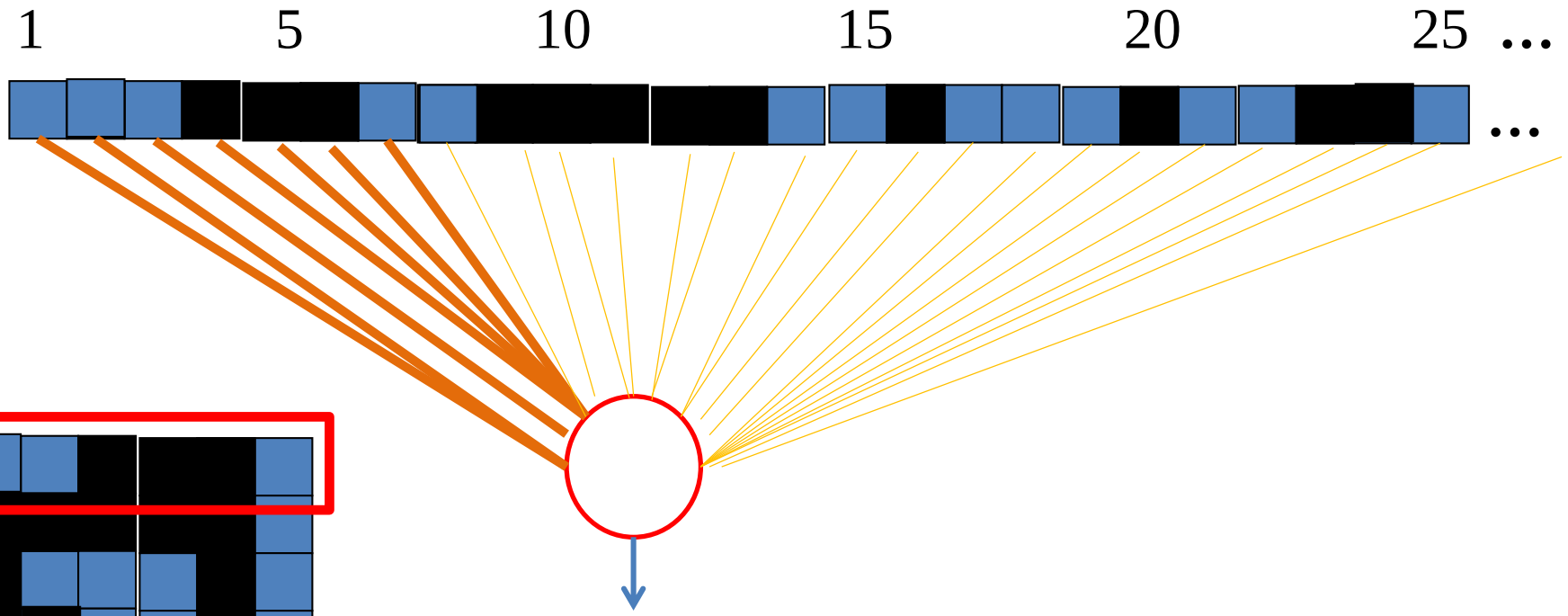


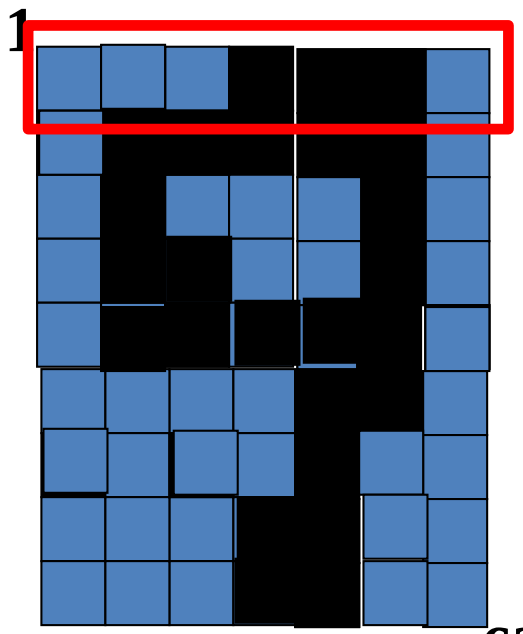


Fuerte peso

Bajo peso

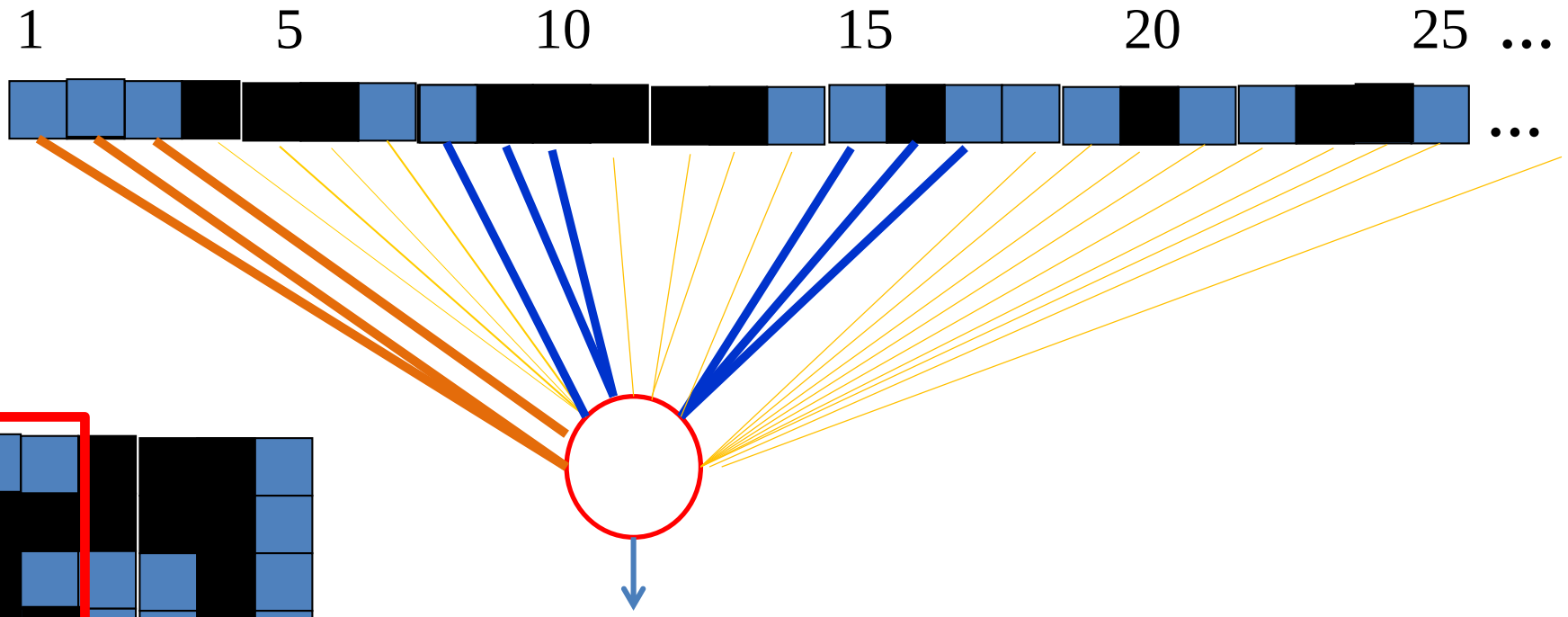
¿Qué detecta?

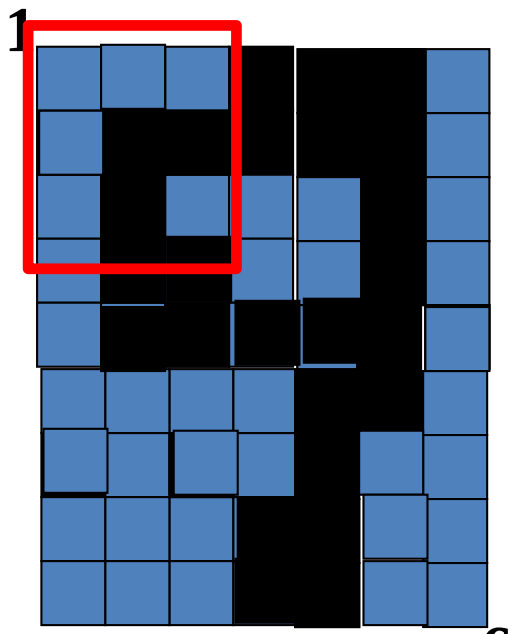




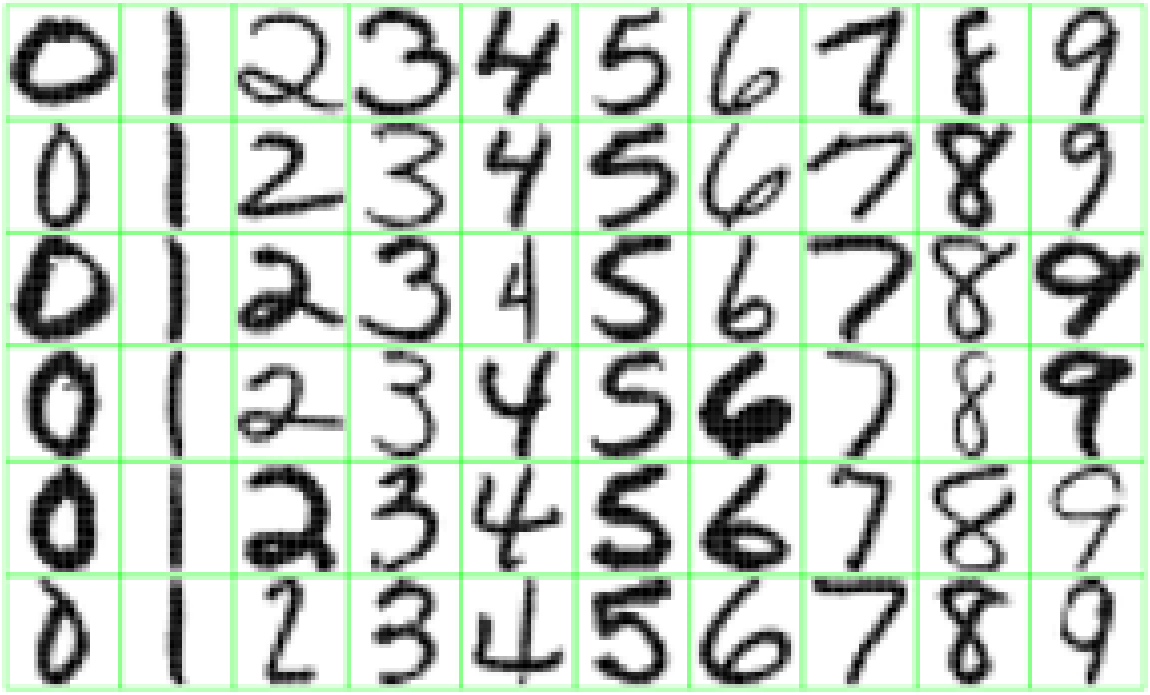
Enviará la señal fuerte para una línea horizontal en la fila superior, haciendo caso omiso de todos los demás sitios

¿Qué detecta?

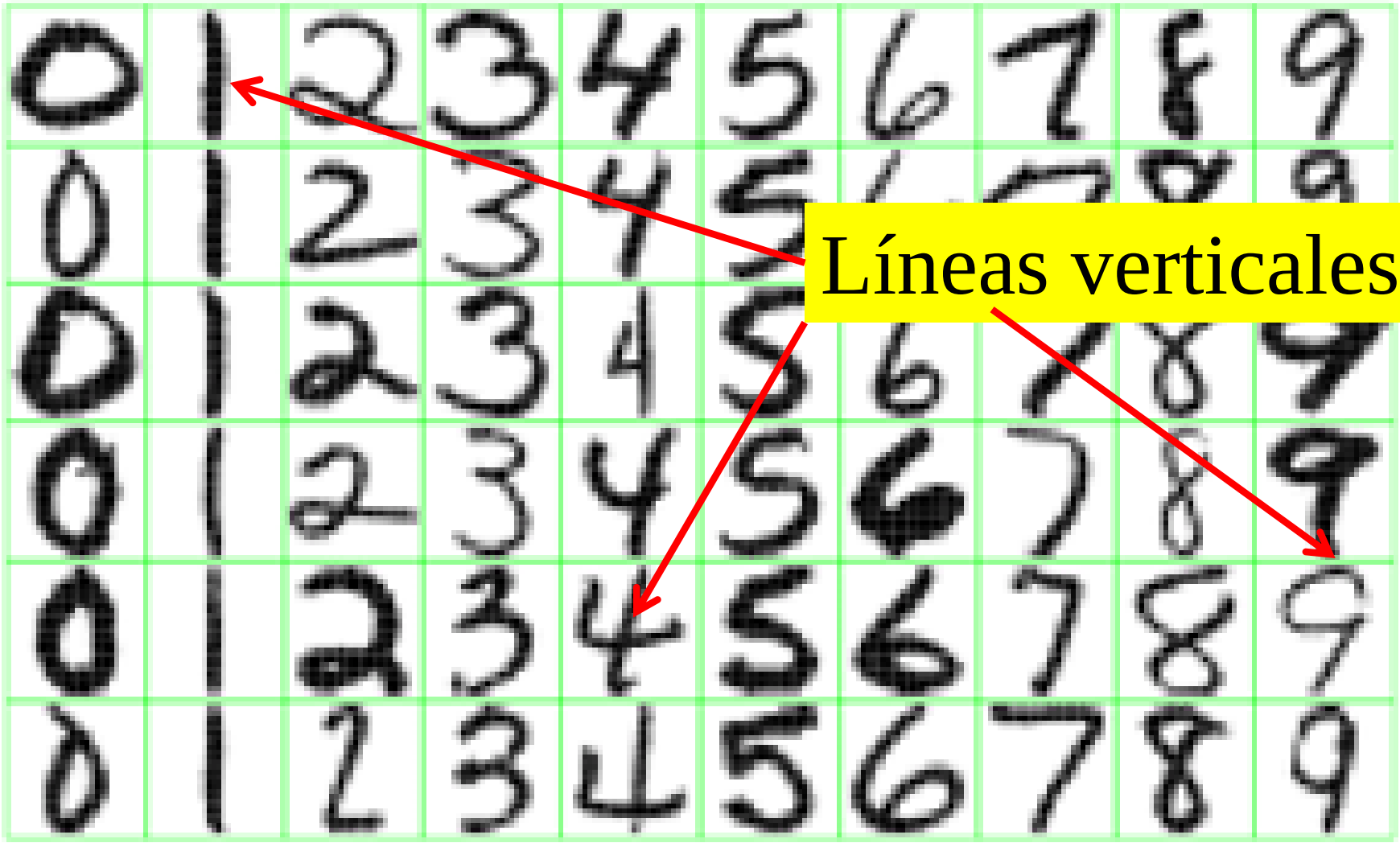




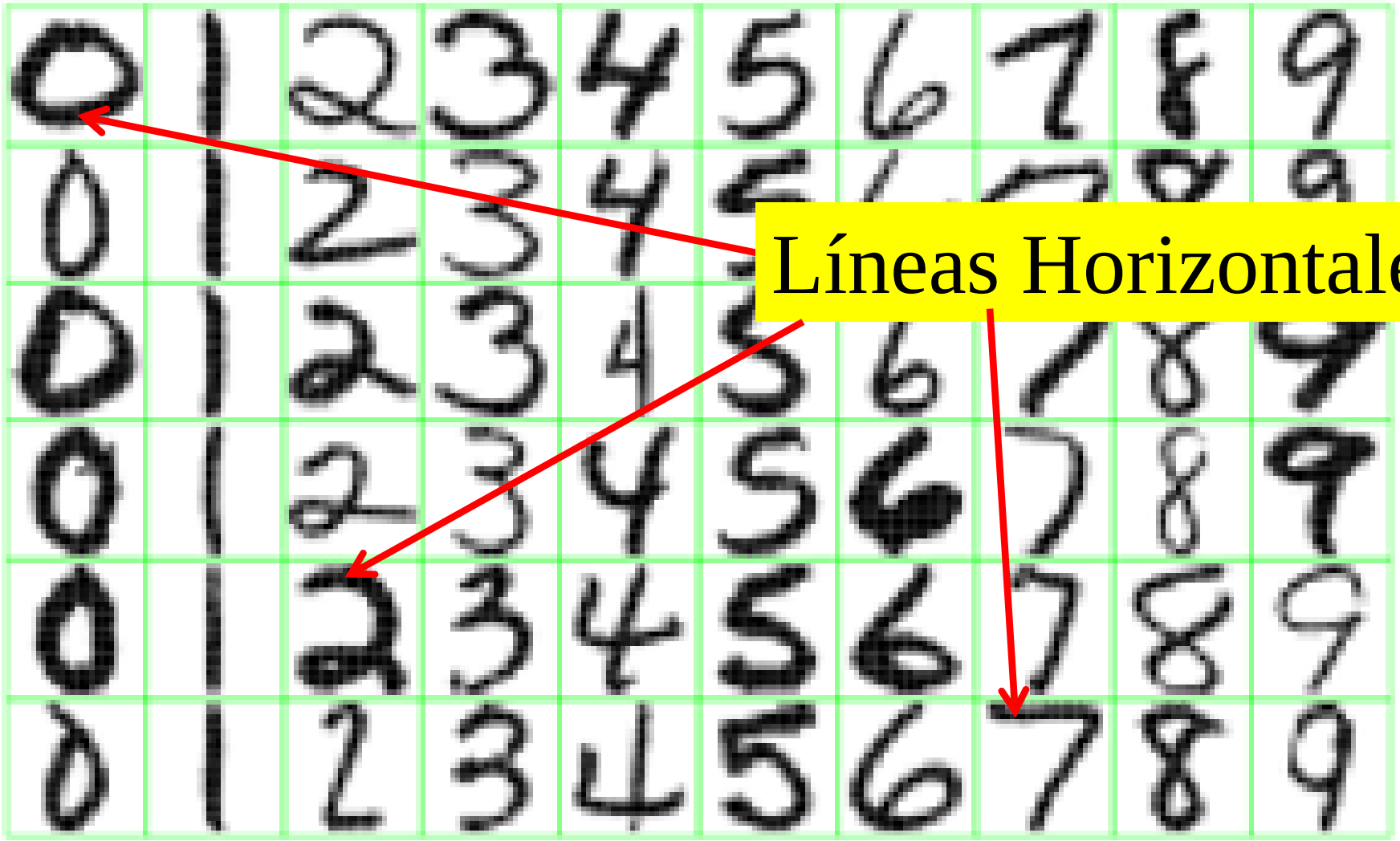
señal fuerte para un área oscura en la esquina superior izquierda



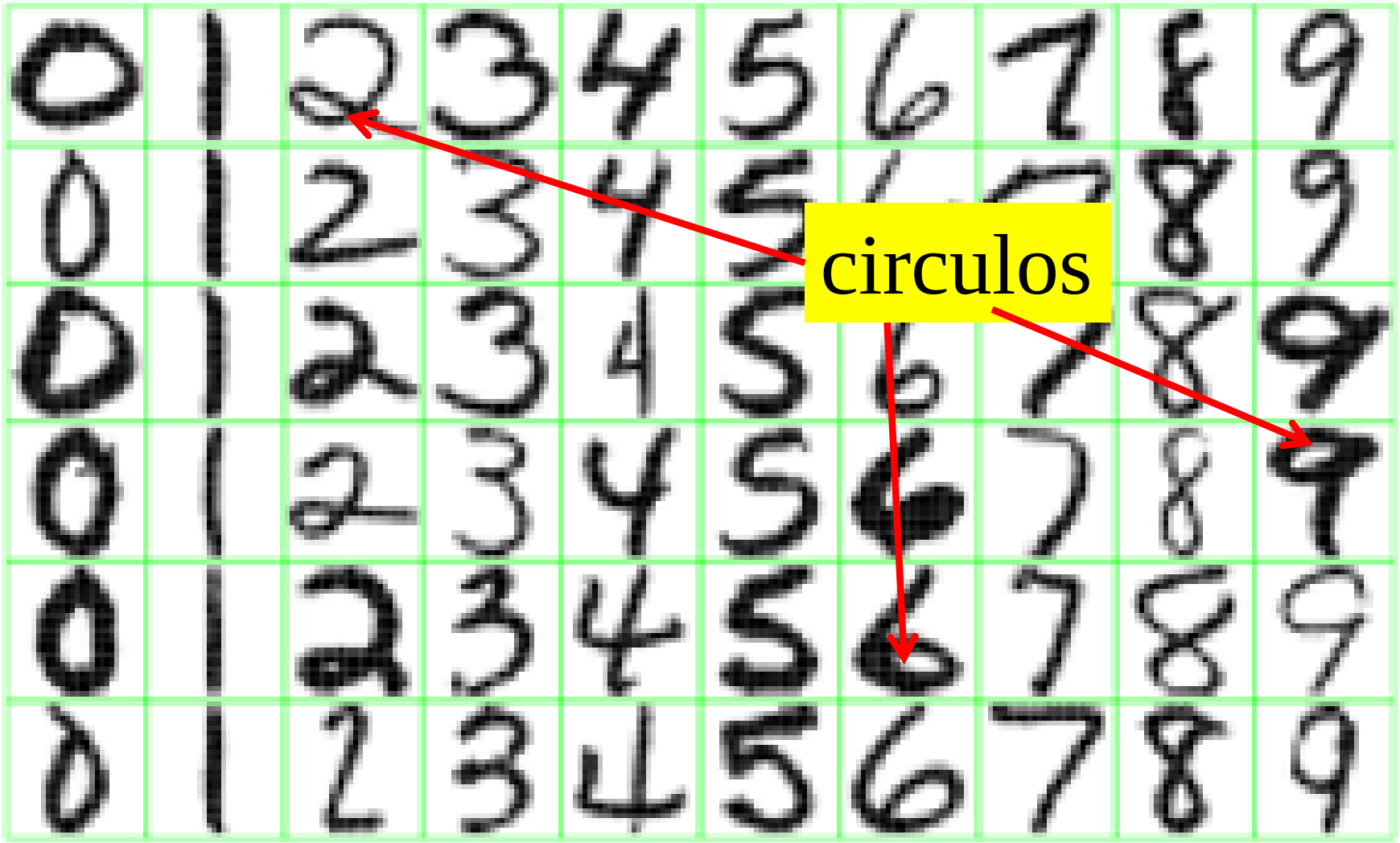
¿Qué características podría esperar una buena NN aprender, entrenados con datos como este?



1

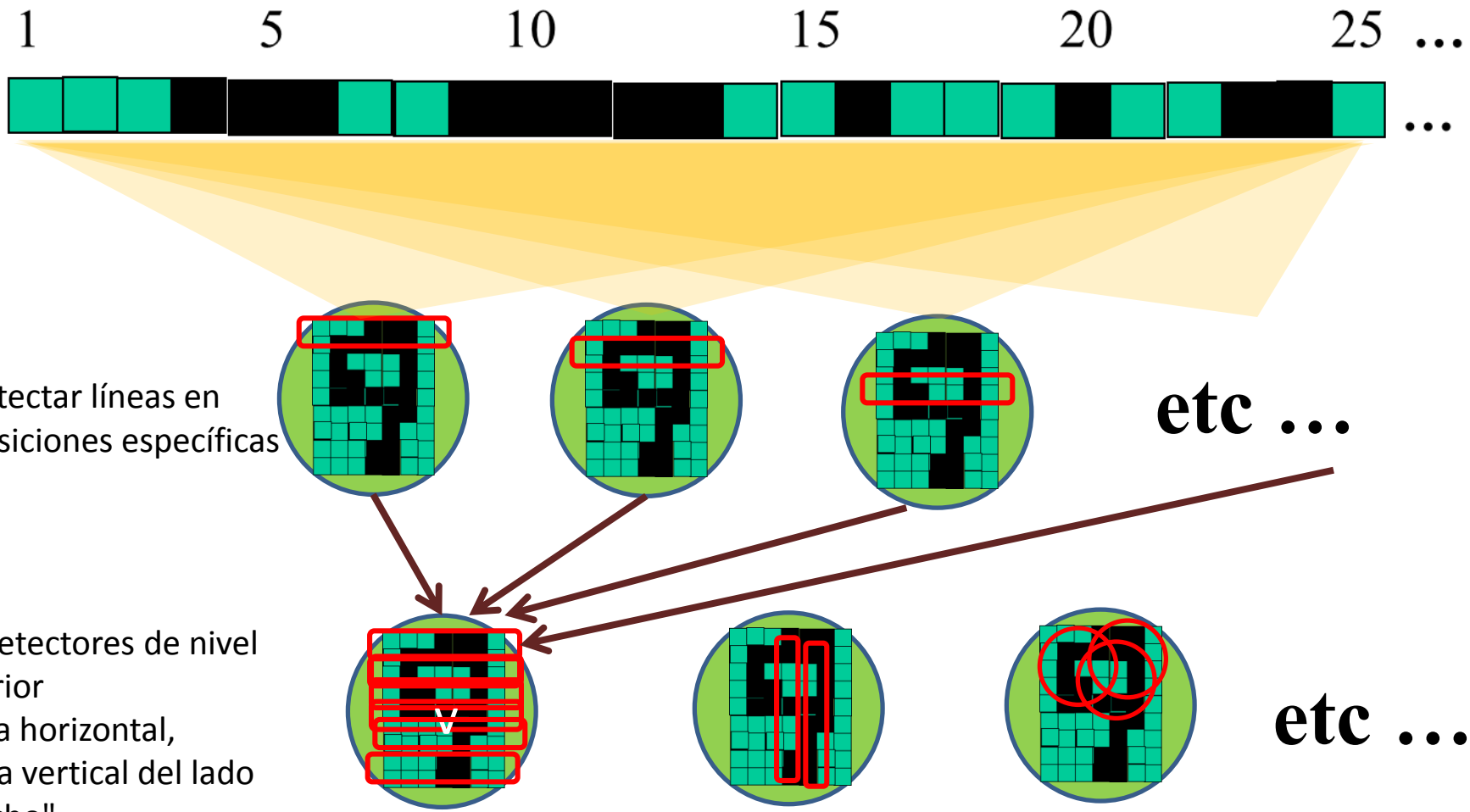


1

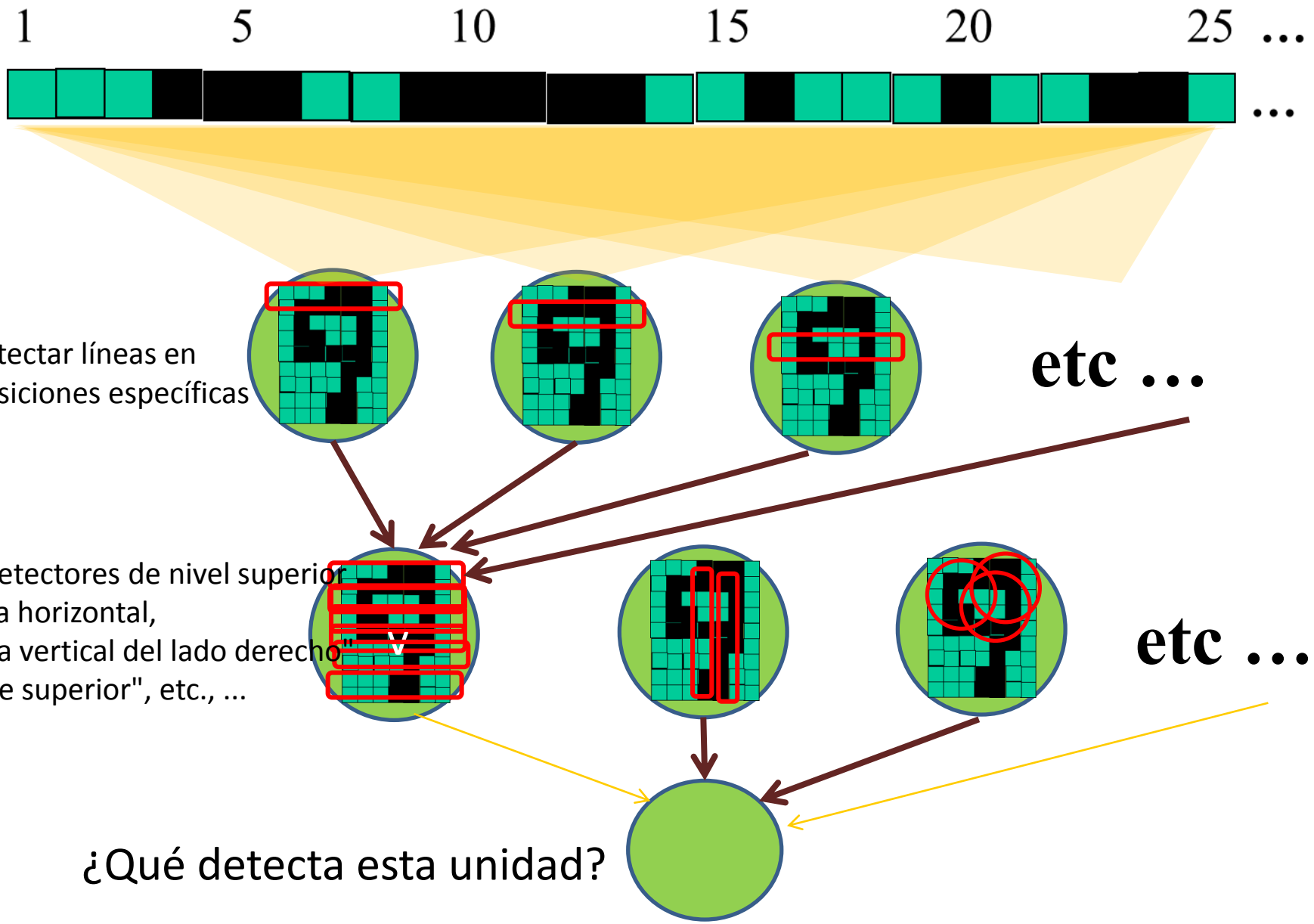


1

capas sucesivas pueden aprender características de nivel superior …



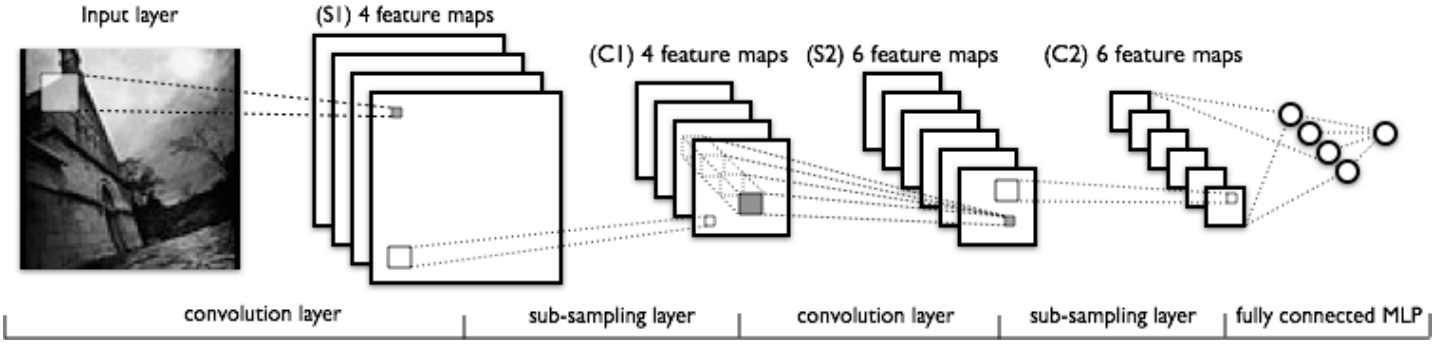
capas sucesivas pueden aprender características de nivel superior …



Convolutional Neural Networks

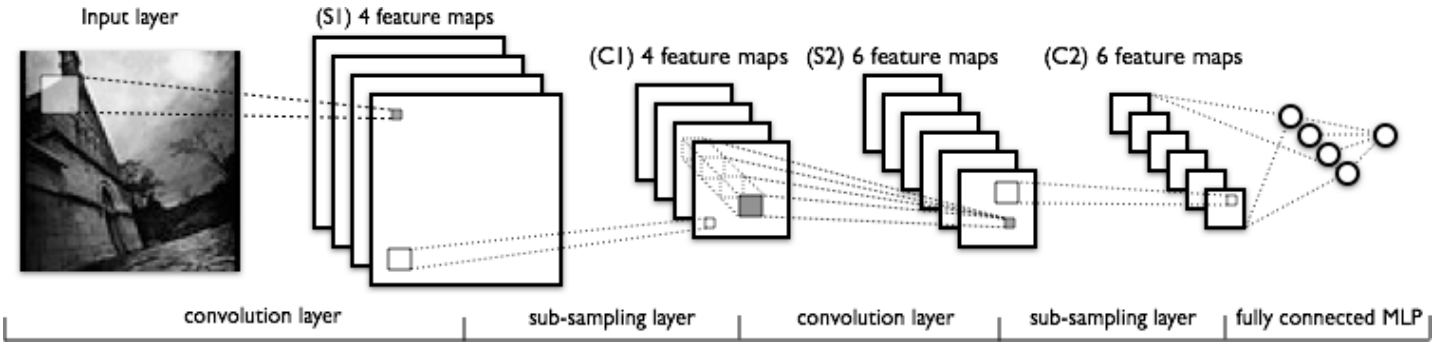
Mientras que los nodos en una estándar RNA obtiene información de todos los nodos de la capa anterior, **en CNN un nodo recibe sólo un pequeño conjunto de características que son espacial o temporalmente cercanas unos de otros**, llamados **campos receptivos** de una capa a la siguiente (por ejemplo,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

Puede, haber bordes, esquinas, puntos finales,



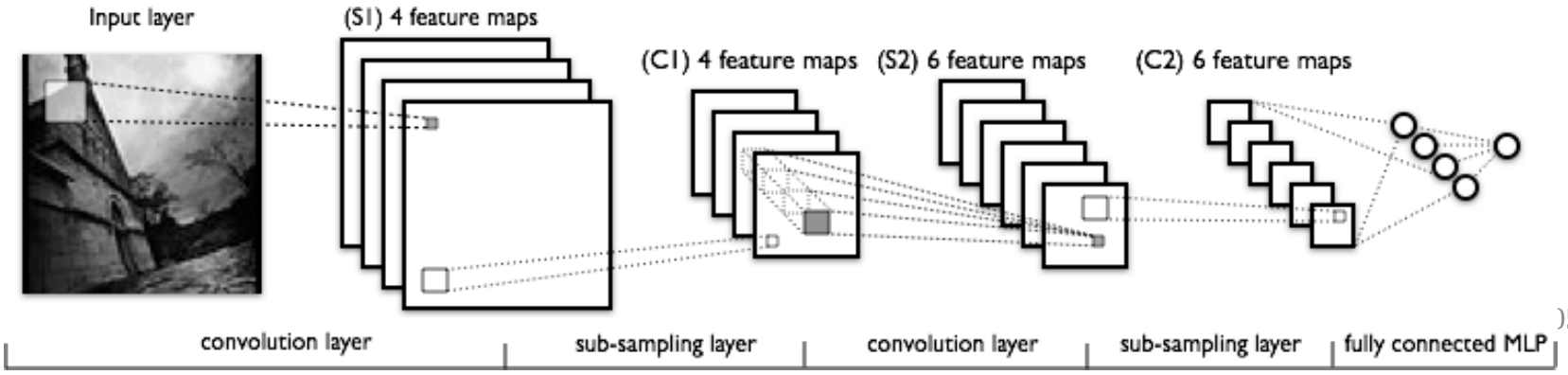
CNN

- Cada nodo es calculado para cada campo receptivo en la capa anterior
  - Durante el entrenamiento, los pesos correspondientes, 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 hace que el total de características aumentar



Submuestreo (agrupaciones)

- capas de convolución y sub-muestreo se intercalan
- Submuestreo (agrupaciones), permite varias características ser disminuidas, no superpuestas
  - 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
  - Puesto que después de la primera capa, siempre hay varios mapas de características de conectarse a la siguiente capa, se trata de una decisión humana en cuanto a que mapas anteriores recibe entradas del mapa actual



Formación de Redes de profundas

- Construir un espacio de características
  - entrenamiento no supervisado entre las capas puede descomponer el problema en sub-problemas distribuidos (con mayores niveles de abstracción), que se descomponen más a fondo en las capas posteriores

Representación de características

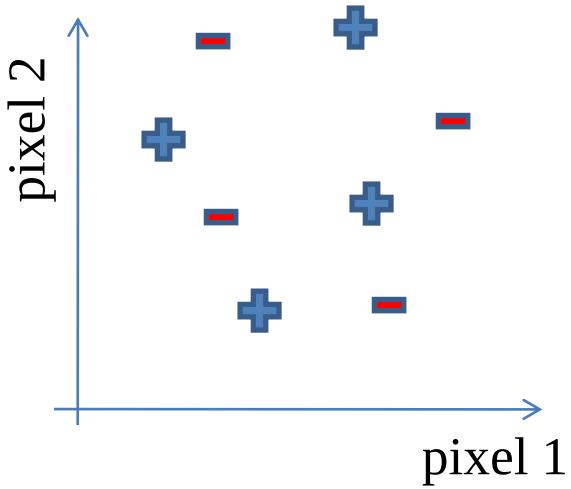
Algoritmo aprendizaje

Entrada

Espacio entrada

Motos

"No"-Motos



Representación de características

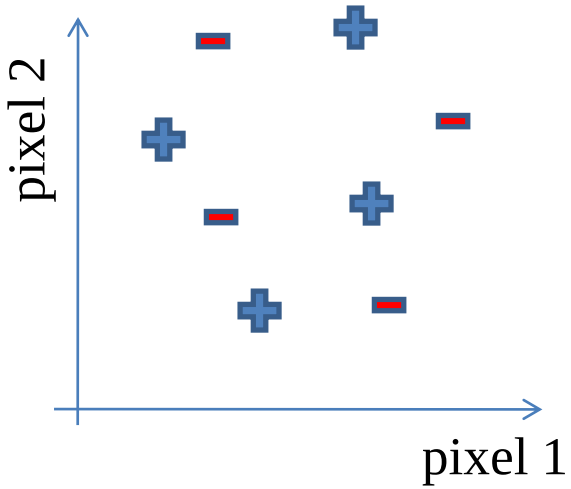
Entrada

representación de características

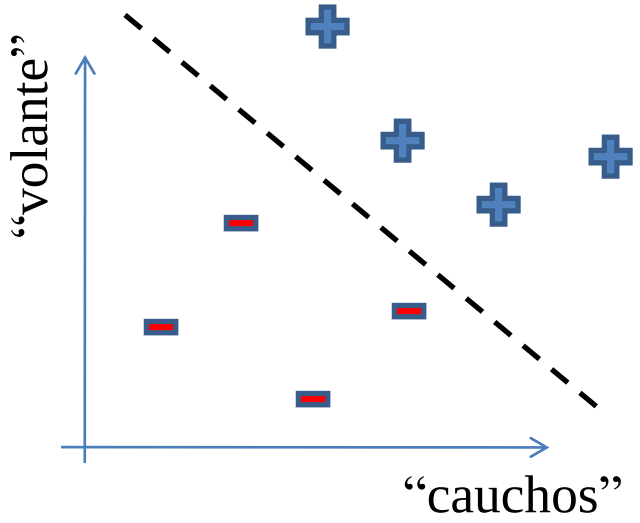
Algoritmo aprendizaje

Espacio entradas

Motos
Non"-Motorbikes

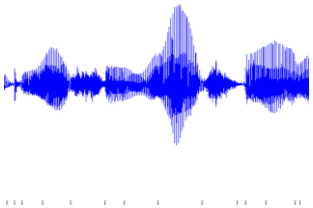


Espacio Características

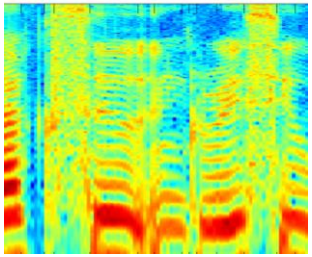


How is computer perception done?

Clasificación audio



Audio

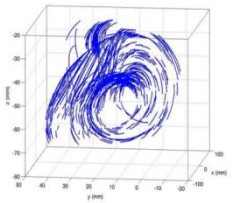


características de audio de bajo nivel

Identificación de quien habla

Control helicóptero

Helicóptero



características de estado de bajo nivel

Acción

Algunas Deep Learning NN

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

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

Agente Inteligente

- **Sabe plantearse un problema y buscarle soluciones**

  => técnicas de búsqueda y resolución de problemas

- **Tiene formas de representar el conocimiento, razonar y manejar la incertidumbre**

  => Lógicas y Mecanismos de manejo de conocimiento, incluso incierto, para posibilitar las diferentes formas de razonamiento

- **Es capaz de planificarse y aprender**

- **Es capar de comunicarse**

  => Sistemas Multiagentes