Tipos de RNA

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

ADALINE Y MADALINE

- **ADAptive Linear Element Y MULTIPLE ADALINE (WIDROW 1960)**
- **PARECIDO AL PERCEPTRON**
  - FUNCION DE TRANSFERENCIA ESCALON
  - UNA UNICA SALIDA
- **DIFERENCIA**
  - MECANISMO DE APRENDIZAJE

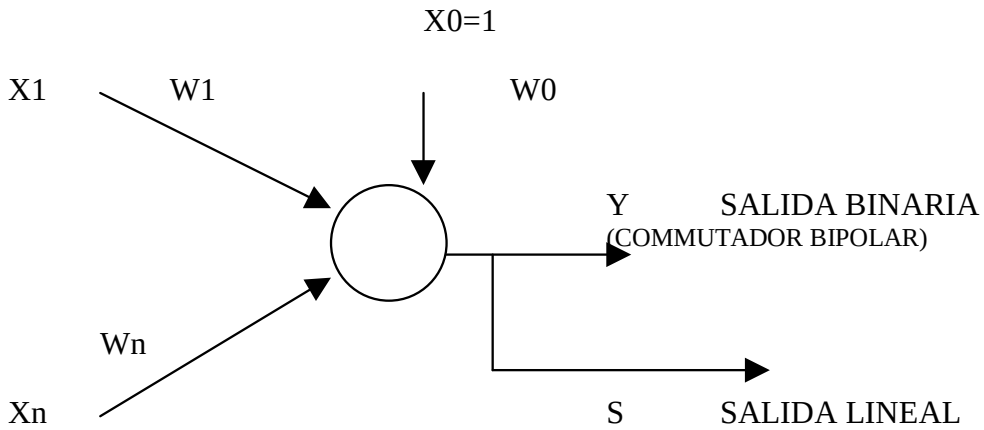
ADALINE Y MADALINE

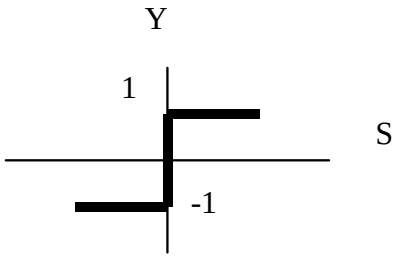
- APRENDIZAJE
  - REGLA DE MINIMO CUADRADO
  - DIFERENCIA ENTRE d(t) Y SALIDA LINEAL (PERCEPTRON ES CON RESPECTO A SALIDA BINARIA)
  - FUERA DE LINEA
  - SUPERVISADO

$E_k=d_k-y_k$ (perceptron) **$E_k=d_k-S_k$ (adaline)**

ADALINE Y MADALINE

COMBINADOR ADAPTATIVO LINEAL





ADALINE Y MADALINE

$$E_k = 1/2 \sum_{k=1}^{l} E_k^2$$

$$\Delta Wij(t) = -\alpha \, \partial E_k^2 / \partial Wi = \alpha \, E_k \, x_{ki}$$

$$Wij(t+1) = Wij(t) + \alpha \, (d_k - S_k) \, x_{ki}$$

ADALINE Y MADALINE

1. APLICAR VECTOR ENTRADA X_k

2 OBTENER SALIDA LINEAL

$$S_k = \sum W_k X_k$$

3. CACULAR ERROR E_k

4. ACTUALIZAR PESOS

5. SI E_k^2 *< VALOR PEQUENO*

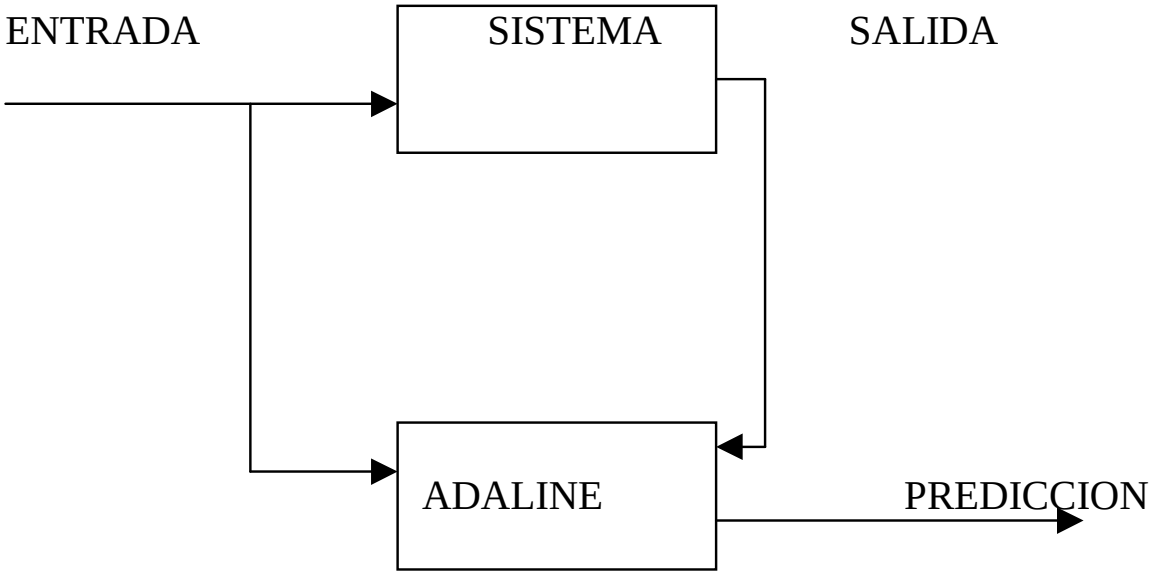
PARAR

DE LO CONTRARIO

REGRESAR A 1 CON TODOS LOS PATRONES

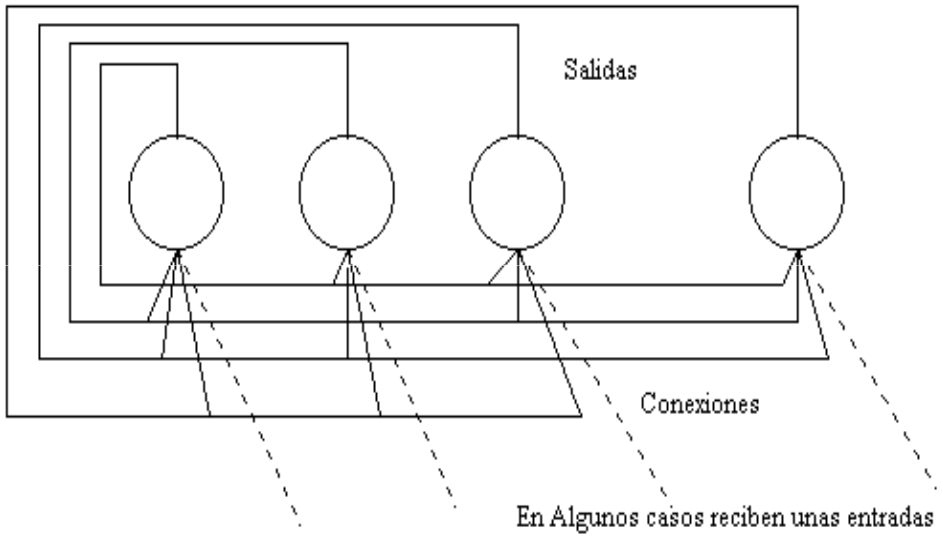
ADALINE (APLICACIÓN)

PROCESAMIENTO DE SENALES



Modelo de Hopfield

- John Hofield, fue uno de los responsables del desarrollo de 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]

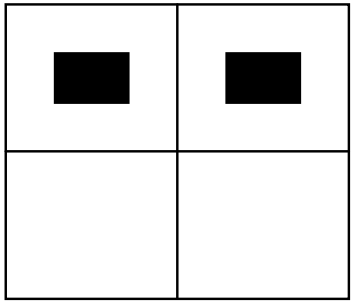
$$= \begin{cases} \sum_{k=1}^{M} (2e_i^{(k)} - 1)(2e_j^{(k)} - 1) & ;1 \le i,j \le N; i \ne j \\ 0 & ;1 \le i,j \le N; i = j \end{cases}$$

MODELO DE HOPFIELD

ENTRENAMIENTO

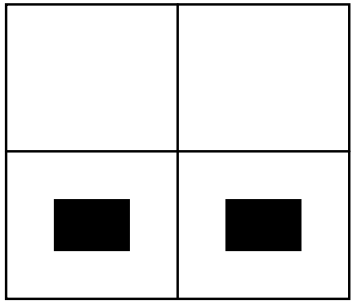
- FIGURA A, B, ...

Figura A



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

Figura B



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

MODELO DE HOPFIELD

ENTRENAMIENTO

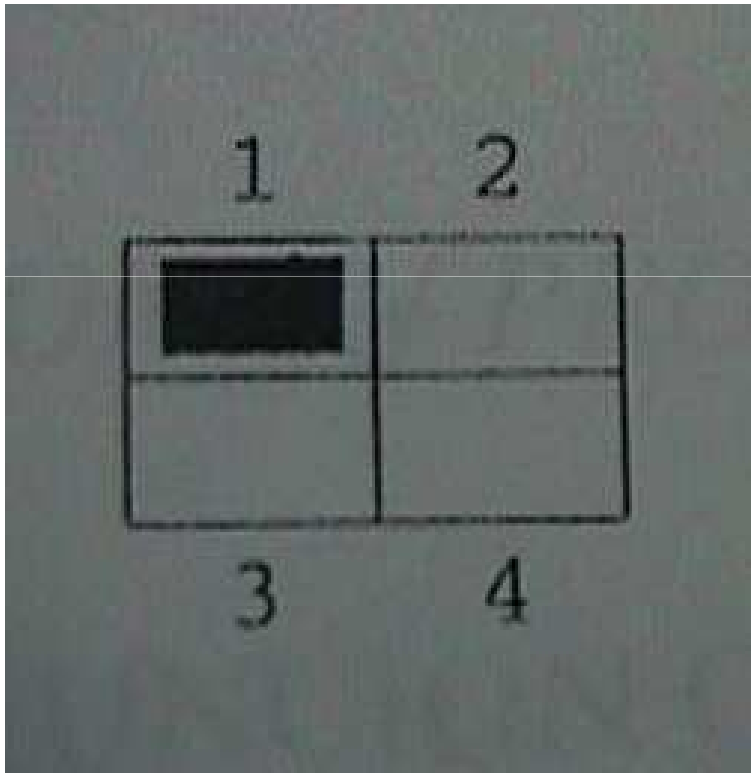
$$E_1 = [1,1,-1,-1] \qquad E_2 = [-1,-1,1,1]$$

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

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

MODELO DE HOPFIELD

FUNCIONAMIENTO



E = [1 -1 -1 -1]

MODELO DE HOPFIELD

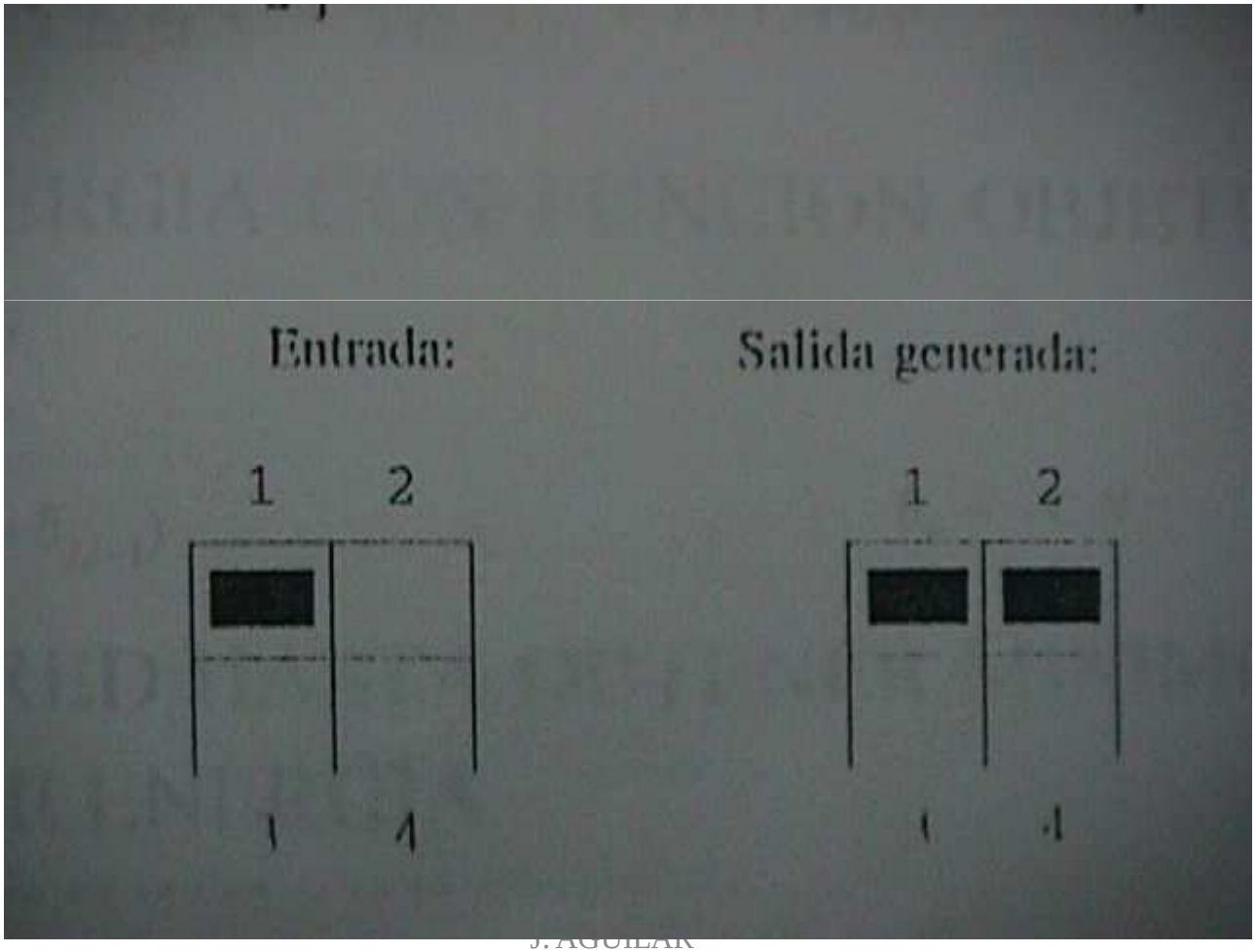
FUNCIONAMIENTO

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

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

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:

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

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

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

MODELO DE HOPFIELD

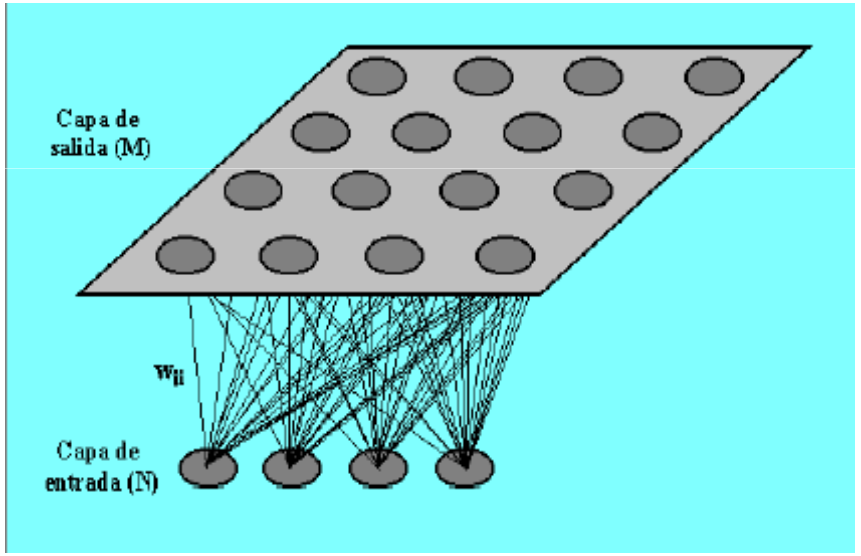
APLICACIÓN EN PROBLEMAS DE OPTIMIZACIÓN

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

=> MÍNIMO VALOR FUNCIÓN OBJETIVO

Modelo de Kohonen

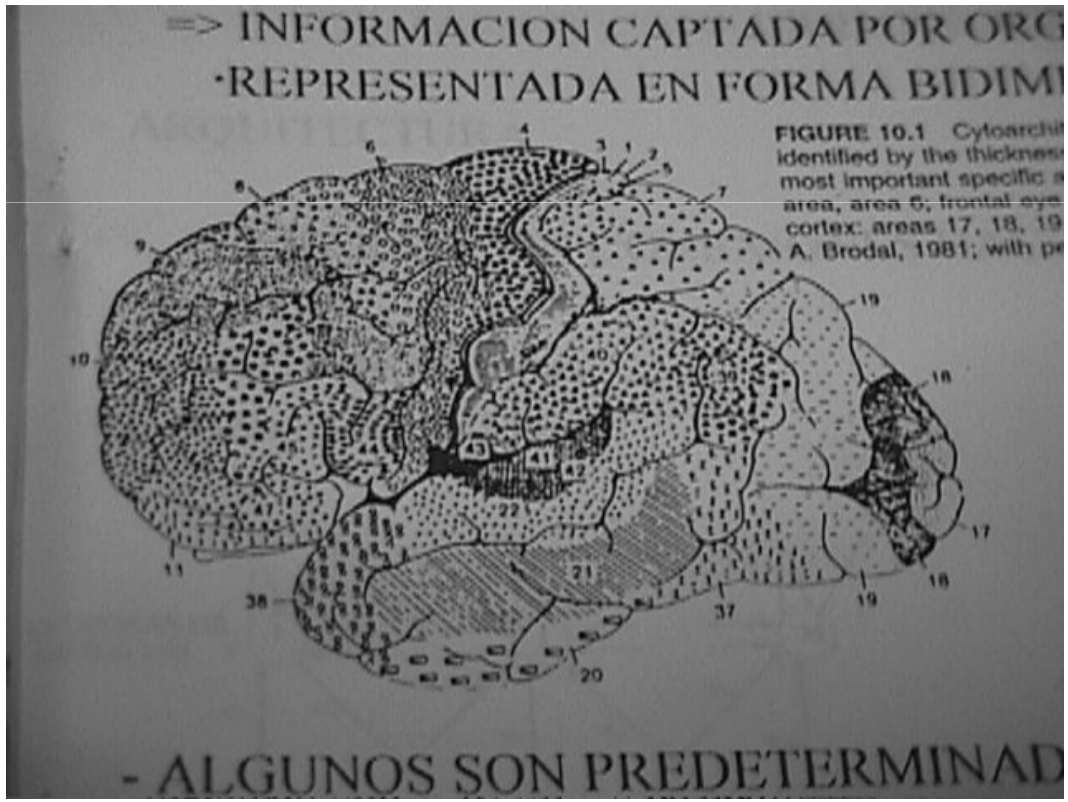
- ✓ Fue propuesto por Teuvo Kohonen en 1982.
- ✓ Este modelo de red es denominado mapas autoorganizados o SOM (*Self-Organizing Maps*).
- ✓ Este tipo de red se caracteriza por poseer un aprendizaje no supervisado competitivo.
- ✓ Un modelo SOM está compuesto por dos capas de neuronas. La capa de entrada, la capa de salida.
- ✓ Este tipo se aplica a los problemas típicos de agrupamiento de patrones, de clasificación. optimización.



MODELO DE KOHONEN

- **EN EL CEREBRO LAS NEURONAS SE ORGANIZAN EN ZONAS**

=> INFORMACIÓN CAPTADA POR ÓRGANOS SENSORIALES REPRESENTADA EN FORMA BIDIMENSIONAL



MODELO DE KOHONEN

- ALGUNOS SON PREDETERMINADOS GENETICAMENTE, OTROS PROVIENEN DEL APRENDIZAJE

  => FORMAR MAPAS TOPOLÓGICOS

- 1982 KOHONEN PROPONE RNA CON CAPACIDAD DE FORMAS MAPAS DE CARACTERÍSTICAS

  => ESTABLECER CARACTERÍSTICAS COMUNES ENTRE LA INFORMACIÓN DE ENTRADA A LA RED

MODELO DE KOHONEN

- DOS VERSIONES:

  - LVQ (LEARNING VECTOR QUANTIZATION)

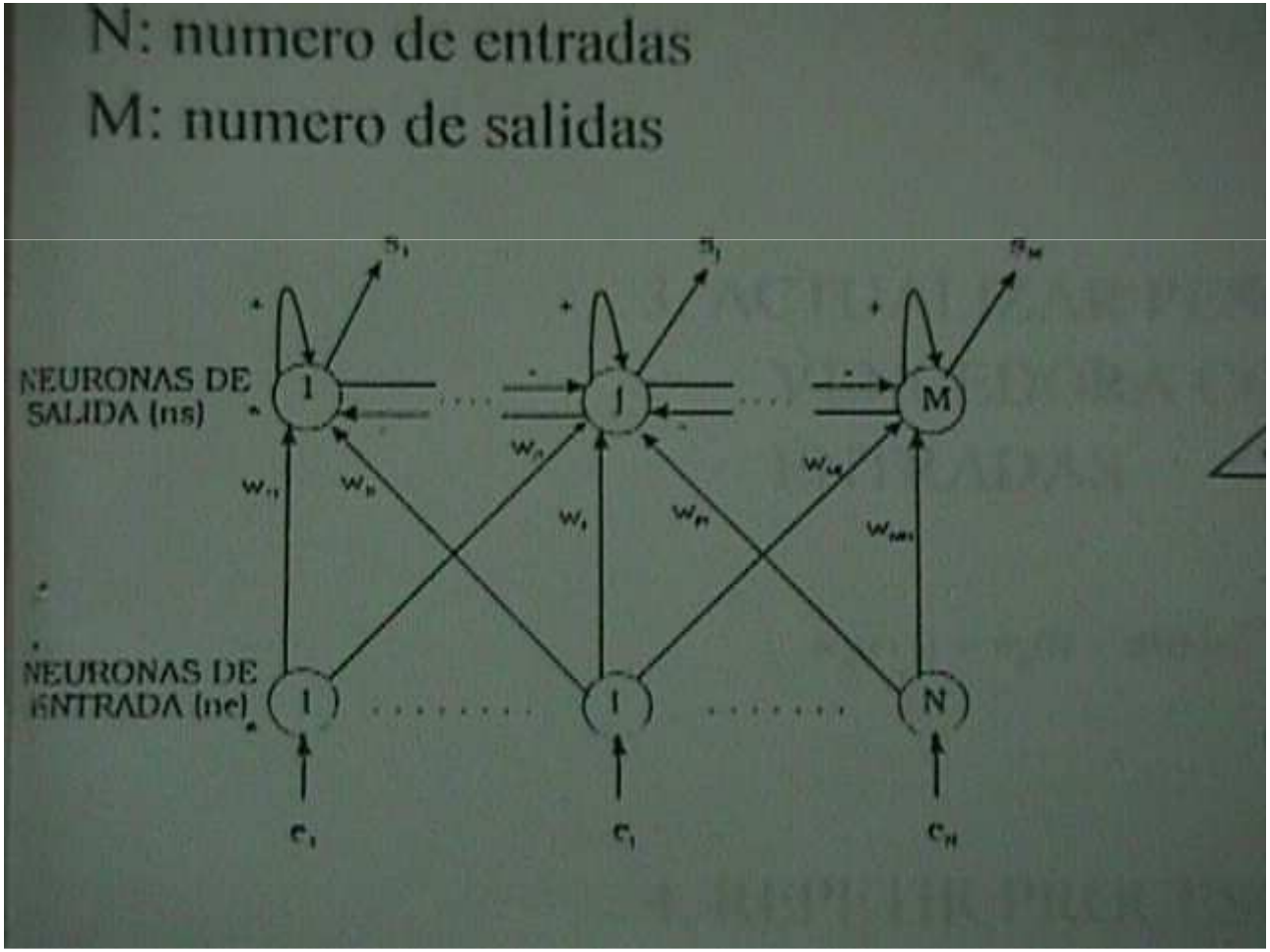
    -> UNIDIMENSIONAL

  - SOM (SELF-ORGANING MAP)

    -> BIDIMENSIONAL

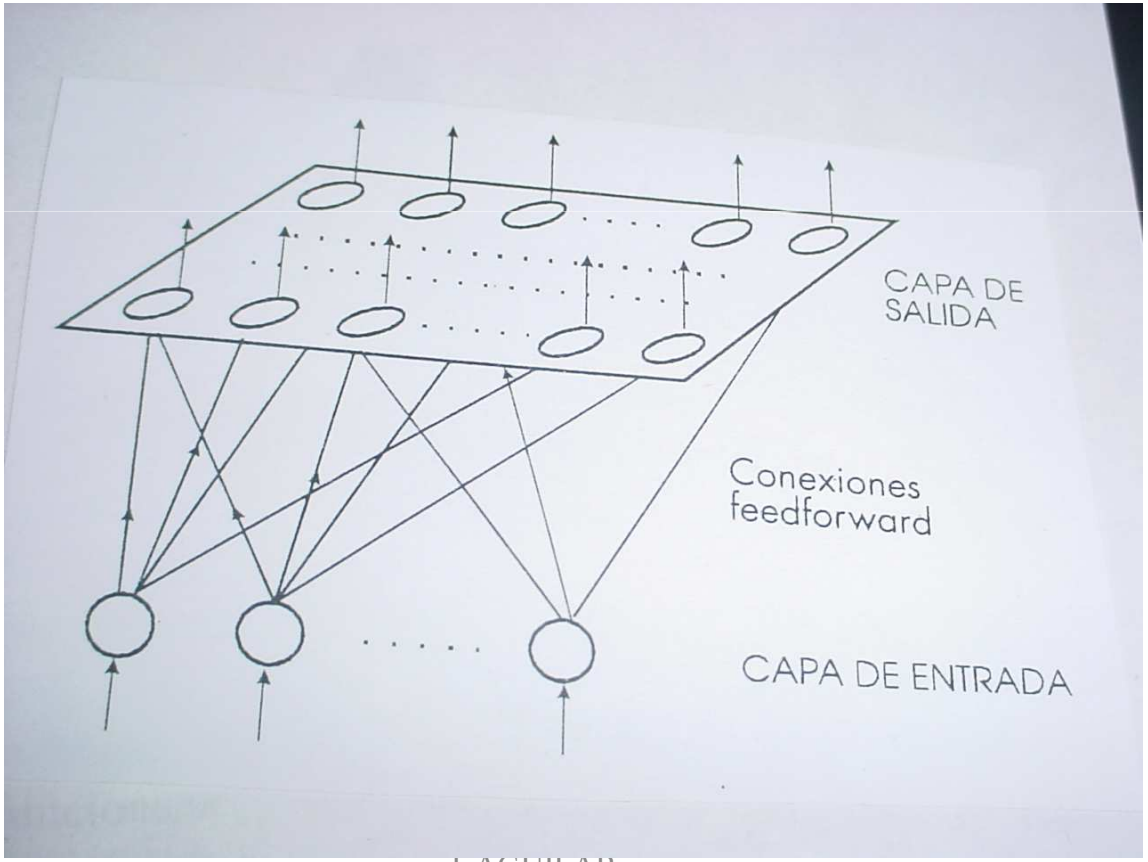
MODELO DE KOHONEN

- ARQUITECTURA
  - 2 CAPAS CON CONEXIONES HACIA ADELANTE



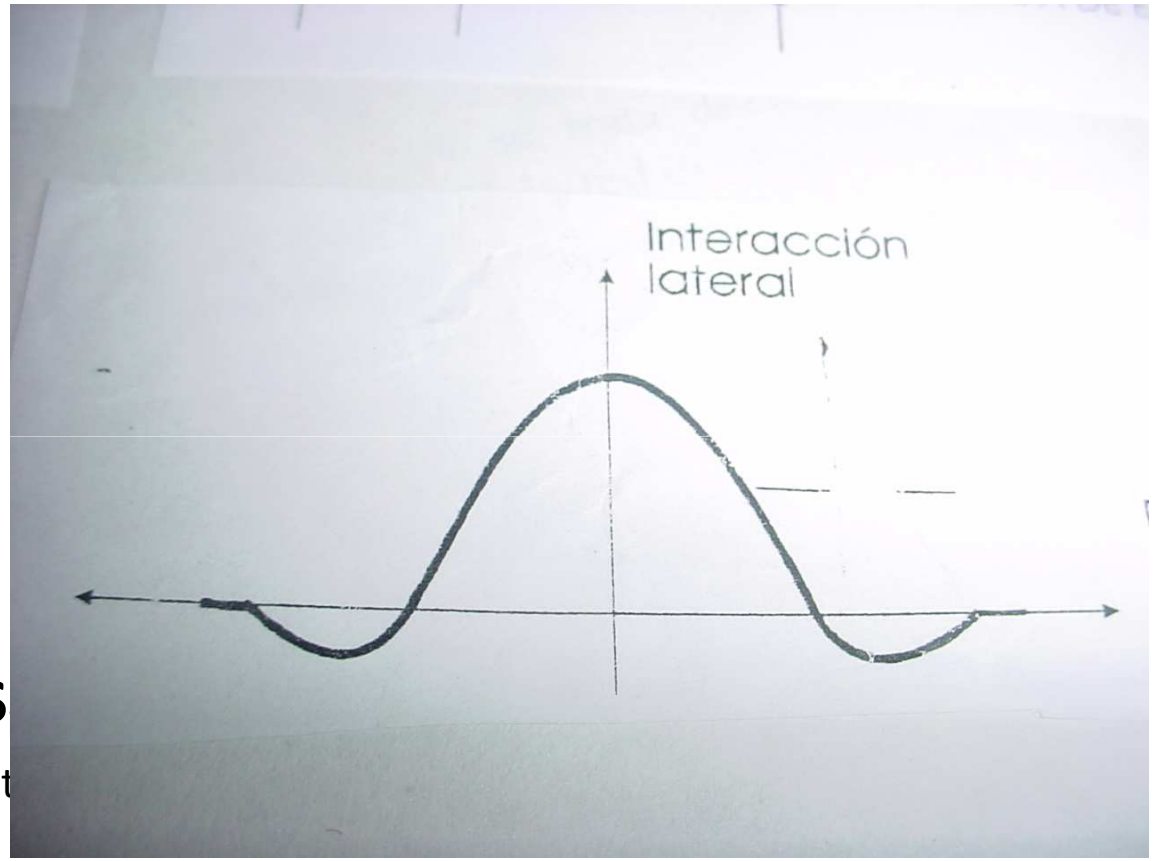
MODELO DE KOHONEN

- ARQUITECTURA
  - 2 CAPAS CON CONEXIONES HACIA ADELANTE



MODELO DE KOHONEN

funcionamiento



S

Int

MODELO DE KOHONEN

- APRENDIZAJE:
  - FUERA DE LINEA
  - NO SUPERVISADO COMPETITIVO
  - REPETIR VARIAS VECES => MEJOR APRENDE

MODELO DE KOHONEN

- PROCEDIMIENTO:

  1. PRESENTAR ENTRADA
  2. DETERMINAR NEURONA VENCEDORA

$$d_i=\Sigma_i^N (e_i^k - w_{ji})^2 \qquad 1<=j<=M$$

  3. ACTUALIZAR PESOS ENTRE NEURONA j* VENCEDORA CON SUS VECINOS Y ENTRADAS

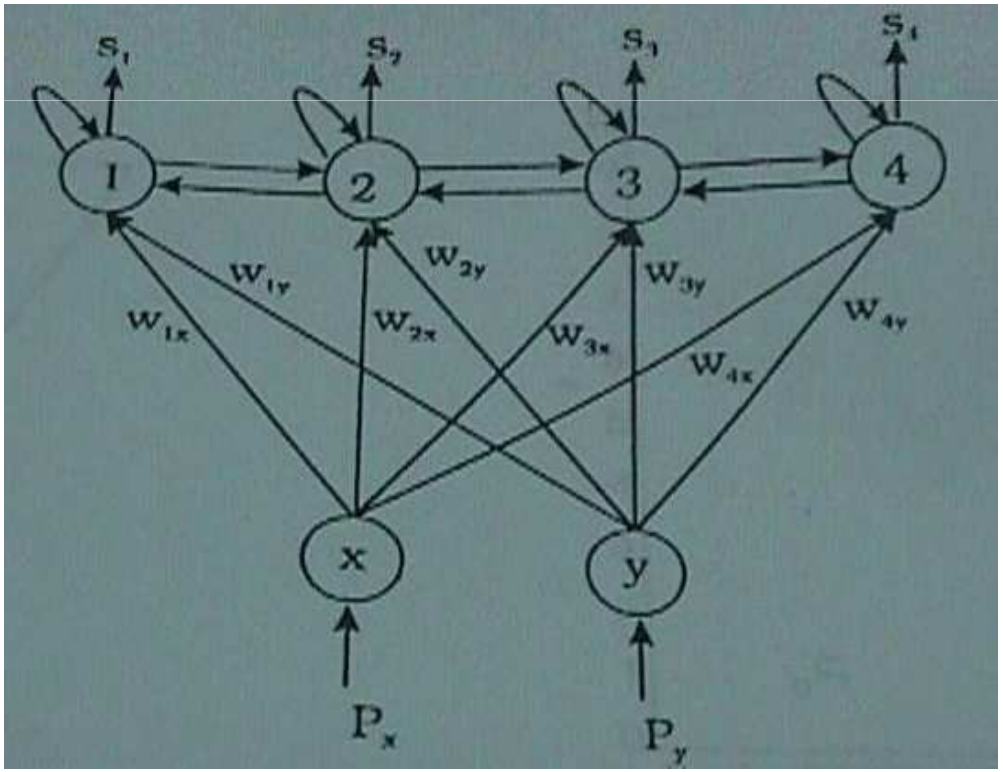
$$w_{ji}(t+1)= w_{ji}(t)+ \alpha(t)[(e_i^k - w_{j*i}(t)] \quad \text{para } j \in Zona_{j*}(t)$$

  4. REPETIR PROCESOS PARA TODOS LOS PATRONES 500 VECES

MODELO DE KOHONEN

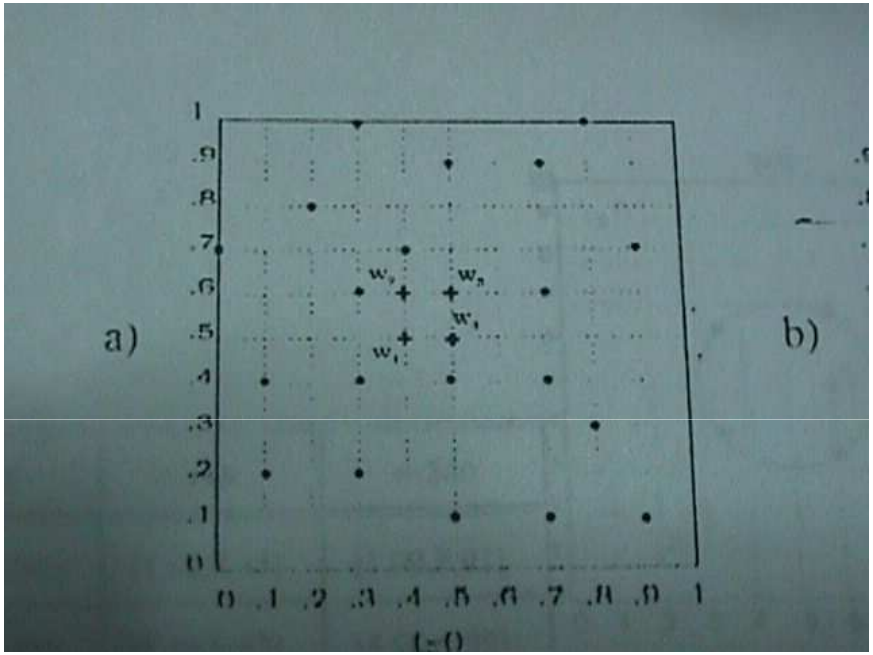
APLICACIÓN:CLASIFICACION

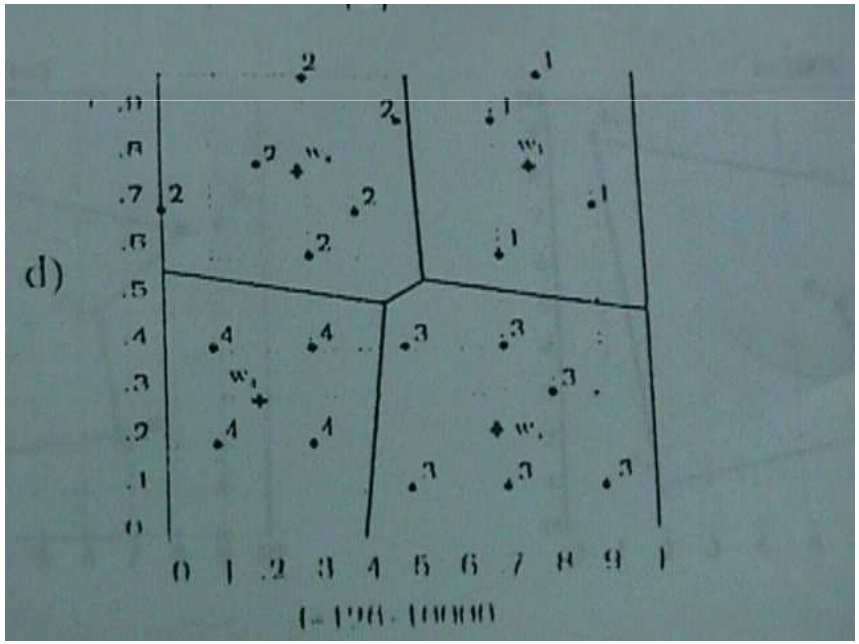
- RECONOCIMIENTO DE VOZ
- CODIFICACION DE DATOS
- OPTIMIZACION



MODELO DE KOHONEN

APLICACIÓN:CLASIFICACION

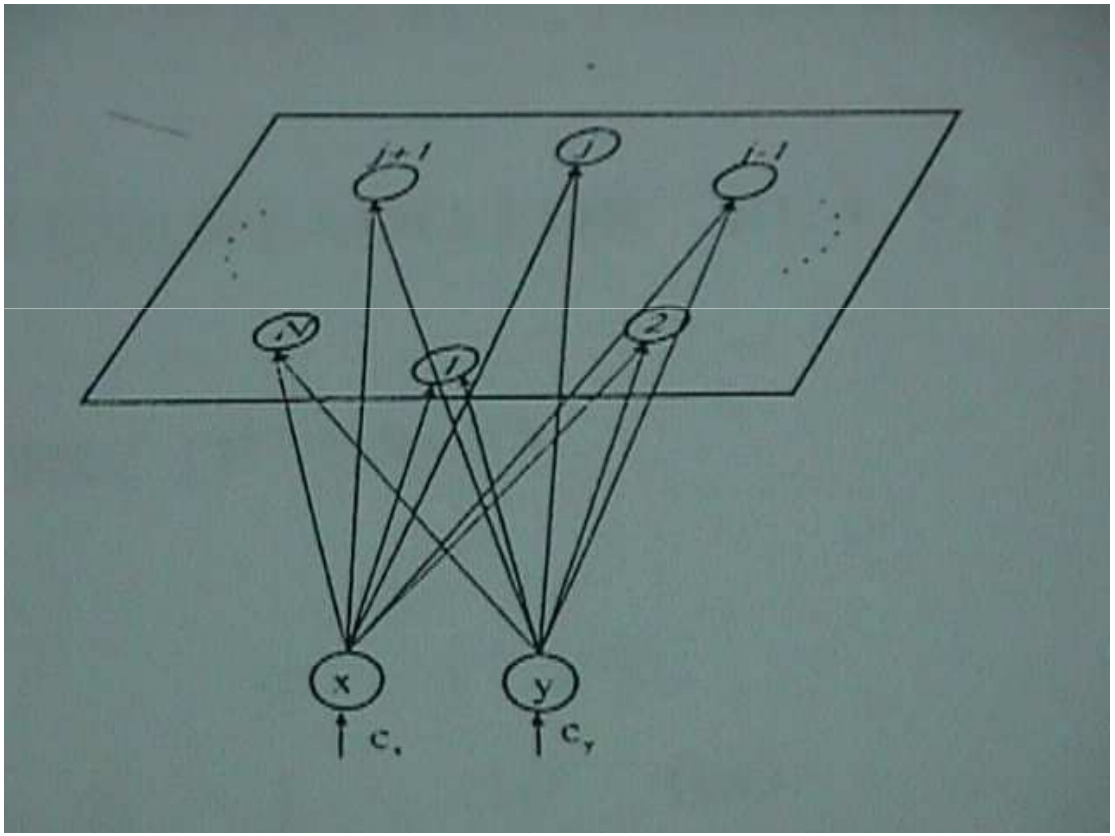




MODELO DE KOHONEN

APLICACIÓN: OPTIMIZACION

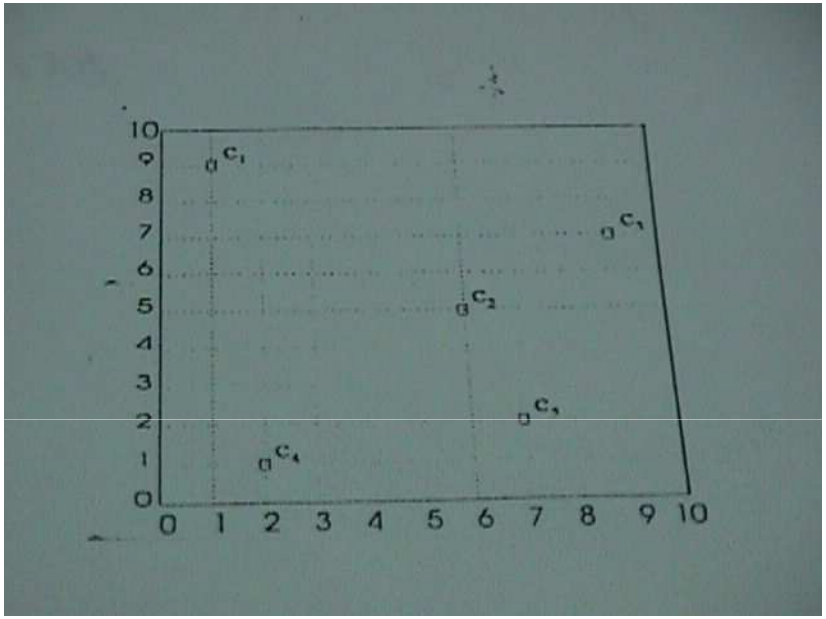
- VIAJERO DE COMERCIO



MODELO DE KOHONEN

APLICACIÓN: OPTIMIZACION

- VIAJERO DE COMERCIO

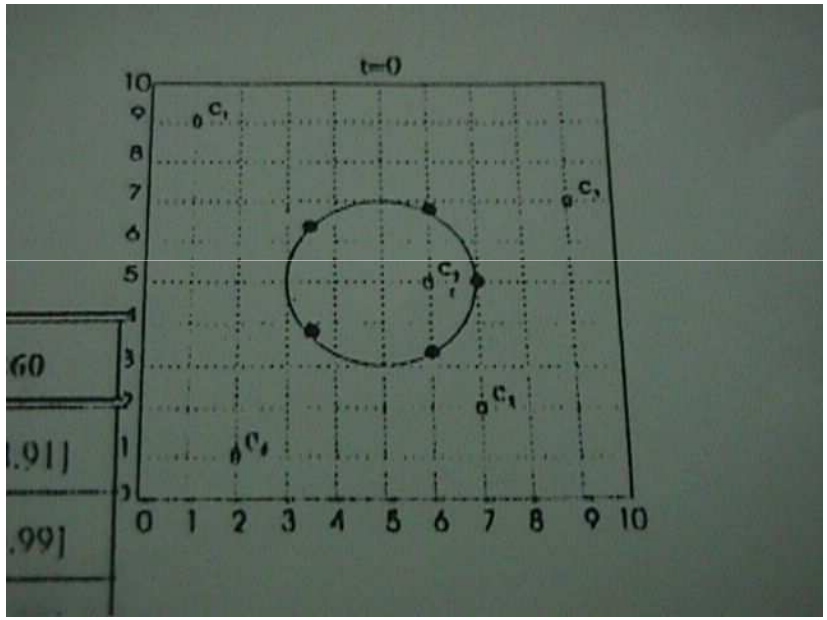


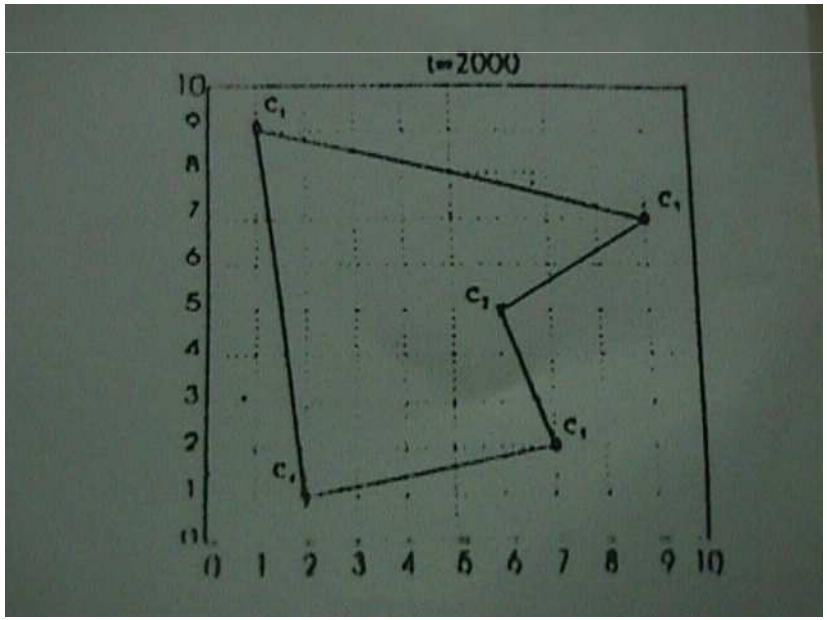
W_j	t=0	t=1	t=2	t=5	t=260
W_1	[3.5,6.32]	[2.25,7.66]	[1.94,7.99]	[1.62,8.34]	[1.09,8.91]
W_2	[6.0,6.73]	[7.50,6.86]	[7.88,6.90]	[8.26,6.93]	[8.90,6.99]
W_3	[7.0,5.00]	[6.50,5.00]	[6.38,5.00]	[6.25,5.00]	[6.03,5.00]
W_4	[6.0,3.27]	[6.50,2.63]	[6.62,2.48]	[6.75,2.31]	[6.97,2.04]
W_5	[3.5,3.68]	[2.75,2.34]	[2.56,2.01]	[2.37,1.66]	[2.05,1.09]

MODELO DE KOHONEN

APLICACIÓN: OPTIMIZACION

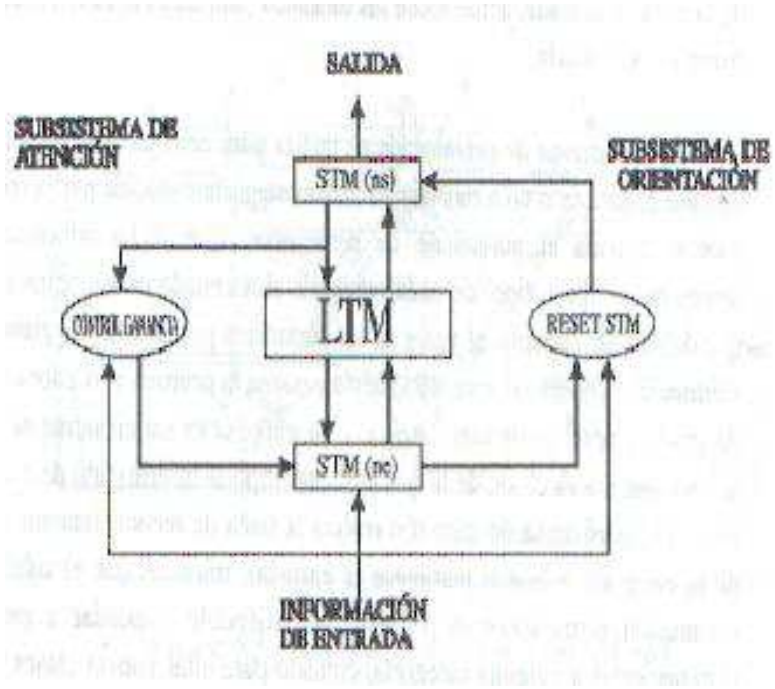
- VIAJERO DE COMERCIO





Modelo de la Teoría de Resonancia Adaptativa (ART)

- Fue propuesto por Grossberg y Carpenter, en 1986.
- Este modelo se basa en el dilema de la estabilidad y la plasticidad del aprendizaje.
- Se aplica a sistemas competitivos en los cuales cuando se presenta cierta información de entrada sólo una de las neuronas de salida de la red se activa.
- La teoría de la resonancia adaptativa se basa en la idea de hacer resonar la información de entrada con los representantes o prototipos de las categorías que reconoce la red.



ART

DILEMA DE ESTABILIDAD Y PLASTICIDAD DEL APRENDIZAJE
=> CONTRA EL OLVIDO

COMO RNA APRENDE NUEVOS PATRONES (PLAST.)

COMO RNA RETIENE PATRONES YA APRENDIDOS (ESTAB.)

OBJETIVO: CLASIFICAR DATOS

APRENDIZAJE NO SUPERVISADO, COMPETITIVO
(CORRELACION ENTRE LOS DATOS DE ENTRADA)

RESONAR INFORMACION ENTRADA CON CLASES QUE
RECONOCE RNA

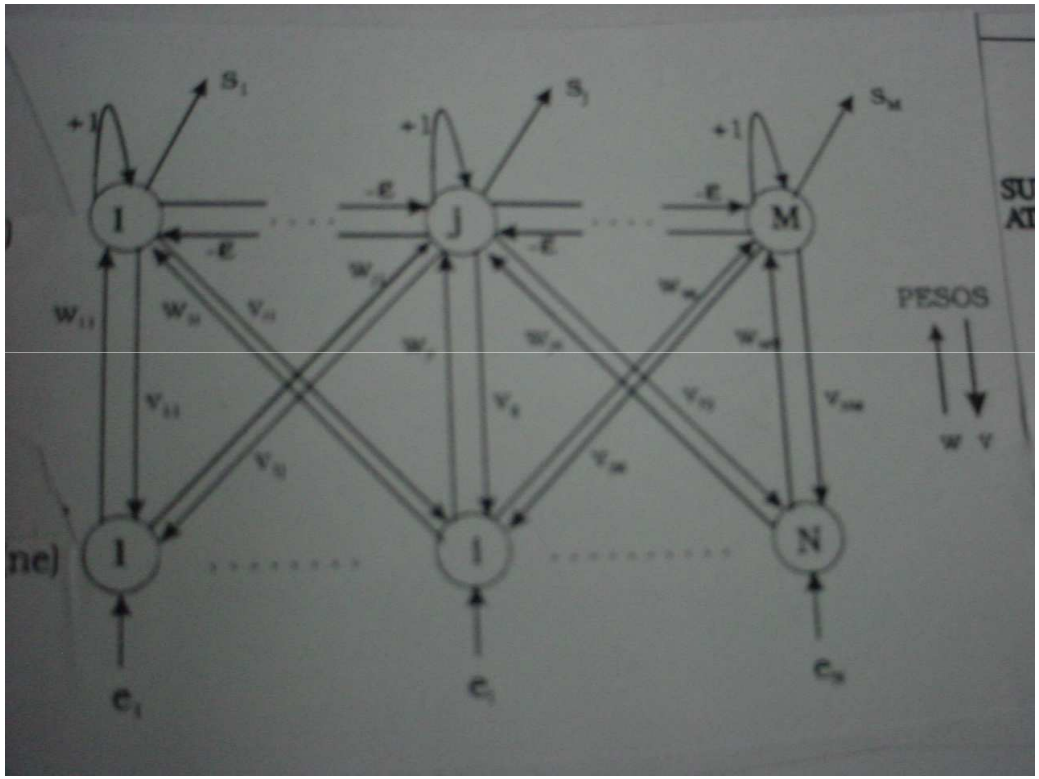
RESONANCIA => PERTENECE A CLASE CONOCIDA

NO RESONANCIA => NUEVA CLASE

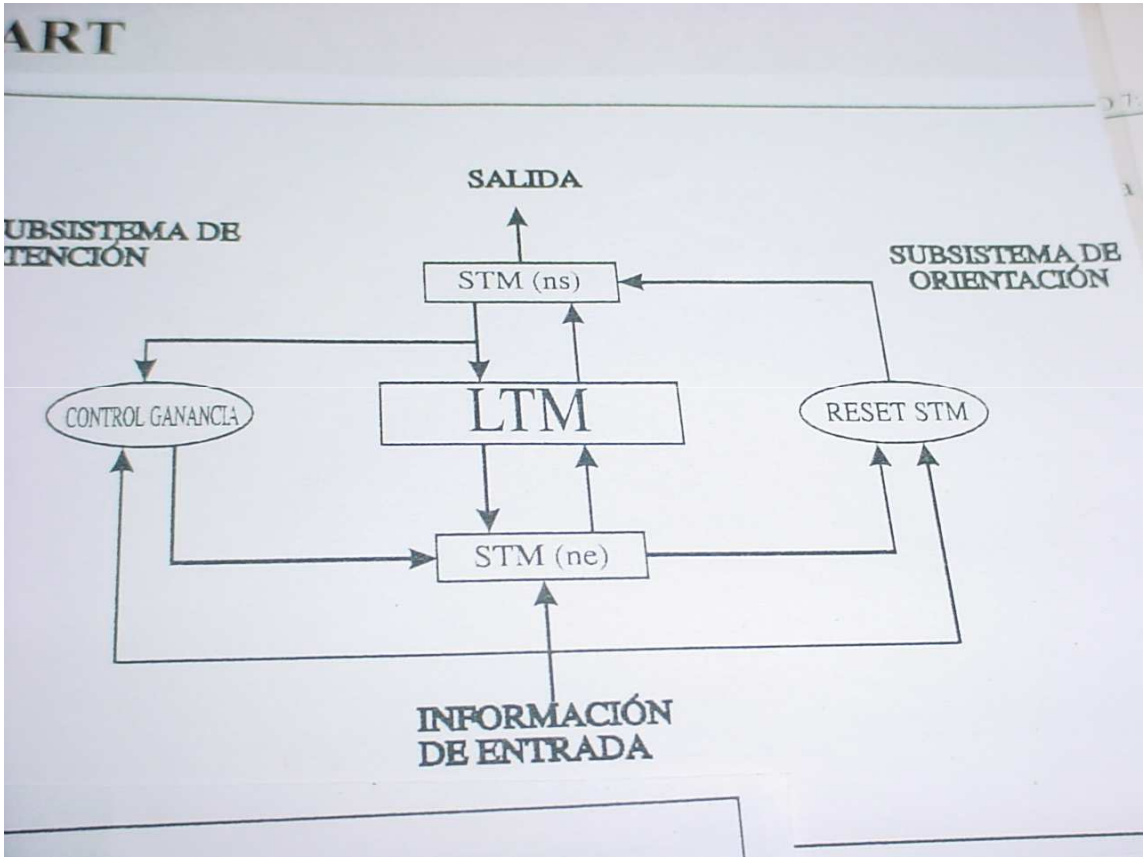
ART (GROSSBERG 1980)

ARQUITECTURA

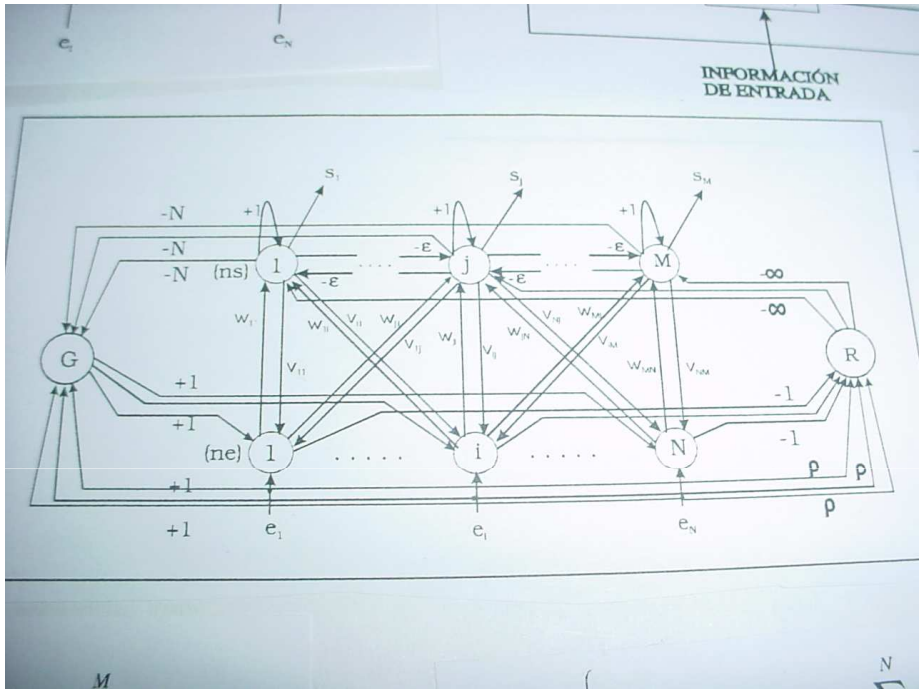
- 2 CAPAS CON CONEXIONES HACIA DELANTE Y ATRÁS
- CAPA SALIDA CON CONEXIONES LATERALES
- PESO CAPA SALIDA ES FIJO
  - +1: RECURRENTE
  - e: LATERALES



ART (GROSSBERG 1980)



ART (GROSSBERG 1980)



$s_G = \{ 1 \quad \Sigma_i^N e_i^k - N\Sigma_i^M s_{nsj} >= 0.5 \qquad 0 \quad \Sigma_i^N e_i^k - N\Sigma_i^M s_{nsj} < 0.5$

$s_R = \{ \text{Reset } ns_j \quad \Sigma_i^N \rho e_i^k - N\Sigma_i^N s_{nei} > 0 \qquad 0 \quad \Sigma_i^N \rho e_i^k - N\Sigma_i^N s_{nei} <= 0$

$net_{nei} = e_i^k + \Sigma_j^M v_{ij} s_{nsj} + s_G \qquad s_{nsi} = \quad 1 \quad net_{nei} >= 1.5 \qquad 0 \quad net_{nei} < 1.5$

ART

- FUNCIONAMIENTO:

  1. VECTOR DE ENTRADA

  2. NEURONAS CAPA ENTRADA ENVIAN $\mathbf{e_i^k}$ A NEURONAS CAPA SALIDA

  3. NEURONAS CAPA SALIDA COMPITEN

$$\mathbf{s_{nsj}(t+1)=f[s_{nsj}(t)- \varepsilon \Sigma_p^M s_{nsp}(t)+ \Sigma_i^N w_{ji} s_{nei}(t)]}$$

  f: FUNCION ESCALON

  SE ITERA HASTA QUE ESTABILICE VALOR SALIDA DE LA RED

  GANADORA $\mathbf{s_{nsj}=}$ **1** $\mathbf{MAX(\Sigma_i^N w_{ji} e_i^k)}$

  0 **resto**

ART

4. NEURONA VENCEDORA (ns_{j*}) ENVIA SU SALIDA HACIA ATRAS

$$\mathbf{x_i=\sum_j^M v_{ij}\, s_{nsj}= v_{ij*}} \quad \textbf{donde} \quad \mathbf{s_{nsj}=} \begin{cases} \mathbf{1} & \mathbf{j=j^*} \\ \mathbf{0} & \mathbf{j\neq j^*} \end{cases}$$

5. COMPARAR INFLUENCIA ENTRADA CON INFORMACION REALIMENTADA

=> RELACION SEMEJANZA (RS)=||EkX||/||Ek||

ART

6. COMPARAR RS CON PARAMETRO VIGILANCIA (P)

SI RS<P ENTONCES

NEURONA VENCEDORA NO PERTENECE A CLASE

RESET NEURONA VENCEDORA

SI QUEDAN NEURONAS SIN RESET

REPETIR DESDE 2. SIN NEURONA

VENCEDORA Y MISMA ENTRADA

DE LO CONTRARIO

NEURONA VENCEDORA ES CLASE APROPIADA

AJUSTAR PESOS

ART

APRENDIZAJE:

NO SUPERVISADO COMPETITIVO

LENTO=> ENTRADA ASOCIADA A CLASE

RAPIDO=>NUEVA CATEGORIA

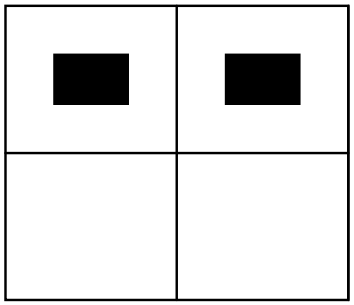
$$v_{ij*}(t+1) = v_{ij*}(t) e_i^k \qquad w_{j*i}(t+1) = \frac{v_{ij}*(t)\, e_i^k}{\gamma + \sum_i^N v_{ij}*(t)\, e_i^k}$$

ART

APLICACIÓN:

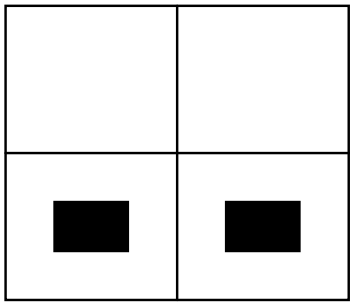
- RECONOCIMIENTO DE IMÁGENES
- RECONOCIMIENTO DE SENALES ANALOGICAS

Figura 1



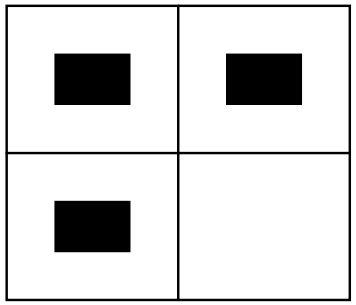
E1={1,1,0,0}

Figura 2



E2={0, 0,1,1}

Figura 3



E3={1,1,1,0}

ART

E1= V= 1 1 0 0 W= 0.2 0.2

1 1 1 1 0.4 0.2

0 0.2

0 0.2

E2= $\sum_i^4 w_{1i} e_i^2 = 0$ $\sum_i^4 w_{2i} e_i^2 = 0.4$

E3= $\sum_i^4 w_{1i} e_i^3 = 0.8$ $\sum_i^4 w_{2i} e_i^3 = 0.4$

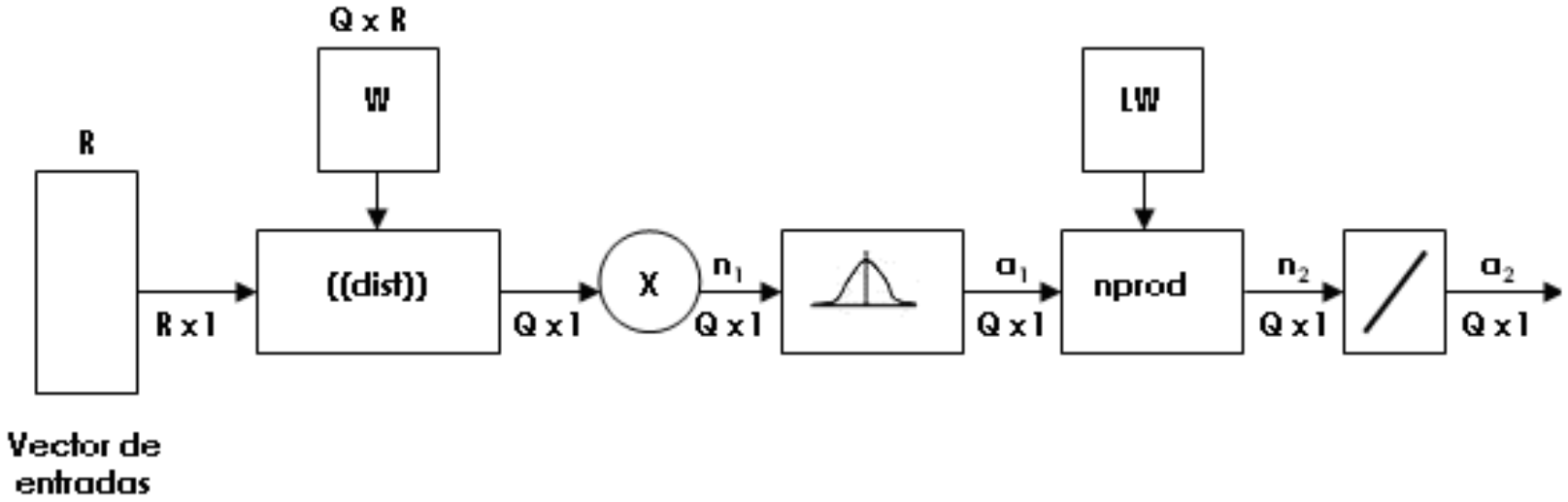
RELACION SEMEJANZA: ||E3V1||/||E3||=

$$(v_{11} e_1^3 + \ldots + v_{41} e_4^3)/(e_1^3 + \ldots + e_4^3) = 0.6$$

TIPOS DE RNA

Redes de Función de Base Radial (RBF). Está formada por tres capas: la de entrada, la oculta y la de salida. La capa oculta consta de una función de base radial como función de activación que frecuentemente es la función Gaussiana y el entrenamiento de las neuronas es no supervisado; mientras que la capa de salida se rige por una función lineal y el entrenamiento es supervisado para las neuronas de esta capa. Las RBF se caracterizan por su conexión hacia delante.

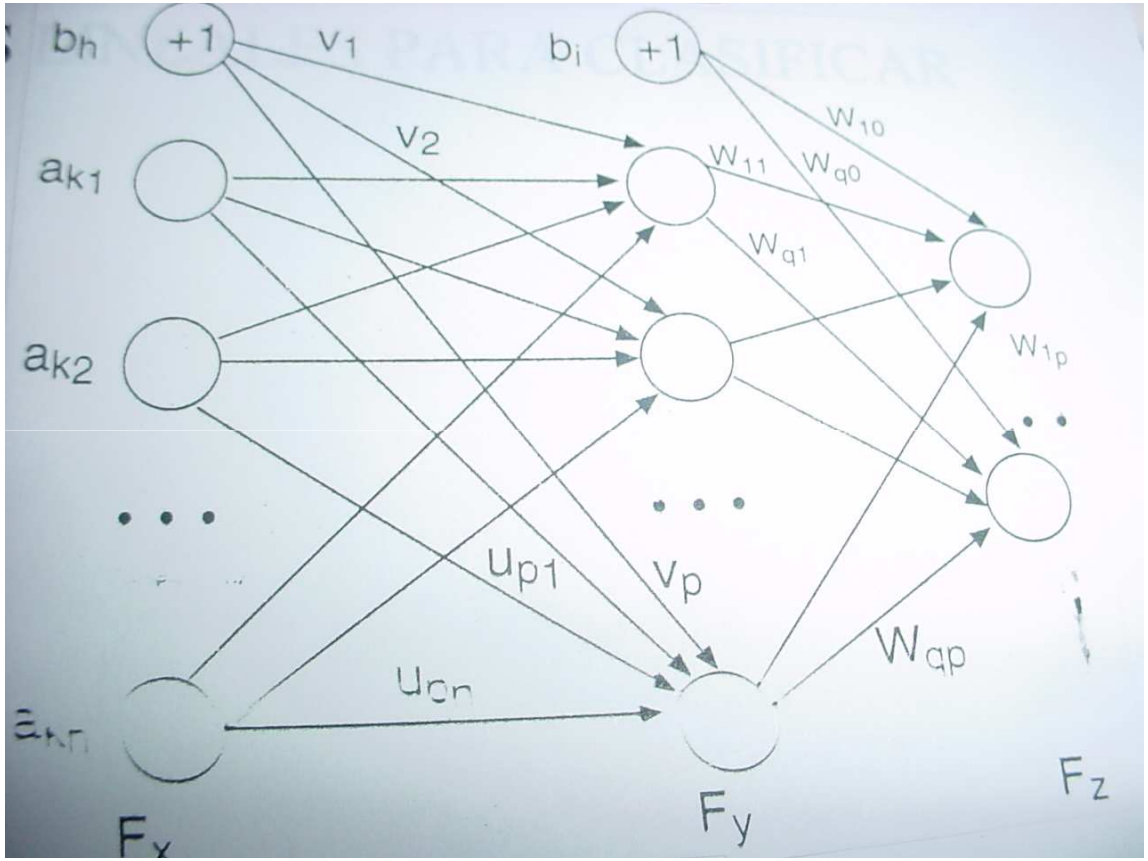
Algunas de las aplicaciones de las redes de función de base radial son: Predicción de series de tiempo, Aproximación de funciones, Control, Problemas de clasificación, Reconocimiento de patrones, Procesamiento del le



RBF

- RADIAL BASIS FUNCTION
- COMBINACION BACKPROPAGATION Y KOHONEN
- TAREAS DE CLASIFICACION
  - ENCONTRAR CENTROIDE DE GRUPOS DE DATOS
  - USAR CENTROIDE COMO FUNCION DE DENSIDAD
  - FORMAR COMBINACIONES

RBF


bh
+1
v1
bi
+1
v2
w10
ak1
w11
wq0
wq1
ak2
w1p
up1
vp
wqp
ukn
akn
Fx
Fy
Fz

RBF

- FUNCION DE ACTIVACION CAPAS OCULTAS

$$y_{ki}=\exp\left(-\sum_h (u_{ih}-a_{ih})^2 / v_i^2\right)$$

RADIAL BASIS FUNCTION

- FUNCION ACTIVACION CAPA SALIDA

$$z_{kj}=\sum_i^p w_{ji} y_{kj}$$

RBF

- ENTRENAMIENTO
  - SUPERVISADO
  - PAR ENTRADA-SALIDA
  - ERROR

$$\mathbf{E_j}=\Sigma_k^m(z_{kj}-b_{kj})^2$$

- ACTUALIZACION DE PESOS
  - CAPA OCULTA

$$\mathbf{u_{ih}}=\mathbf{u_{ih}}+\eta\,(\mathbf{u_{ih}}-a_{kh}) \text{ para } \max(\mathbf{y_{kj}}) \text{ solamente}$$

  - CAPA SALIDA

$$\mathbf{w_{ji}}=\mathbf{w_{ji}}+\eta\,\Sigma_k\,(b_{\mathbf{kj}}-z_{kj})\mathbf{y_i}$$

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

- EMISION OCURRE ALEATORIAMENTE SEGÚN UNA DISTRIBUCION EXPONENCIAL

- SENALES SE ENVIAN A OTRAS NEURONAS O AL EXTERIOR

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

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,

Random Neural Model

- CADA VEZ QUE UNA NEURONA i EMITE, SE RESETEA POTENCIAL, Y SALE UNA SENAL DE EL COMO SENAL POSITIVA A NEURONA j 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

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

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

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

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

- SI UNA SOLUCION EXISTE, ENTONCES LA DISTRIBUCION DE PROBABILIDADES ESTACIONARIA ES

$$p(k) = \Sigma^{n}_{i=1} (1-q(i))q(i)^{k(i)}$$

k(t): vector of signal potentials at time *t*.

- PARA GARANTIZAR ESTABILIDAD

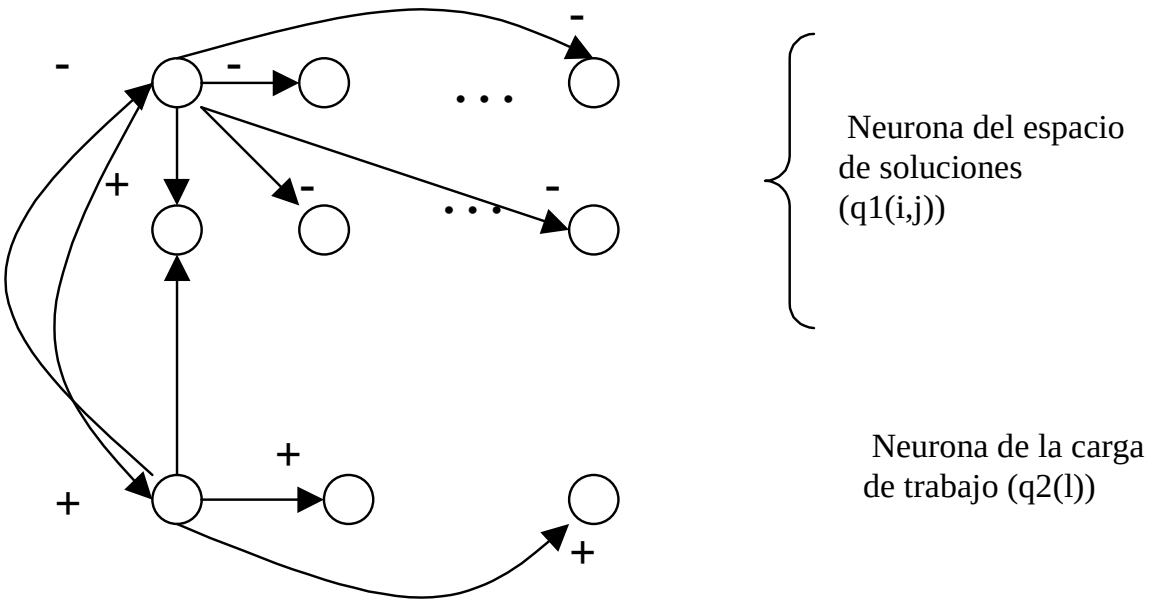
$$\lambda^{+}(i) =< [\, r(i) + \lambda^{-}(i)]$$

Random Neural Model

APLICACIONES

- Partición de Grafos

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



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

- Exogena positiva senal de clase *c* llega a neurona *i* a una tasa de Poisson $\Lambda(i, c)$,
- Neurona *i* emite senales excitatorias de clase *c* a la tasa $r(i, c)>0$, con probabilidad Kic/Ki a la neurona *j* a su clase φ positiva con probabilidad

$$r(i,c)[Kic/Ki]p^{+}(i, c; j, \varphi)$$

- o senal negativa con probabilidad

$$r(i,c)[Kic/Ki]p^{-}(i, c; j)$$

Multiple Classes Random Network Model

- Probabilidad que neurona se “pierda”

$$r(i,c)[Kic/Ki]d(i, c).$$

- Claramente:

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

- Probabilidad excitacion 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

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

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***
  - $Y_k(i, c)>0 \Rightarrow X_k(i, c) = (\Lambda_k(i, c), \lambda_k(i)) = (Lic, 0)$
  - $Y_{ik}(i, c)=0 \Rightarrow (\Lambda_k(i, c), \lambda_k(i)) = (0, 0)$
  - 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.*

Multiple Classes Random Network Model

- ***Procedimiento de Recuperación***
  - progresivo proceso de recuperación con umbral adaptativo

    X'= {X'(1, 1), ..., X'(n, C)} vector de entrada
  - Para calcular Y= {Y(1,1), ..., Y(n, C)},

1. Calcular Q=(q(1, 1), ..., q(n, C)).
2. Intervalo de incertitud Z los q(i, c) en $1\text{-}T<q(i, c)<T/2$ o $1\text{-}T/2<q(i, c)<T$, con T=0.8,
3. Estabilidad de la red => q(i, c)$\in$ Z = 0.

Multiple Classes Random Network Model

- ***Procedimiento de Recuperación***

4. Tratar neuronas con problemas:

$$Y^{(1)}_{(i,\,c)} = Fz(q(i,\,c)) = \begin{cases} 1 & \text{if } q(i,\,c) > T \\ 0 & \text{if } q(i,\,c) < 1-T \\ 0.5 & \text{if } T/2 <= q(i,\,c) <= 1-T/2 \\ x'_i & \text{otherwise} \end{cases}$$

5. Cuando $q(i,\,c) \in Z = 0 \Rightarrow Y = Y^{(1)}$. De lo contrario, Y is obtenido al aplicar la funcion umbral :

$$Y(i,\,c) = f_{\beta}(q(i,\,c)) = \begin{cases} 1 & \text{if } q(i,\,c) > \beta \\ 0.5 & \text{if } \beta/2 < q(i,\,c) < \beta \\ 0 & \text{otherwise} \end{cases}$$

ß:umbral

Multiple Classes Random Network Model

- ***Procedimiento de Recuperación***

Cada valor $q(i, c) \in Z$ es un potencial umbral.

$$\beta = \begin{cases} q(i, c) & \text{if } q(i, c) > 0.666 \\ 1-q(i, c) & \text{otherwise} \end{cases}$$

Multiple Classes Random Network Model

- ***Procedimiento de Recuperación***

6. Por cada potencial β, a la RNA se le da $X'^{(1)}(\beta) = f_\beta(Q)$.
7. Calcular $Q^{(1)}(\beta)$ y $Y^{(2)}(\beta) = Fz(Q^{(1)}(\beta))$.
8. Guardar casos $q(i, c) \in Z = 0$ y $X'^{(1)}(\beta) = Y^{(2)}(\beta)$.
9. Si nunca se satisfasen esas 2 condiciones
    - Inicial X' is considerado diferente a los entrenados

de lo contrario

- Si hay varios candidatos se escoge el que minimize:

$$E(\beta) = 1/2 \, \Sigma^{n}_{i=1} [q(i, c)^{(1)}(\beta) - Y(i, c)^{(1)}(\alpha)]^2$$

OTRAS REDES NEURONALES ESTOCASTICAS

- **FUNCION DE ACTIVACION NO DETERMINISTA**

  SALIDA PROBABILISTICA

- **MECANISMO DE APRENDIZAJE**

  PESOS SE MODIFICAN ALEATORIAMENTE Y SE COMPRUEBA SU EFECTO EN LA RED

- **MINIMIZA FUNCION DE ENERGIA**
  - ANALOGIA CON SISTEMA TERMODINAMICO
  - TECNICA DE ENFRIAMIENTO “TEMPLE SIMULADO

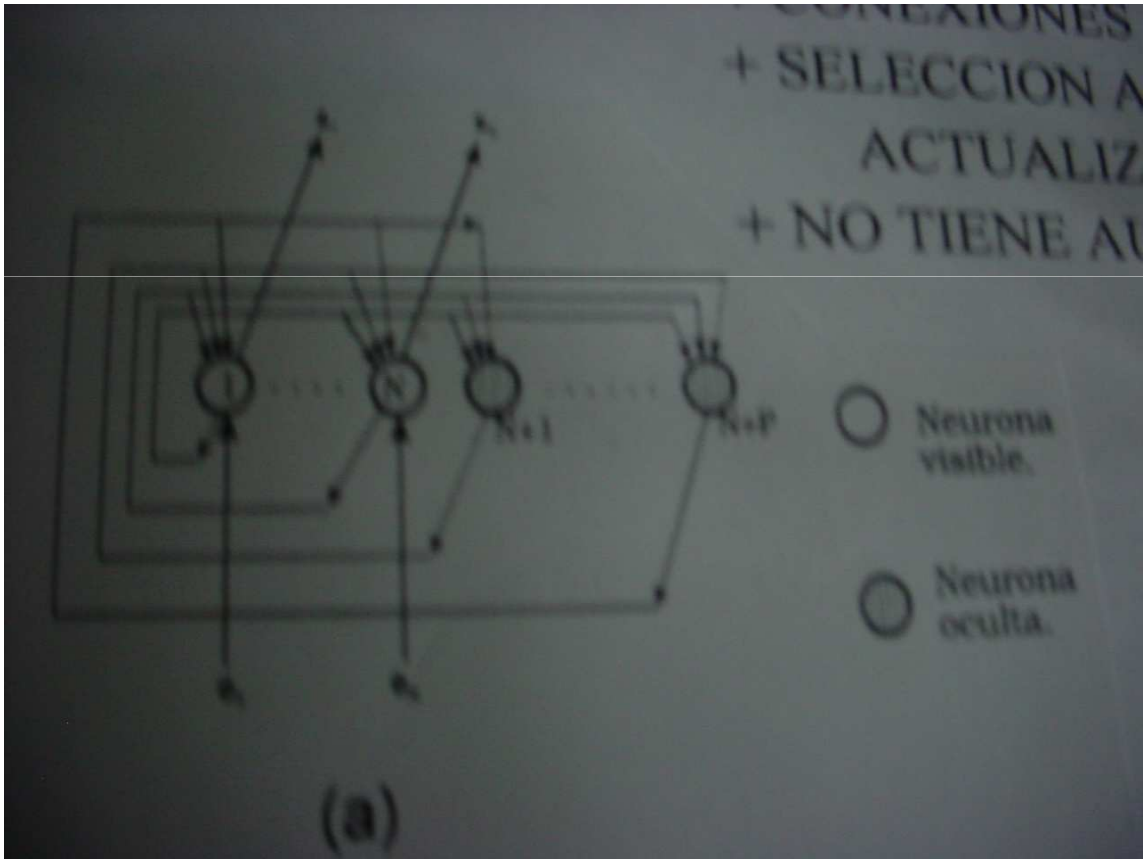
MAQUINA DE BOLTZMAN

- **ARQUITECTURA:**
  - NEURONA BINARIA
  - CONEXIONES SIMETRICAS
  - SELECCIÓN ALEATORIA NEURONA A ACTUALIZAR SU PESO
  - NO TIENE AUTORECURRENCIA

- **FUNCIONAMIENTO Y APRENDIZAJE**
  - BASADO EN TEMPLE SIMULADO

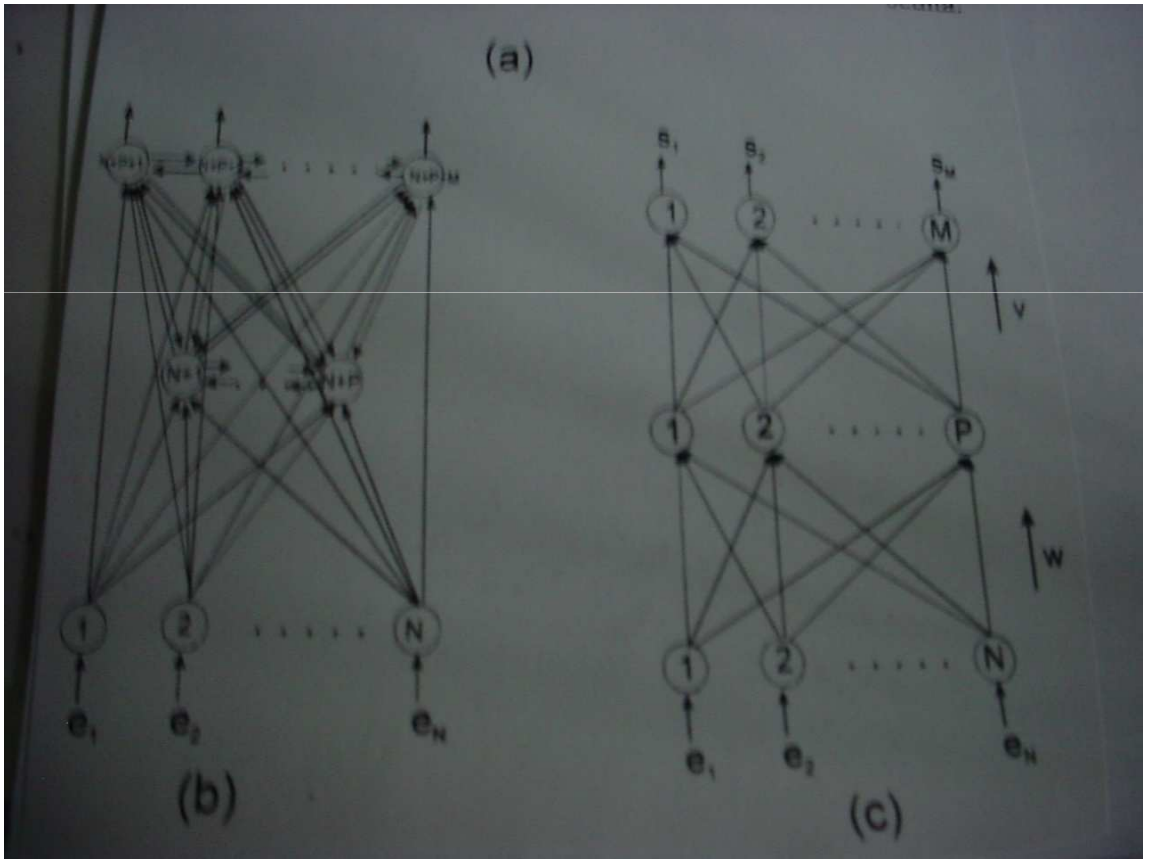
MAQUINA DE BOLTZMAN

- **ARQUITECTURA**



MAQUINA DE BOLTZMAN

- **ARQUITECTURA**



MAQUINA DE BOLTZMAN

- PROBABILIDAD DE ACTIVACION DE LA SALIDA

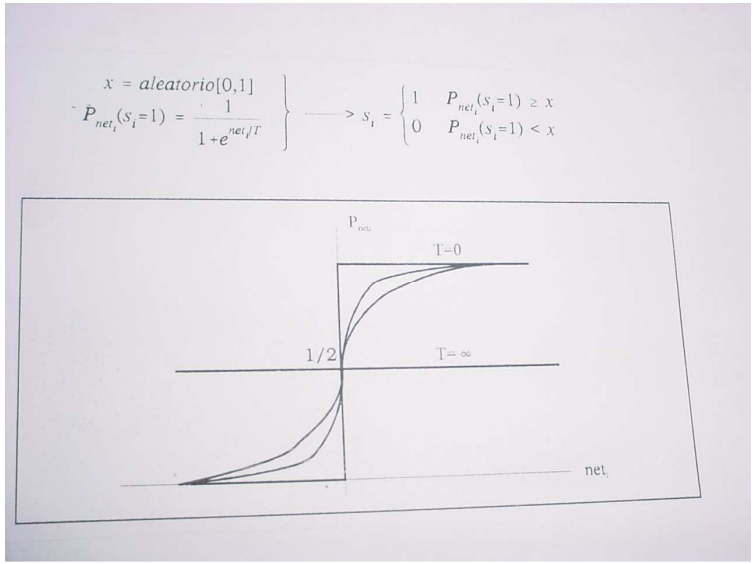
$$P_{neti}(s_i=1)=1/(1+e(^{neti/T}))$$

T:temperatura

- CUANDO ESTA ACTIVA?

x=aleatorio [0, 1] s_i= { 1 $P_{neti}(s_i=1)>=x$

0 $P_{neti}(s_i=1)<x$

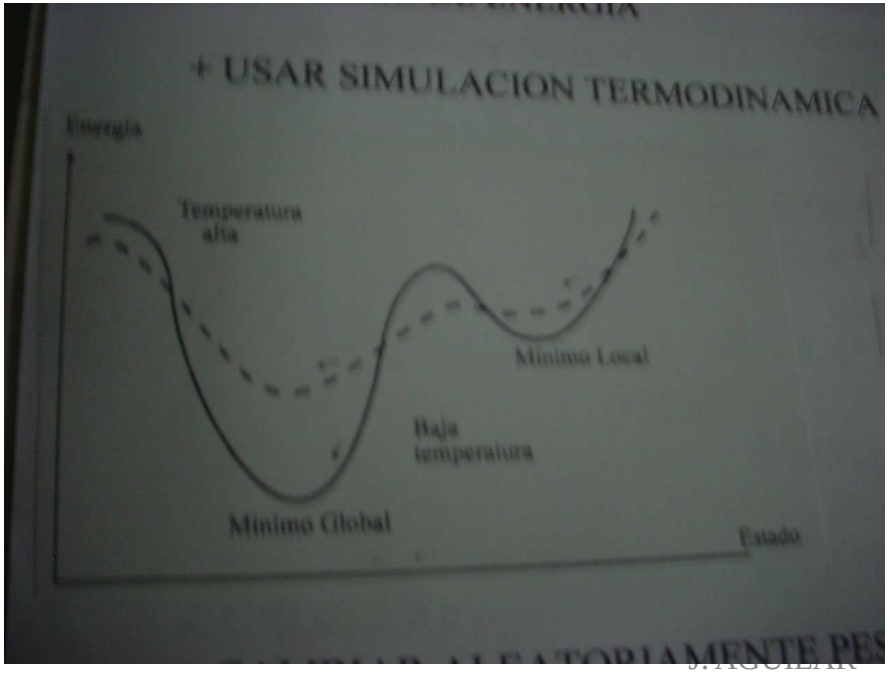


MAQUINA DE BOLTZMAN

- PROCEDIMIENTO
  - AJUSTAR VALOR T INICIAL
  - INICIALIZAR SALIDA DE TODAS LAS NEURONAS
    - NEURONAS OCULTAS: [0,1]
    - RED 1 CAPA: NEURONAS VISIBLES CON LOS VALORES DEL VECTOR DE ENTRADA (AUTO)
    - ENTRADA-SALIDA:SALIDA ALEATORIA [0,1], ENTRADA CON VECTOR DE ENTRADA (HETERO)
  - SELECCIONAR UNA NEURONA ALEATORIAMENTE Y CALCULAR SU SALIDA (NEURONA≠CAPA ENTRADA)
  - REPETIR PASO ANTERIOR N VECES (N: NUMERO NEURONAS)
  - INCREMENTAR TIEMPO Y SE REPITE PROCESO

MAQUINA DE BOLTZMAN

- APRENDIZAJE
  - LOGRAR QUE LA RED SE ESTABILICE EN UN VALOR MINIMO DE ENERGIA
  - USAR SIMULACION TERMODINAMICA



$$E = -1/2 \, \Sigma_i \, \Sigma_j \, w_{ij} s_j s_i$$

$$Prob(s_i \text{->} -s_i) = 1/(1+\exp(-\Delta E_i/T))$$

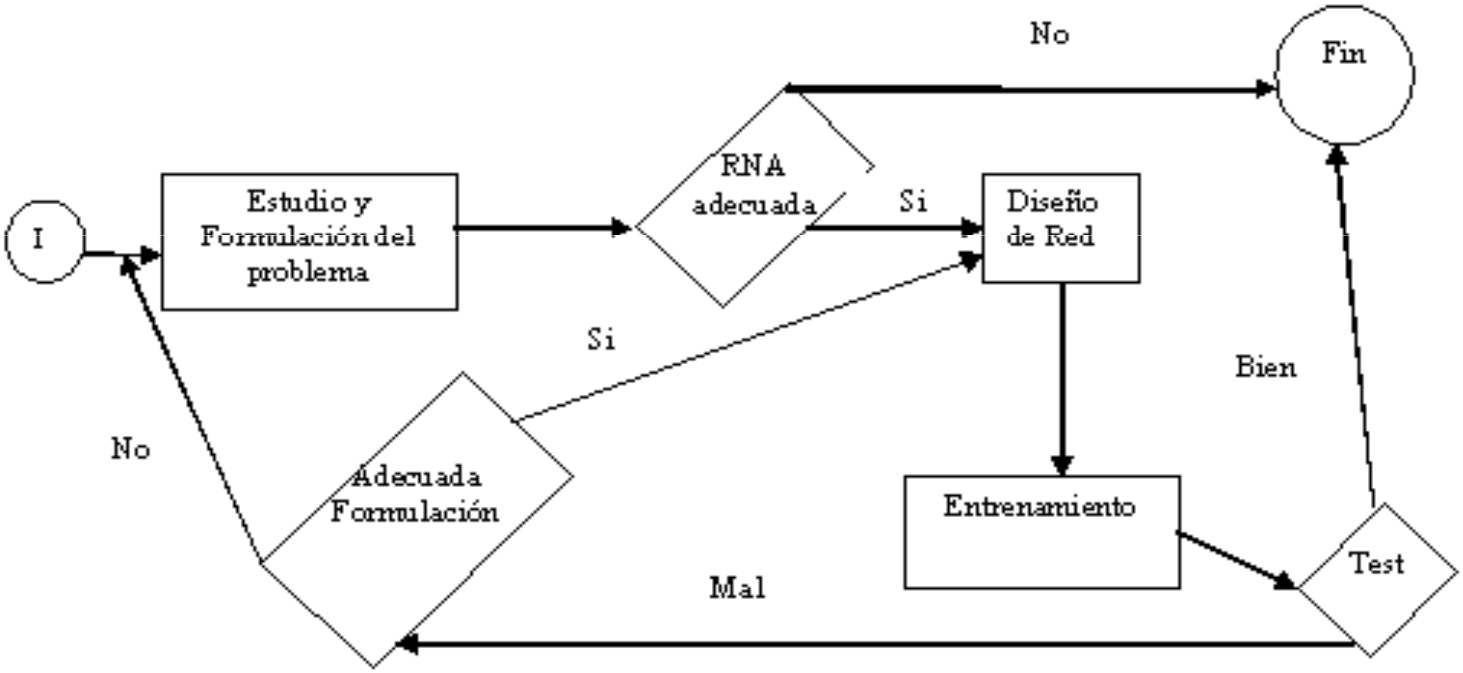
MAQUINA DE BOLTZMAN

- APRENDIZAJE
  - CAMBIAR ALEATORIAMENTE PESOS DE LAS CONEXIONES Y COMPROBAR SU EFECTO
    - ACEPTAR CAMBIOS QUE DISMINUYEN FUNCION DE ENERGIA
    - ACEPTAR CAMBIOS QUE AUMENTEN FUNCION DE ENERGIA, SEGÚN CIERTA PROBABILIDAD
    - A MEDIDA QUE AVANZA PROCESO DE APRENDIZAJE, DECRECE PROBABILIDAD DE ACEPTAR CAMBIOS QUE AUMENTEN FUNCION DE ENERGIA

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

Cómo aplicar una RNA a un problema?



METODOLOGIA PARA EL DISENO DE APLICACIONES USANDO RNA

- ANALISIS Y DESCRIPCION DEL PROBLEMA
  - DESCRIPCION GENERAL DEL PROBLEMA
  - ANALISIS DE FACTIBILIDAD PARA EL USO DE UNA RNA
  - ANALISIS DE DATOS
- ESPECIFICACION DE REQUERIMIENTOS
  - PERFIL DE LOS USUARIOS FINALES DE LA APLICACIÓN
  - VERIFICACION DE LOS REQUERIMIENTOS
  - DETERMINACION DE LOS REQUERIMIENTOS DE INFORMACION

METODOLOGIA PARA EL DISENO DE APLICACIONES USANDO RNA

- ESPECIFICACION DE REQUERIMIENTOS
  - DETERMINACION DE LOS REQUERIMIENTOS FUNCIONALES
  - DETERMINACION DE LOS REQUERIMIENTOS DE ENTRADA DE DATOS
  - DEFINICION DE LOS REQUERIMIENTOS DE H&S
- ANALISIS DE COSTOS, TIEMPO Y RECURSOS
  - ESTIMACION DEL TIEMPO PARA DISENAR Y ENTRENAR LA RED
  - PLAN DE ACTIVIDADES DE DESARROLLO
  - ESTIMACION DEL RECURSO COMPUTACIONAL

METODOLOGIA PARA EL DISENO DE APLICACIONES USANDO RNA

- DISENO DE LA RNA
  - ANALISIS Y PROCESAMIENTO DE DATOS
  - SELECCIÓN DE LA TOPOLOGIA
- DISENO DEL SISTEMA COMPUTACIONAL QUE UTILIZA LA RNA
  - DISENO DE LA ARQUITECTURA COMPT.
  - SELECCIÓN DE LA HERRAMIENTA
  - DISENO DE LOS PROCESOS DE ADQUISICION YALMACENAMIENTO DE DATOS
  - DISENO DEL PROCESO DE INTERCONEXION

METODOLOGIA PARA EL DISENO DE APLICACIONES USANDO RNA

- IMPLANTACION DEL SISTEMA COMPUTACIONAL QUE UTILIZA LA RNA
  - CONSTRUCCION DEL PROTOTIPO
  - VALIDACION DEL PROTTIPO
  - CONSTRUCCION DEL MODELO OPERACIONAL
  - PRUEBA Y DEPURACION
  - MANTENIMIENTO Y ACTUALIZACION