



**MODELOS COMPUTACIONALES: CUARTO CURSO  
DEL GRADO  
DE ING. INFORMÁTICA EN COMPUTACION**

**MODELOS DE REDES  
NEURONALES**

**César Hervás-Martínez  
Pedro A. Gutiérrez Peña  
Grupo de Investigación AYRNA**

**Departamento de Informática y Análisis  
Numérico  
Universidad de Córdoba  
Campus de Rabanales. Edificio Einstein.  
Email: [chervas@uco.es](mailto:chervas@uco.es)  
[pagutierrez@uco.es](mailto:pagutierrez@uco.es)**

**2021-2022**



# MODELOS DE REDES NEURONALES



- 1) REDES DE FUNCIONES DE BASE RADIAL**
- 2) REDES DE HOPFIELD**



# **REDES NEURONALES DE BASE RADIAL**



## Redes de funciones de Base Radial



- Se dice que una función es de base radial (RBF) si su salida depende de la distancia del vector de entrada a un vector almacenado dado.
- La red neuronal RBF tiene una capa de entrada, una capa oculta y una capa de salida.
- En este tipo de redes RBF, la capa oculta utiliza las neuronas con funciones de activación de tipo RBF
- Las salidas de todas estas neuronas ocultas se combinan linealmente en el nodo de salida de la capa oculta

Estas redes tienen una amplia variedad de aplicaciones tales como:

Aproximación de funciones,

Predicción de series temporales,

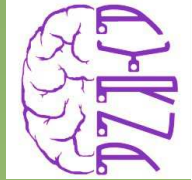
Control y regresión

Tareas de clasificación de patrones en problemas complejos (no lineales).



# FUNCIONES DE BASE RADIAL

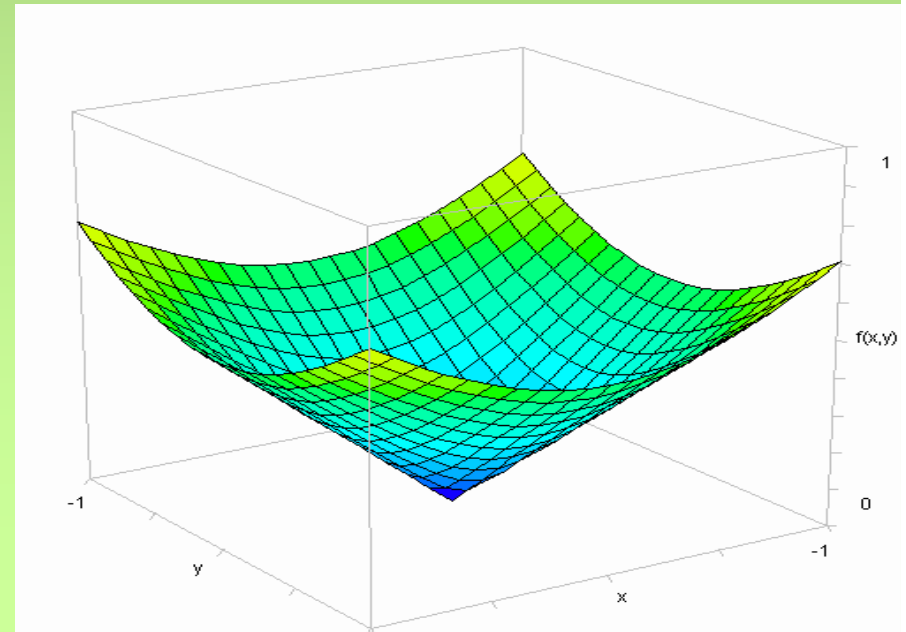
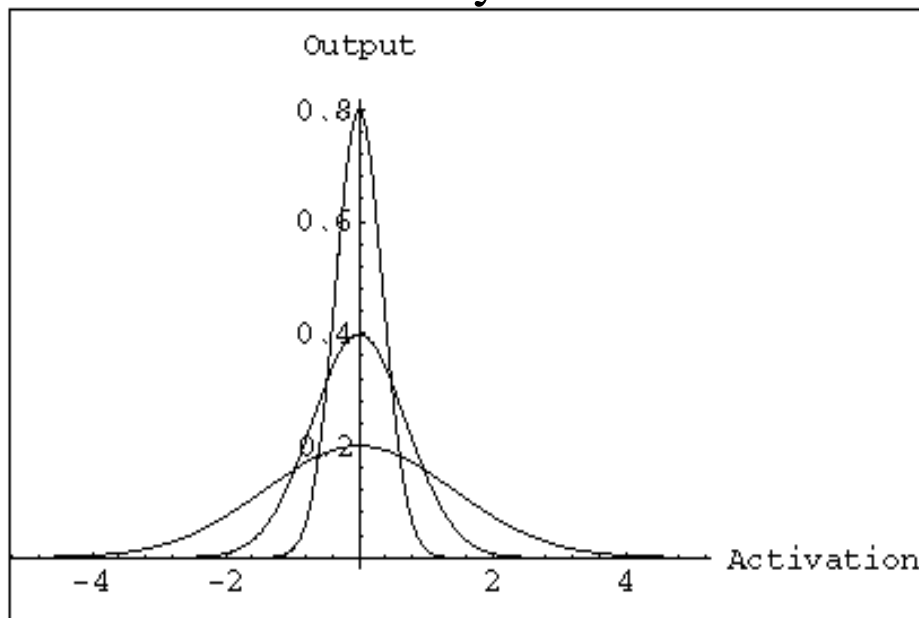
## Características



- ❑ Una sola capa oculta
- ❑ La función de activación de una unidad oculta esta determinada por la distancia entre el vector de entrada y un vector prototipo llamado **centroide**.

$$B_j(\mathbf{x}; (\mathbf{c}_j | r_j)) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2r_j^2}\right), \text{ donde } \mathbf{c} \text{ es el centroide}$$

y  $r$  es el radio del núcleo o "kernel"





# FUNCIONES DE BASE RADIAL



- ❑ La capa oculta de la red RBF tiene unas funciones de base asociadas a los nodos de la capa, cada una con un campo de atracción dado por su centroide.
- ❑ Por lo general esta función es Gaussiana  $B_j$
- ❑ La función de la capa de salida,  $s$ , es lineal si tenemos un problema de regresión o de clasificación con funciones softmax

$$s(\mathbf{x}) = \beta_0 + \sum_{j=1}^K \beta_j B_j \left( \|\mathbf{x} - \mathbf{c}_j\| \right)$$

$$B_j \left( \|\mathbf{x} - \mathbf{c}_j\| \right) = \exp - \frac{1}{2} \left( \frac{\|\mathbf{x} - \mathbf{c}_j\|}{r_j} \right)^2$$



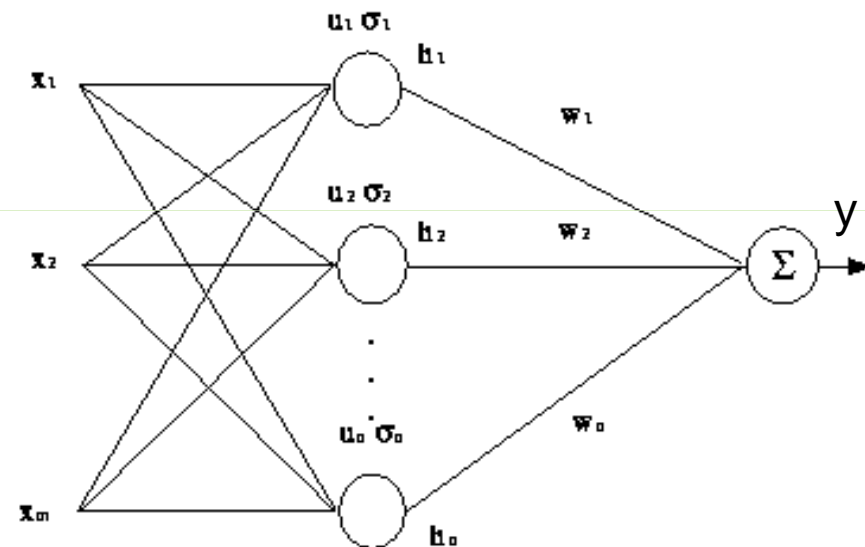
## Redes de funciones de Base Radial



### Radial Basis Function Network

- ❑ Las funciones de base radial de las unidades ocultas no almacenan medias y desviaciones típicas, sino en general, centroides y radios. Las unidades ocultas calculan una función Gaussiana de las entradas  $x_1, \dots, x_n$  que constituyen el vector de entrada  $x$
- ❑ Se aprenden los pesos  $w_i$ , los centroides  $\mu_i$  o  $c_i$ , y los radios  $\sigma_i$  o  $r_i$  minimizando la función del error cuadrático (aprendizaje por gradiente descendente)

INPUTS      HIDDEN LAYER      OUTPUT



$$h_i = \exp\left[-\frac{(\mathbf{x} - \mathbf{u}_i)^T (\mathbf{x} - \mathbf{u}_i)}{2\sigma^2}\right], \quad y = \sum_i h_i w_i$$

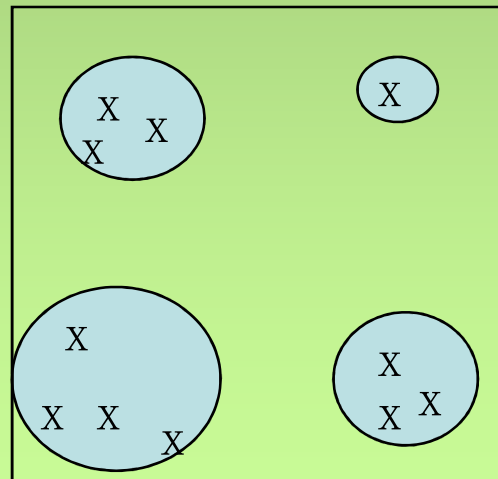
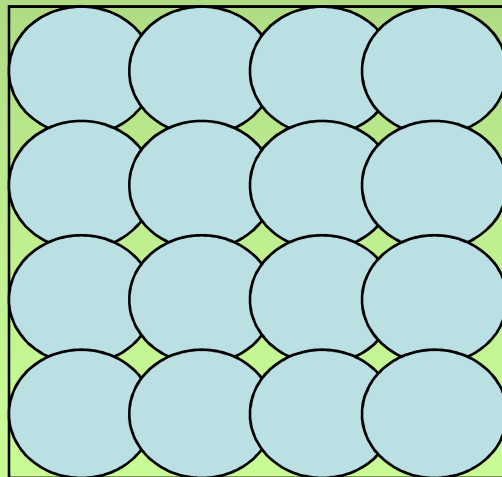


## REDES DE FUNCIONES DE BASE RADIAL (RBF)



En la aproximación que vamos a ver se supone tácitamente que se dan las funciones de transferencia de las redes RBF, esto es, se considera que tanto el centóide  $\mu$  como el radio  $r$  se consideran conocidos. ¿Pero como se eligen  $\mu$  y  $r$  ?

### a) Recubrimiento uniforme



b) Si los patrones de entrenamiento se distribuyen de forma no homogénea entonces hay que hacer en primer lugar un análisis cluster, eligiendo los centroides,  $c$ , de las funciones de base para cada cluster y utilizando el tamaño del cluster para definir  $r$





## Aprendizaje de las redes RBF mediante clustering particional



Queremos encontrar una asignación de los datos a los  $k$  clusters (cada punto a que cluster) y un conjunto de vectores que representen a cada cluster (prototipos o centroides)

Para cada  $x_j$  definimos las variables indicador  $r_{ij}$ , que valen 1 si  $x_j$  es asignado al cluster  $i$  y valen 0 en otro caso,  $i = 1, \dots, k$ ;  $j = 1, \dots, N$ , siendo  $N$  el número de patrones del conjunto de entrenamiento

**Problema del clustering:** encontrar  $\{r_{ij}\}$  y  $\{\mu_i\}$  tal que

$$\min J = \sum_{j=1}^N \sum_{i=1}^k r_{ij} \|\mathbf{x}_j - \mu_i\|^2 = \sum_{x_j \in Cl_i} \sum_{i=1}^k \|\mathbf{x}_j - \mu_i\|^2$$

es decir, minimizar la suma ponderada de cuadrados de las distancias de cada objeto (patrón)  $x_j$  a su prototipo o centroide  $\mu_i$



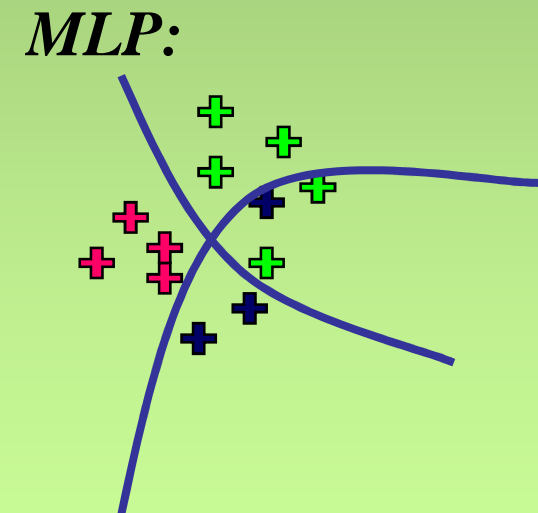
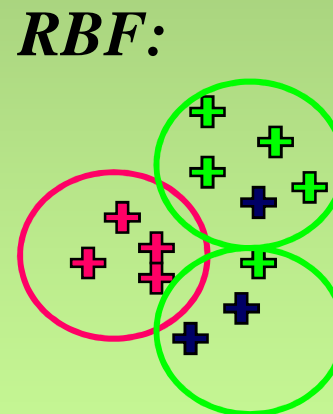
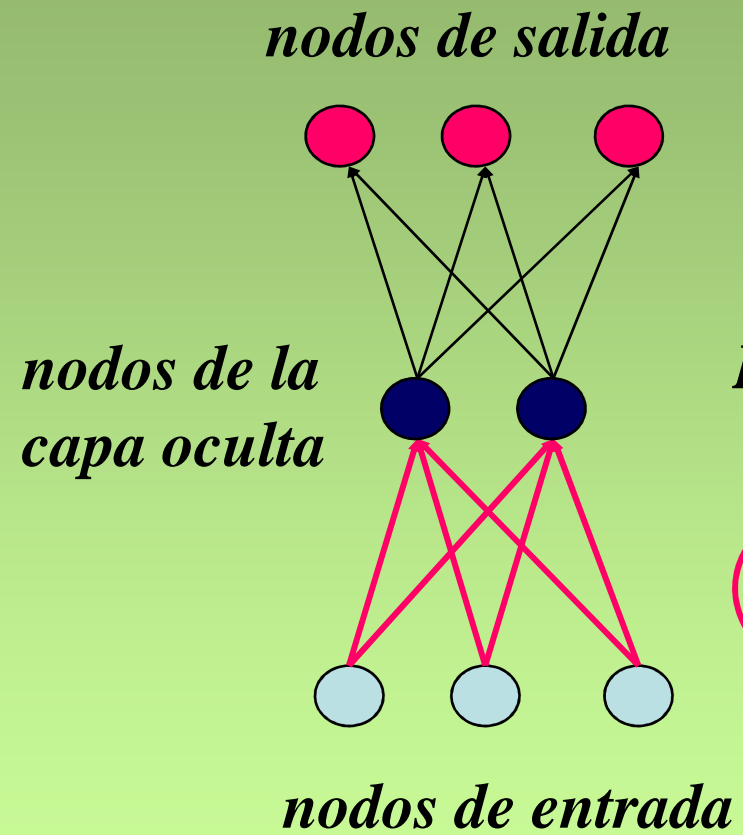
### Procedimiento general

- 1 Seleccionar k semillas iniciales (centroides de los clusters)**
- 2 Asignar cada objeto al cluster (centroide) más cercano**
- 3 Actualizar los centroides**
- 4 Repetir el proceso hasta que los centroides no cambian (convergencia)**

**Los distintos métodos difieren sobre todo en los pasos 1 y 3**



# MLPs versus RBFs





## MLPs versus RBFs



### Clasificación

- MLPs separa clases mediante funciones no lineales
- RBFs separa clases via hiperesferas

### □ Aprendizaje

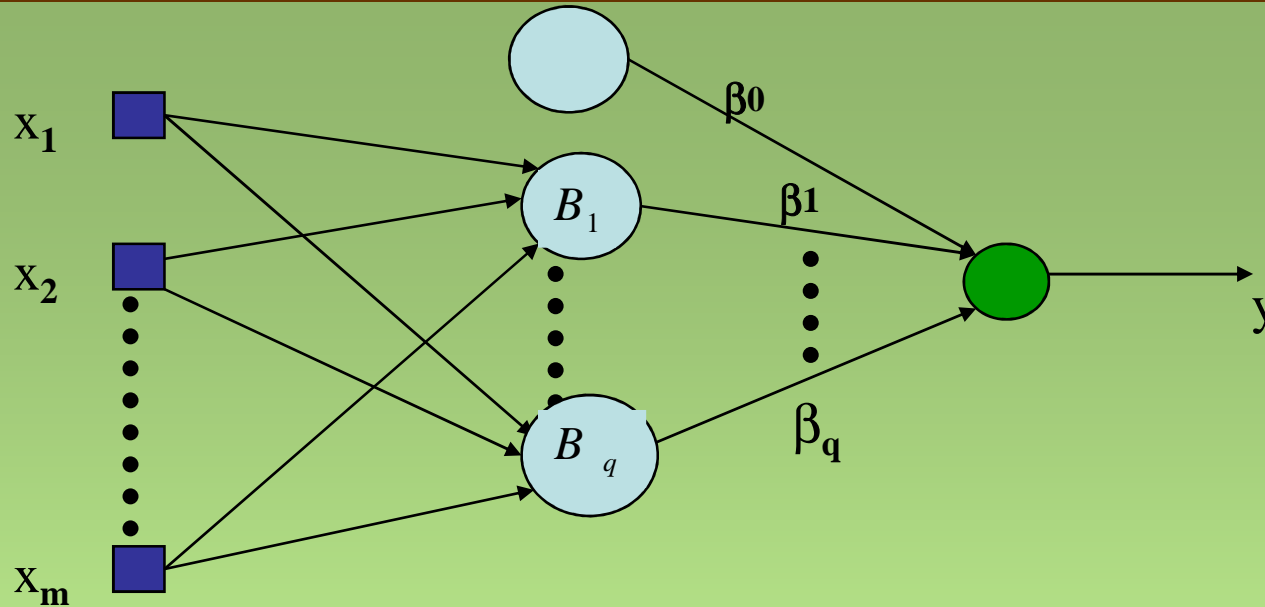
- MLPs usa aprendizaje distribuido
- RBFs usa aprendizaje localizado
- RBFs entrena más rápidamente

### □ Estructura

- MLPs tiene una o más capas ocultas
- RBFs tiene una sola capa oculta
- RBFs necesita, en general, más neuronas ocultas => problema de “curse of dimensionality” (maldición de la dimensionalidad)



## Arquitectura de una red RBF



- Una capa oculta con funciones de activación de tipo RBF
- Una capa de salida con función de activación lineal.

$$y = \beta_0 + \beta_1 B_1(\|\mathbf{x} - \mathbf{c}_1\|) + \dots + \beta_q B_q(\|\mathbf{x} - \mathbf{c}_q\|)$$

$\|\mathbf{x} - \mathbf{c}\|$  distancia de  $\mathbf{x} = (x_1, \dots, x_m)$  al centroide  $\mathbf{c}$  de la función



## REDES DE FUNCIONES DE BASE RADIAL (RBF)



**Definición:** Una función  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  se dice que es una red de funciones de base radial (RBF) sii

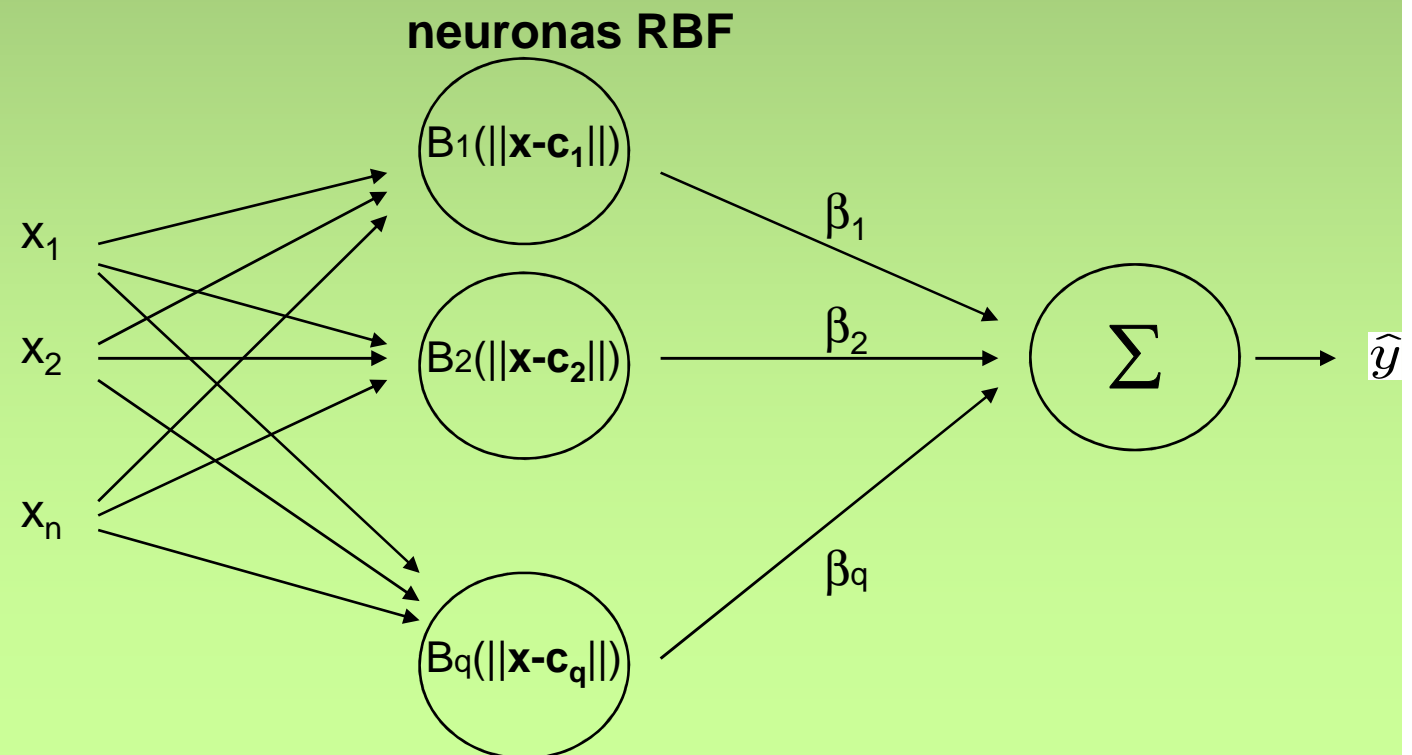
$$f(\mathbf{x}) = \beta_1 \phi(\|\mathbf{x} - \mathbf{c}_1\|) + \beta_2 \phi(\|\mathbf{x} - \mathbf{c}_2\|) + \dots + \beta_q \phi(\|\mathbf{x} - \mathbf{c}_q\|).$$

Donde  $\|\mathbf{x}\|$  denota la norma Euclidea del vector  $\mathbf{x}$ .

**Ejemplo función Gaussiana:**

$$B(\mathbf{x}; \mathbf{c}, r) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}\|^2}{2r^2}\right),$$

$\mathbf{c}$  es el centroide y  $r$  es el radio





## Clasificador binario con una red de unidades RBF



$$f(\mathbf{x}, \boldsymbol{\theta}) = \beta_0 + \beta_1 B(\mathbf{x}; (\mathbf{c} | r))$$

para  $\mathbf{x}=(x_1, x_2, x_3)^T$  y  $\mathbf{c}=(c_1, c_2, c_3)^T$ , siendo

$$B(\mathbf{x}; \mathbf{c}, r) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}\|^2}{2r^2}\right),$$

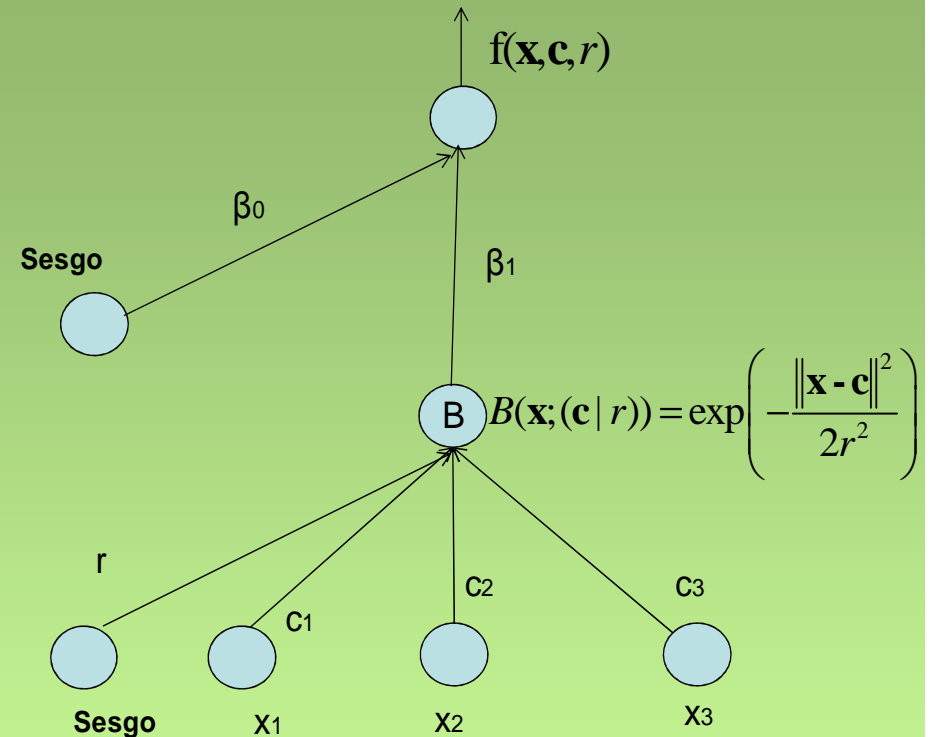
donde  $\mathbf{c}$  es el centroide y  $r$  es el radio del núcleo

Definimos la función de clasificación en la forma

$$p(\mathbf{x}, \boldsymbol{\theta}) = \frac{\exp(f(\mathbf{x}, \boldsymbol{\theta}))}{1 + \exp(f(\mathbf{x}, \boldsymbol{\theta}))}$$

La función de decisión es de la forma

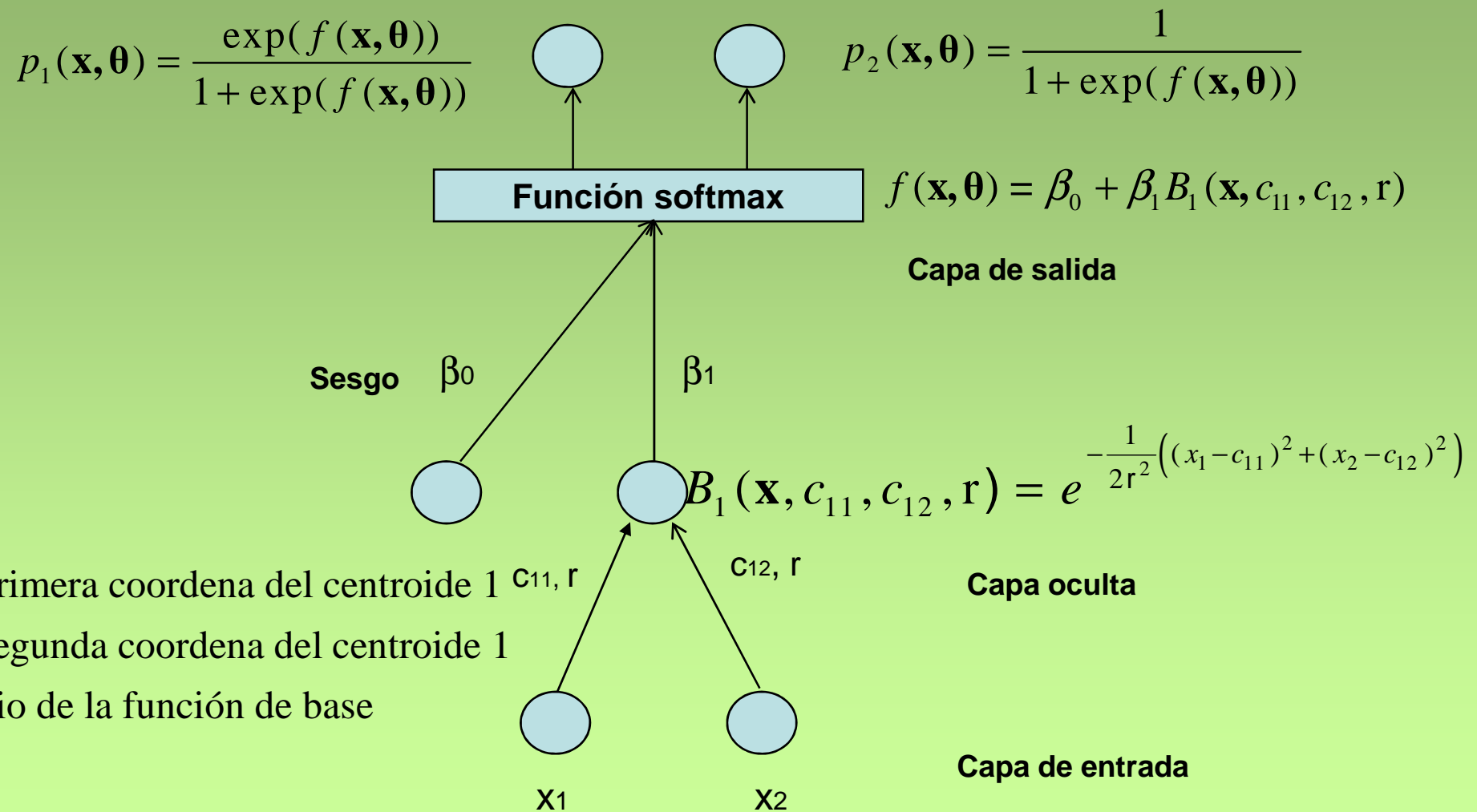
Si  $p(\mathbf{x}, \boldsymbol{\theta}) > 0,5$  entonces  $\mathbf{x} \in C_1$  (clase positiva)





# Ejemplo de red neuronal de funciones de base radial: para clasificación binaria

## Representación de un modelo







## Arquitectura de una red RBF



Aquí necesitamos solamente aprender los pesos,  $\beta_i$ , de la capa oculta a la capa de salida.

Los pesos  $\beta_i$  se pueden determinar con la ayuda de cualquiera de los métodos iterativos estándar **descritos anteriormente** para las redes neuronales.

Sin embargo, ya que la función de aproximación dada abajo es lineal con respecto a  $\beta_i$ , se puede calcular directamente utilizando matrices asociadas a métodos lineales de mínimos cuadrados **sin** tener que determinar explícitamente  $\beta_i$  de forma iterativa.

Cabe señalar que la función  $f(x)$  es diferenciable con respecto a  $\beta_i$



# REDES DE FUNCIONES DE BASE RADIAL



Dados: N patrones de entrenamiento  $(x_i, y_i)$  y q neuronas RBF

Encontrar: pesos  $\beta_1, \dots, \beta_q$  con minimo error

Solución:

Sabemos que  $f(x_i) = y_i$  para  $i = 1, \dots, N$  o de forma equivalente

$$\sum_{k=1}^q \underbrace{\beta_k \cdot B_k(\|x_i - c_k\|)}_{\text{Valor conocido}} = \underbrace{y_i}_{\text{valor conocido}}$$

Diagram illustrating the equation  $\sum_{k=1}^q \beta_k \cdot B_k(\|x_i - c_k\|) = y_i$  for  $i = 1, \dots, N$ . The term  $\beta_k$  is labeled "desconocido" (unknown). The term  $B_k(\|x_i - c_k\|)$  is labeled "Valor conocido" (known value) with the subscript  $p_{ik}$  below it. The term  $y_i$  is labeled "valor conocido" (known value).

Tenemos, por tanto, N ecuaciones lineales con q incognitas



## REDES DE FUNCIONES DE BASE RADIAL (RBF)



Tenemos  $N$  patrones y  $q$  funciones de base radial

En forma matricial :  $\mathbf{P}_{N \times q} \boldsymbol{\beta}_{q \times 1} = \mathbf{y}_{N \times 1}$ , donde  $\mathbf{P} \equiv (p_{ik})$

Caso  $N=q$ ; entonces  $\boldsymbol{\beta} = \mathbf{P}^{-1} \mathbf{y}$ , si  $\mathbf{P}$  es de rango completo

Caso  $N < q$ ; muchas soluciones, pero entonces no tiene ninguna relevancia práctica y los pesos habría que optimizarlos mediante una heurística

Caso  $N > q$ ; entonces  $\boldsymbol{\beta} = \mathbf{P}^+ \mathbf{y}$ ,

donde  $\mathbf{P}^+$  es la matriz pseudo inversa de Moore Penrose

En la ecuación  $\mathbf{P}\boldsymbol{\beta} = \mathbf{y}$ , si multiplicamos por  $\mathbf{P}^T$  por la izquierda tenemos

$\mathbf{P}^T \mathbf{P} \boldsymbol{\beta} = \mathbf{P}^T \mathbf{y}$ ; y si ahora multiplicamos por  $(\mathbf{P}^T \mathbf{P})^{-1}$  también por la izquierda

tenemos  $(\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{P} \boldsymbol{\beta} = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{y}$ , donde  $(\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{P} = \mathbf{I}$ , y

$(\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T$  es la matriz de Moore Penrose,  $\mathbf{P}^+$



## REDES DE FUNCIONES DE BASE RADIAL (RBF Nets)



**Complejidad (simple)**

$$\beta = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{y} = \mathbf{P}^+ \mathbf{y}$$

Para el producto matricial  $\mathbf{P}^T \mathbf{P}$ : la complejidad es  $N^2 q$ ; para la inversión  $q^3$  ;  
para el producto de matriz por vector  $\mathbf{P}^T \mathbf{y}$ :  $qN$   
y para la multiplicación  $q^2$ ; por tanto la complejidad es  $O(N^2 q)$

**Observación:** Si  $N$ , el número de patrones de entrenamiento, es grande entonces el calculo de  $(\mathbf{P}^T \mathbf{P})^{-1}$  será probablemente inexacto. Si la solución es analítica, entonces utilizaremos el algoritmo de gradiente descendente a partir de esta solución. Pero para ello se necesitan funciones de base diferenciables.



## REDES DE FUNCIONES DE BASE RADIAL (RBF)



### Ventajas:

Patrones de entrenamiento adicionales implican sólo un ajuste local de los pesos.

Pesos óptimos obtenidos en tiempo polinómico

Las regiones no reconocidas por la red RBF pueden ser identificadas por salidas iguales a cero.

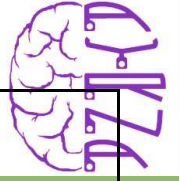
### Inconvenientes:

El número de neuronas aumenta exponencialmente con la dimensión del espacio de entrada.

No puede extrapolar (ya que no hay centros fuera del entorno de los patrones de entrenamiento y las redes RBFs son locales).



## Comparación de redes RBF y MLP



### RBF

### MLP

Redes con capas no lineales con activación hacia adelante

Redes con capas no lineales con activación hacia adelante

La capa oculta de la red RBF es no-lineal, la capa de salida es lineal.

Las capas oculta y de salida de una red MLP son por lo general no-lineales, a no ser que utilicemos la función softmax.

Una sola capa oculta.

Puede tener más de una capa oculta.

El modelo de neurona de la capa oculta es diferente del modelo de los nodos de la capa de salida.

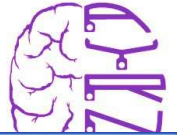
Las neuronas ocultas y de salida comparten, por lo general, el mismo modelo de neurona.

La función de activación de cada neurona de la capa oculta en una RBF calcula la distancia euclídea entre el vector de entrada y el centro de esa unidad.

La función de activación de cada neurona de la capa oculta en una MLP calcula el producto interno de vector de entrada y del vector de pesos sinápticos de esa neurona.



# Ejemplo Indias Pima



\*ejemploADLPimadef.sav [Conjunto\_de\_datos1] - Editor de datos SPSS

Archivo Edición Ver Datos Transformar Analizar Gráficos Utilidades Ventana ?

1 : PRE\_1 0.821993929850541 Visible: 12 de 12 va

	conjunto	nembara	cgluc osa	pdiastol	gdobpi el	ninsulin	imasacor	einfluge	edad	clase	PRE_1	PGR_1
1	entrenamiento	.235294	.855	.59016	.00	.000000	.6498	.171221	.083333	diabetes	.82199	diabetes
2	entrenamiento	.117647	.525	.65574	.45	.225768	.5022	.270282	.133333	diabetes	.19128	nod diabetes
3	entrenamiento	.058824	.530	.62295	.00	.000000	.5589	.050811	.083333	nod diabetes	.16202	nod diabetes
4	entrenamiento	.294118	.495	.60656	.27	.000000	.4322	.053373	.183333	nod diabetes	.12729	nod diabetes
5	entrenamiento	.235294	.455	.57377	.32	.104019	.4933	.157131	.016667	nod diabetes	.13510	nod diabetes
6	entrenamiento	.294118	.580	.60656	.29	.000000	.4814	.248506	.233333	diabetes	.35858	nod diabetes
7	entrenamiento	.588235	.540	.54098	.00	.000000	.4829	.082835	.350000	diabetes	.44250	nod diabetes
8	entrenamiento	.117647	.440	.60656	.19	.062648	.4322	.064475	.016667	nod diabetes	.05313	nod diabetes
9	entrenamiento	.235294	.600	.55738	.00	.000000	.4411	.269428	.216667	nod diabetes	.32378	nod diabetes
10	entrenamiento	.176471	.530	.59016	.00	.000000	.3845	.055081	.100000	nod diabetes	.08386	nod diabetes
11	entrenamiento	.117647	.410	.42623	.22	.135934	.4247	.692143	.066667	nod diabetes	.24225	nod diabetes
12	entrenamiento	.352941	.825	.55738	.26	.198582	.5007	.236123	.466667	nod diabetes	.80126	diabetes
13	entrenamiento	.235294	.625	.57377	.18	.144208	.4307	.455167	.400000	diabetes	.49341	nod diabetes
14	entrenamiento	.294118	.620	.60656	.00	.000000	.5067	.060632	.283333	diabetes	.35791	nod diabetes
15	entrenamiento	.235294	.755	.73771	.38	.000000	.4426	.092229	.250000	nod diabetes	.35208	nod diabetes
16	entrenamiento	.117647	.415	.53279	.28	.078014	.5484	.235269	.050000	nod diabetes	.16005	nod diabetes
17	entrenamiento	.000000	.655	.00000	.00	.000000	.6438	.081981	.083333	diabetes	.70419	diabetes
18	entrenamiento	.176471	.540	.50820	.24	.000000	.3875	.061913	.066667	nod diabetes	.10256	nod diabetes
19	entrenamiento	.000000	.900	.63934	.63	.016549	.8852	1.000000	.066667	diabetes	.99062	diabetes
20	entrenamiento	.117647	.460	.62295	.20	.000000	.3607	.691716	.116667	nod diabetes	.17368	nod diabetes

Vista de datos / Vista de variables /

Área de recuento de casos SPSS El procesador está preparado



# EJEMPLO DE REDES RBF EN NNEP:INDIAS PIMA



Number of Executions

Algorithm

Type of Algorithm

Type of Problem

Specie

Type of Specie

Evaluator

Error Function

☒ Normalize Data

Evaluation Method

Train Data File

Test Data File

Mutators

Structural Mutator

Parametric Mutator

Listeners

Statistical Listener

☒ Statistical Reports Number of Decimals

Report Directory Name  Extended Report Frequency

Chart Listener

☒ Chart Reports Options: ☒ Best Fitness ☒ CCR/MSE ☒ Means Fitness

## Algorithm Properties Configuration

Population Size

Max. Number of Generations

Max. Gen. Without Improving Mean

Max. Gen. Without Improving Best

Creation Ratio

Percentage Second Mutator (%)

Significative Fitness Difference

Seed for Random Generators





# EJEMPLO DE REDES RBF EN NNEP:INDIAS PIMA



Number of Executions

Algorithm

Type of Algorithm

Type of Problem

Specie

Type of Specie

Evaluator

Error Function

☒ Normalize Data

Evaluation Method

Train Data File

Test Data File

Mutators

Structural Mutator

Parametric Mutator

Listeners

Statistical Listener

☒ Statistical Reports Number of Decimals

Report Directory Name  Extended Report Frequency

Chart Listener

☒ Chart Reports Options: ☒ Best Fitness ☒ CCR/MSE ☒ Means Fitness

Pure Layer Properties Configuration

Weight Range

Interval Limits

Left Limit  Right Limit

Input Layer

Number of Input

Hidden Layer

Type of Layer

Initiator  ☒ Biased

Minimum Number of Neuro...

Initial Number of Neurons

Max. Number of Neurons

Output Layer

Type of Layer

Initiator  ☒ Biased

Number of Outputs



# EJEMPLO DE REDES RBF EN NNEP:INDIAS PIMA



Number of Executions

Algorithm

Type of Algorithm

Type of Problem

Specie

Type of Specie

Evaluator

Error Function

☒ Normalize Data

Evaluation Method

Train Data File

Test Data File

Mutators

Structural Mutator

Parametric Mutator

Listeners

Statistical Listener

☒ Statistical Reports Number of Decimals

Report Directory Name  Extended Report Frequency

Chart Listener

☒ Chart Reports Options: ☒ Best Fitness ☒ CCR/MSE ☒ Means Fitness

Normalize Data Properties Configuration

Input Data Interval

Interval Limits

Left Limit  Right Limit

Output Data Interval

Interval Limits

Left Limit  Right Limit

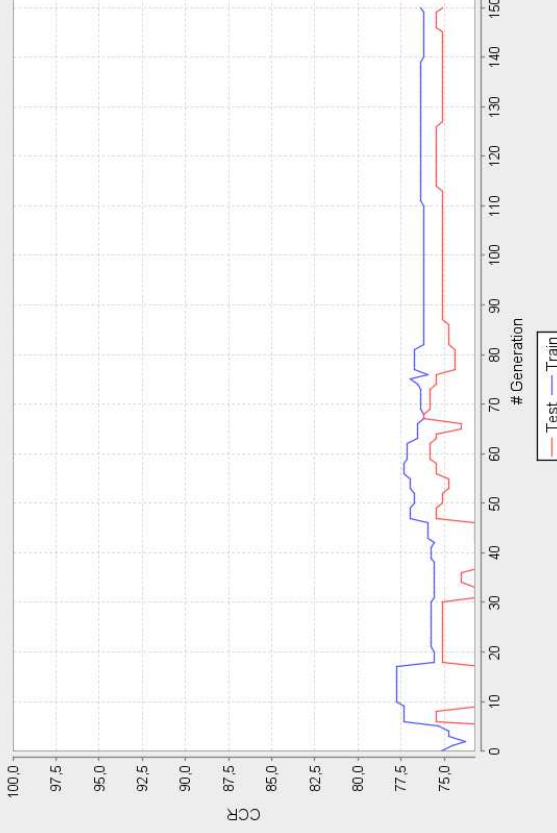
Iteration	Generation	CABestEP	NoflinksBestEP	Iteration	Generation	CABestEP	NoflinksBestEP
1	100	76,1523046	5	1	100	75,0929368	5
2	100	77,7555110	6	2	100	75,4646840	6
3	100	78,9579158	8	3	100	76,5799257	8
4	100	79,3587174	8	4	100	76,5799257	8
5	100	79,1583166	8	5	100	77,3234201	8
	Mean	78,2765531	7,0000000		Mean	76,2081784	7,0000000
	StdDev	1,3413387	1,1142136		StdDev	0,9105910	1,4142136

Executions

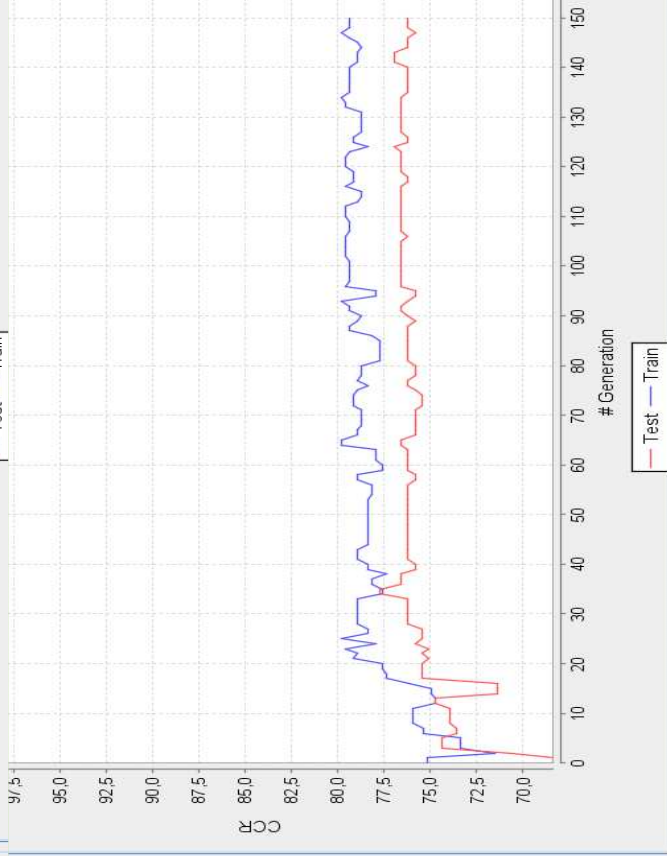
- Execution #0
- Execution #1
- Execution #2
- Execution #3
- Execution #4

Fitness CCR Mean Fitness

Classification



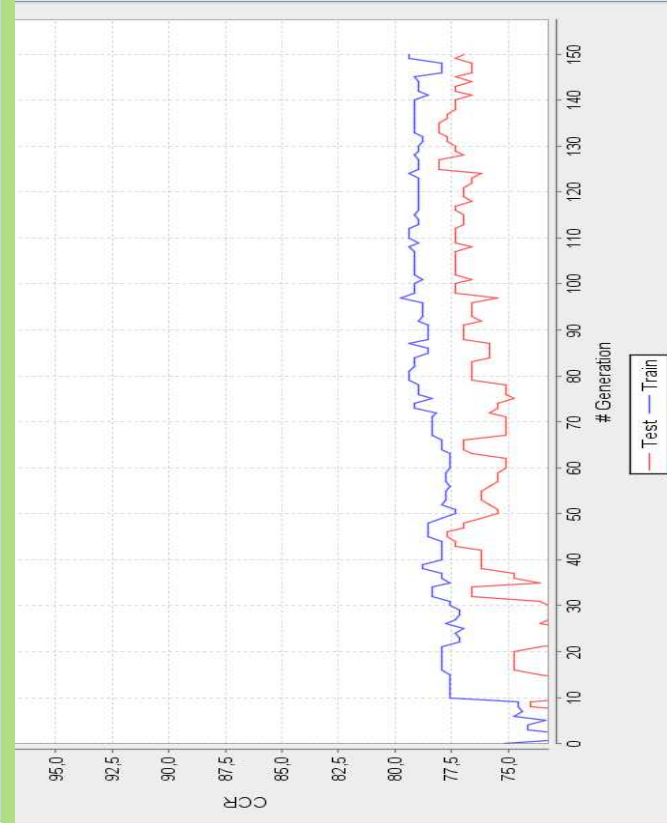
- Execution #2
- Execution #3
- Execution #4



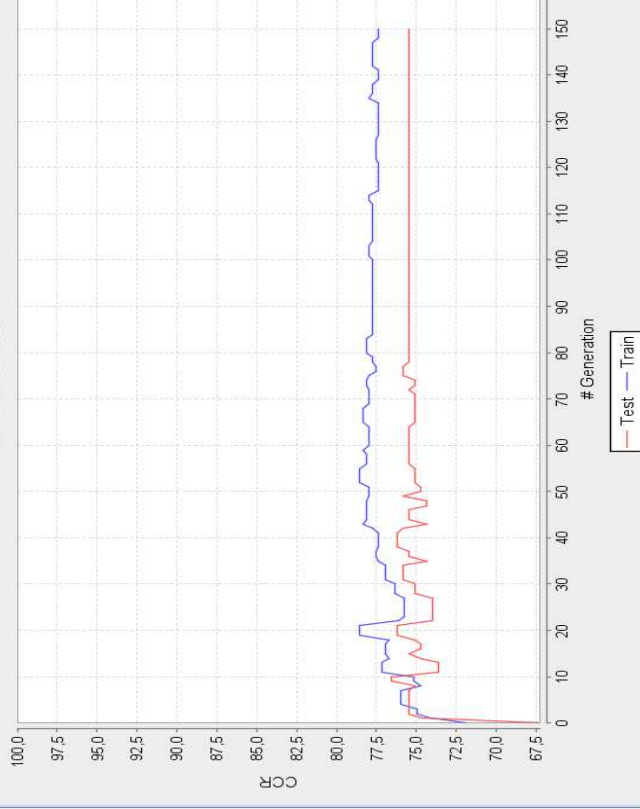
- Execution #0
- Execution #1
- Execution #2
- Execution #3
- Execution #4

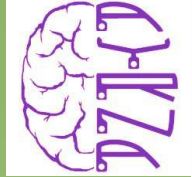


- Execution #3
- Execution #4



Classification





**MODELOS COMPUTACIONALES: CUARTO CURSO  
DEL GRADO  
DE ING. INFORMÁTICA EN COMPUTACION**

# **REDES DE HOPFIELD**

**César Hervás-Martínez  
Grupo de Investigación AYRNA**

**Departamento de Informática y Análisis  
Numérico  
Universidad de Córdoba  
Campus de Rabanales. Edificio Einstein.  
Email: [chervas@uco.es](mailto:chervas@uco.es)**

**2020-2021**



## Redes de Hopfield



**La importancia de las redes de Hopfield en la práctica es limitada debido a las restricciones teóricas de la estructura, pero, en algunos casos, se pueden formar modelos interesantes.**

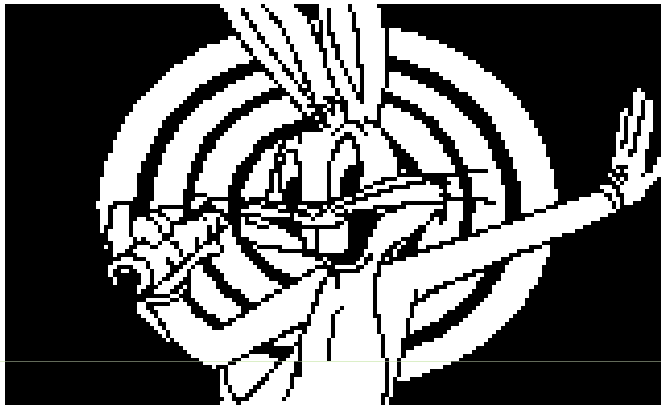


## Redes de Hopfield



Por lo general los modelos de Redes de Hopfield, trabajan en tareas de lógica binaria: Por ejemplo, reconstrucción de patrones y reglas de asociación

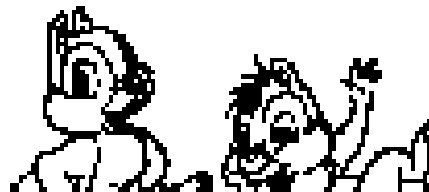
Original



Degraded



Reconstruction



La red de Hopfield reconstruye imágenes degradadas por ruido (ver imagen superior) o por señales parciales (ver imagen inferior)



# Red de Hopfield



Una red de Hopfield es un modelo de memoria asociativa.

Se basa en el aprendizaje hebbiano pero usa neuronas binarias.

Proporciona un modelo formal que puede analizarse para determinar la capacidad de almacenamiento de la red.

Está inspirado en su formulación por modelos de mecánica estadística (modelo de Ising) para materiales magnéticos.

Proporciona un camino para generalizar modelos de redes deterministas para el caso estocástico.





# Red de Hopfield



- Una red de Hopfield es un tipo de red monocapa recurrente cuyos valores de salida son realimentados, de nuevo, a la entrada de una manera no dirigida.
- Esta asociada a resolver problemas de memoria asociativa.
- Se trata de un aprendizaje no supervisado donde se agrupan los patrones por similaridad.
- Se compone de un conjunto de  $N$  neuronas conectadas con pesos que son simétricos y ninguna unidad está conectada a sí misma.
- No hay neuronas especiales de entrada y de salida.
- La activación de una neurona es un valor binario decidido por el signo de la suma ponderada de las conexiones de entrada a la misma.



# Red de Hopfield



- Un **valor umbral** para cada neurona determina si está es, o no, una neurona activa.
- Una neurona activa es una que activa todas las neuronas que están conectadas a ella con un peso positivo.
- La entrada se aplica simultáneamente a todas las neuronas, la cual produce una salida en cada una.
- Este proceso continúa hasta que se alcanza un estado estable.



## Red de Hopfield



El problema de la **memoria asociativa** se resume de la siguiente manera:

Almacena un conjunto de patrones de tal forma que cuando se le presenta un nuevo patrón, la red responde produciendo uno de los patrones almacenados que más se parezca a el.

Tanto los vectores, de los patrones almacenados, como los patrones de prueba, se **codifican mediante -1 (estado no activo) o 1 (estado activo)** en cada una de sus componentes.

Una **memoria asociativa** puede pensarse como un conjunto de **atractores**, cada uno con su propio nicho o cuenca de atracción.



## Redes de Hopfield



Al igual que todos los computadores, un cerebro es un sistema dinámico que lleva a cabo sus cálculos por el cambio del “estado” de sus neuronas a lo largo del tiempo.

En los modelos simples de la dinámica de los circuitos neuronales se describe que tienen propiedades dinámicas colectivas.

Estas pueden ser explotadas en el reconocimiento de patrones sensoriales.

La utilización de estas propiedades colectivas de procesamiento de la información es eficaz, en tanto en cuanto, aprovechen las propiedades espontáneas de las células nerviosas y de los circuitos para producir un cálculo robusto.



## Algoritmo de activacion

Repetir



**Elija cualquier unidad al azar.** La unidad elegida puede estar activa o inactiva.

- Para la unidad elegida, calcular la suma de los pesos de las conexiones sólo a los vecinos activos, si los hay.
- Si la suma es  $> 0$  (**se supone que el umbral es 0**), entonces la unidad elegida se convierte en activa, de lo contrario, se vuelve inactiva.
- Si la unidad elegida no tiene vecinos activos entonces se ignora, y el estado sigue siendo igual.
- **Hasta que la red alcanza un estado estable, esto es, un mínimo de la función de energía.**



# Redes de Hopfield



## La búsqueda de J. Hopfield

**Mientras que el cerebro es totalmente diferente a los ordenadores actuales, gran parte de lo que hace puede ser descrito como computación.**

**Los seres humanos mediante la memoria asociativa, la lógica y la deducción, reconocen un olor o una posición de ajedrez, analizan el mundo a partir de objetos, y generan secuencias apropiadas de mandatos musculares del aparato locomotor que son todos descritos como cálculo.**

**La investigación de Hopfield se centra en el entendimiento de cómo los circuitos neuronales del cerebro producen cálculos potentes y complejos.**



# Redes de Hopfield

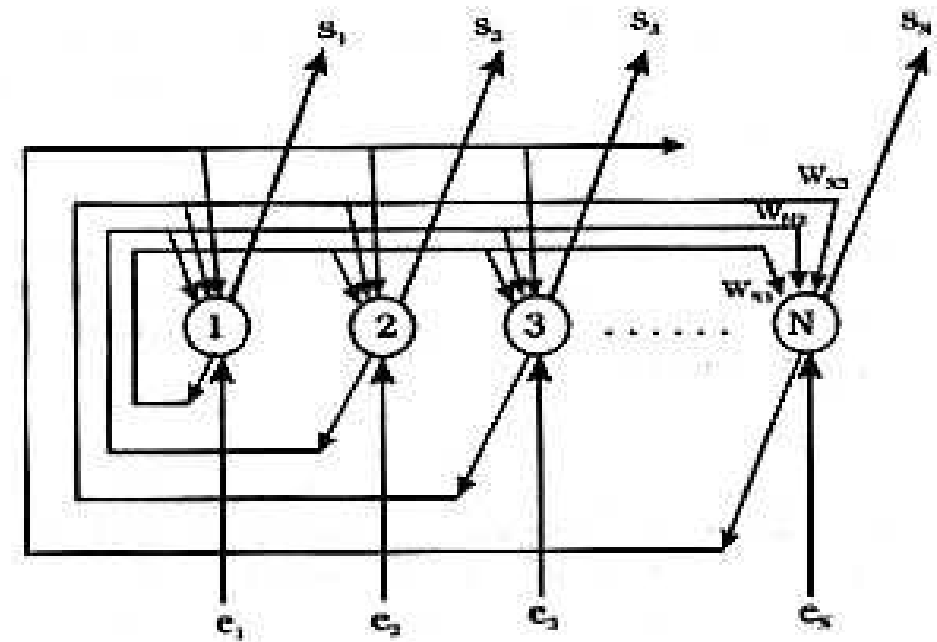
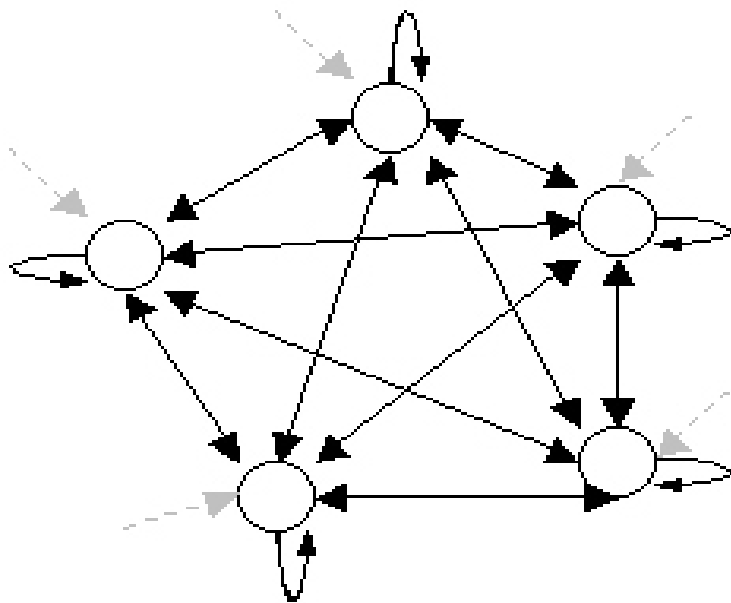


Figura 8: Arquitectura de una Red de Hopfield

*Red Hopfield*



## Redes de Hopfield



### Calculo del potencial de acción

**Para gran parte de la neurobiología, la información está representada por el paradigma de las “*tasas de disparo*”, es decir, se representa la información por la tasa de generación de picos de potenciales de acción, donde el momento exacto de estos picos es poco importante.**





## Redes de Hopfield



### Calculo del potencial de acción

**Dado que los potenciales de acción duran sólo alrededor de una milésima de segundo, el uso del instante del potencial de acción aparece como un medio poderoso de la computación neuronal.**

**Hay casos, como por ejemplo en la determinación auditiva fonética binaria de la ubicación de una fuente de sonido, donde la información se codifica en el momento de los potenciales de acción.**



# Redes de Hopfield



Actúan como las memorias “autoasociativas” para almacenar patrones

- Son neuronas de tipo McCulloch-Pitts con salidas -1 o 1, y umbral  $\Theta$ , en este caso 0

Red completamente conectada

- Todas las neuronas están conectadas entre si

- Pesos simétricos ( $w_{ij} = w_{ji}$ ) y  $w_{ii} = 0$

- Cambio asíncrono de las salidas

- Sea  $s_i$  el estado de la unidad  $i$

- En cada etapa de tiempo se elige una unidad al azar

- Definimos

$$s_i = 1 \quad \text{si} \quad \sum_j w_{ij} s_j \geq \Theta, \text{ para todo nodo } j \text{ activo conectado al nodo } i$$

$$s_i = -1 \quad \text{en otro caso}$$

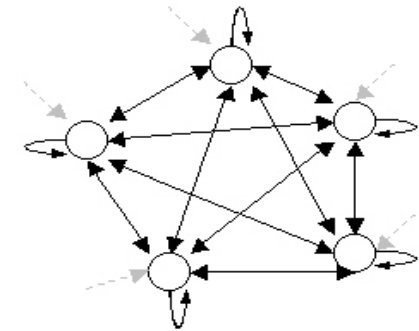


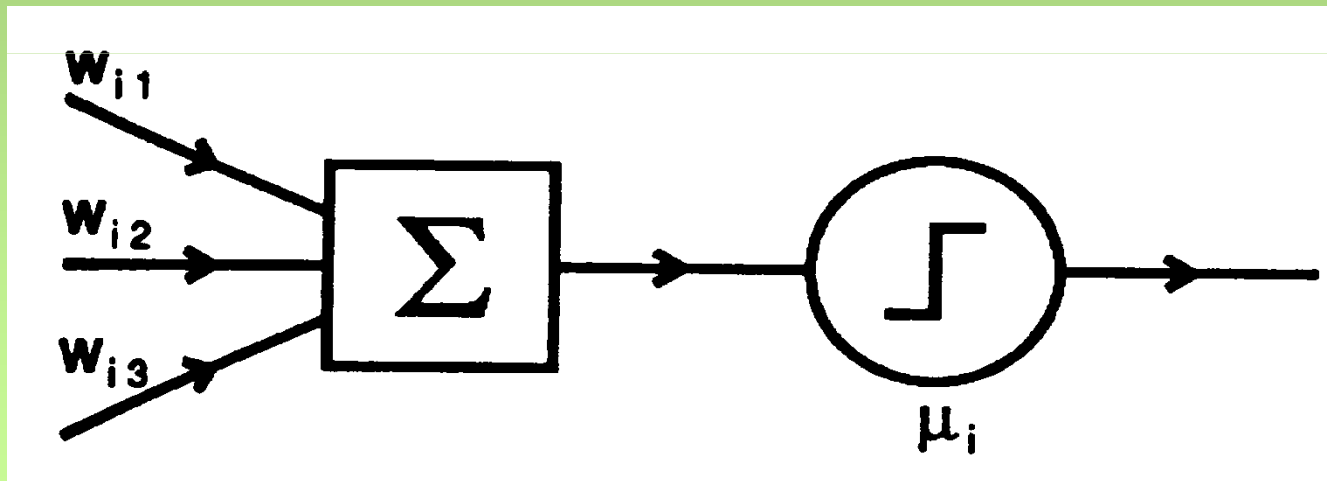
Figura 8: Arquitectura de una Red de Hopfield



## Neurona de McCulloch–Pitts (1943)



- Atributos de la neurona genérica
  - m entradas binarias y una salida (0 o 1)
  - Pesos sinápticos  $w_{ij}$
  - Umbral  $\mu_i$





## Redes de Hopfield



□ Hopfield demostró que la actualización asíncrona en redes simétricas minimiza una función de "energía" y conduce a un estado final estable, un mínimo de la función de energía, para un estado inicial dado.

□ Se define una función de energía a minimizar (análoga a la función de error del gradiente descendente)

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j + \sum_i s_i \Theta_i$$



## Redes de Hopfield: Función de energía



$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j + \sum_i s_i \Theta_i$$

$$\text{Si } \Theta_i = 0$$

$$E(t) = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j$$

$$E(t) = -\frac{1}{2} \sum_i s_i \sum_j w_{ij} s_j = -\frac{1}{2} (\mathbf{s}(\mathbf{W}\mathbf{s}))$$



## Redes de Hopfield



□ Si una unidad al azar ha sido actualizada, entonces la función de energía  $E$  siempre decrece!

$$E(t) = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j + \sum_i s_i \Theta_i$$

- Si  $s_i$  es inicialmente  $-1$  (inactiva) y  $\sum_j w_{ij} s_j > \Theta_i$ , esto es  $-\sum_j w_{ij} s_j < -\Theta_i$  entonces  $s_i$  se convierte en  $+1$  (activa)
- Al pasar a ser  $s_i=1$ , hay un cambio en el valor inicial de  $E$ , en la forma, dado que  $w_{ij} = w_{ji}$

$$E(t+1) = -\frac{1}{2} 2 \sum_j w_{ij} s_j + \Theta_i, \text{ dado que } s_i = 1$$

$$E(t) = \frac{1}{2} 2 \sum_j w_{ij} s_j - \Theta_i, \text{ dado que } s_i = -1$$

$$\begin{aligned} E(t+1) - E(t) &= -\frac{1}{2} 2 \sum_j (w_{ij} s_j - w_{ji} (-s_j)) + \Theta_i - (-\Theta_i) = \\ &= -\frac{1}{2} 2 \sum_j (w_{ij} s_j + w_{ji} s_j) + 2\Theta_i = 2(-\sum_j w_{ij} s_j + \Theta_i) < 0 \quad !! \end{aligned}$$



## Redes de Hopfield



□ Si una unidad al azar ha sido actualizada, entonces la función de energía  $E$  siempre decrece!

$$E(t) = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j + \sum_i s_i \Theta_i$$

- Si  $s_i$  es inicialmente +1 y  $\sum_j w_{ij} s_j < \Theta_i$  entonces  $s_i$  se convierte en -1

- Al pasar a ser  $s_i = -1$ , hay una disminución en  $E$

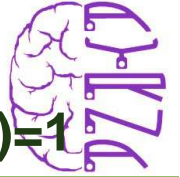
$$E(t+1) = -\frac{1}{2} 2 \sum_j w_{ij} s_j - \Theta_i, \text{ dado que } s_i = -1$$

$$E(t) = -\frac{1}{2} 2 \sum_j w_{ij} s_j + \Theta_i, \text{ dado que } s_i = 1$$

$$E(t+1) - E(t) = \frac{1}{2} 2 \sum_j (w_{ij} s_j + w_{ji} s_j) - 2\Theta_i = 2(\sum_j w_{ij} s_j - \Theta_i) < 0 \quad !!$$



## Redes de Hopfield: Ejemplo



Supongamos que tenemos sólo dos unidades y que  $s_1(t)=-1$  y  $s_1(t+1)=1$

$$E(t) = -\frac{1}{2}w_{12}s_1s_2 - \frac{1}{2}w_{21}s_2s_1 + s_1\Theta_1 + s_2\Theta_2; \text{ para } s_1=-1, \text{ tenemos}$$

$$E(t) = +\frac{1}{2}w_{12}s_2 + \frac{1}{2}w_{21}s_2 - \Theta_1 + s_2\Theta_2 = w_{21}s_2 - \Theta_1 + s_2\Theta_2$$

- Si  $s_1$  es inicialmente  $-1$  (inactiva) y  $w_{12}s_2 > \Theta_1$  entonces  $s_1$  se convierte en  $+1$  (activa)
  - Al pasar a ser  $s_1(t+1)=1$ , hay un cambio en el valor inicial de  $E$ , en la forma, dado que  $w_{ij} = w_{ji}$

$$\begin{aligned} \text{para } s_1(t+1)=1, E(t+1) &= -\frac{1}{2}w_{12}s_2 - \frac{1}{2}w_{21}s_2 + \Theta_1 + s_2\Theta_2 = \\ &= -w_{21}s_2 + \Theta_1 + s_2\Theta_2 \end{aligned}$$

$$E(t+1) - E(t) = -w_{21}s_2 - w_{21}s_2 + 2\Theta_1 = 2(-w_{21}s_2 + \Theta_1) < 0 \quad !!$$





## Redes de Hopfield: Ejemplo



□ Supongamos ahora que tenemos sólo dos unidades y que  $s_1(t)=1$  y  $s_1(t+1)=-1$

Si  $s_1$  es inicialmente 1 (activa) y  $w_{12}s_2 < \Theta_1$  entonces  $s_1$  se convierte en -1 (inactiva), en general

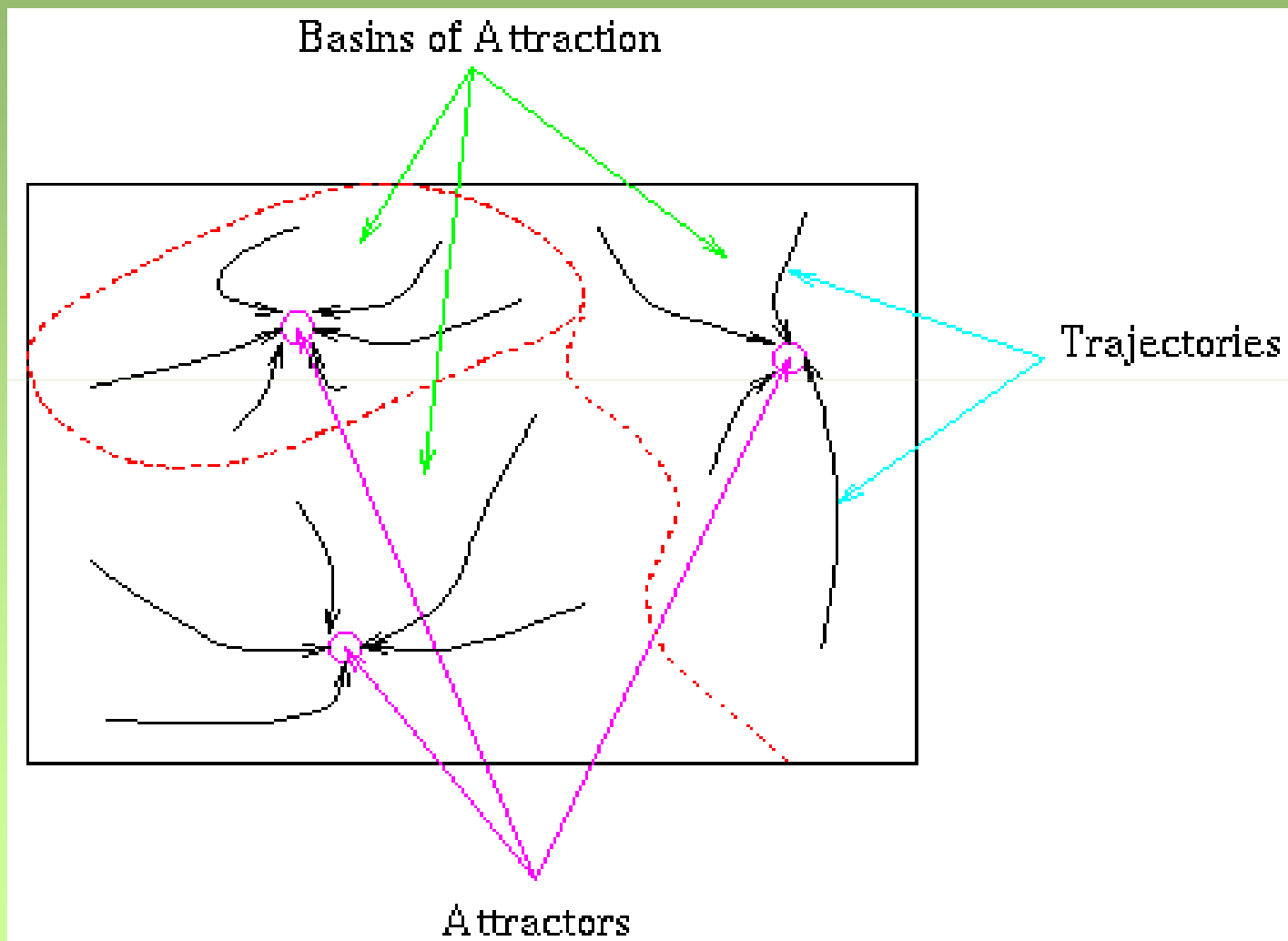
$$E(t+1) - E(t) = \frac{1}{2} 2 \sum_j (w_{ij}s_j + w_{ji}s_j) - 2\Theta_i = 2(\sum_j w_{ij}s_j - \Theta_i) < 0 \quad !!$$

Y en particular

$$E(t+1) - E(t) = w_{21}s_2 + w_{21}s_2 - 2\Theta_1 = 2(w_{21}s_2 - \Theta_1) < 0 \quad !!$$

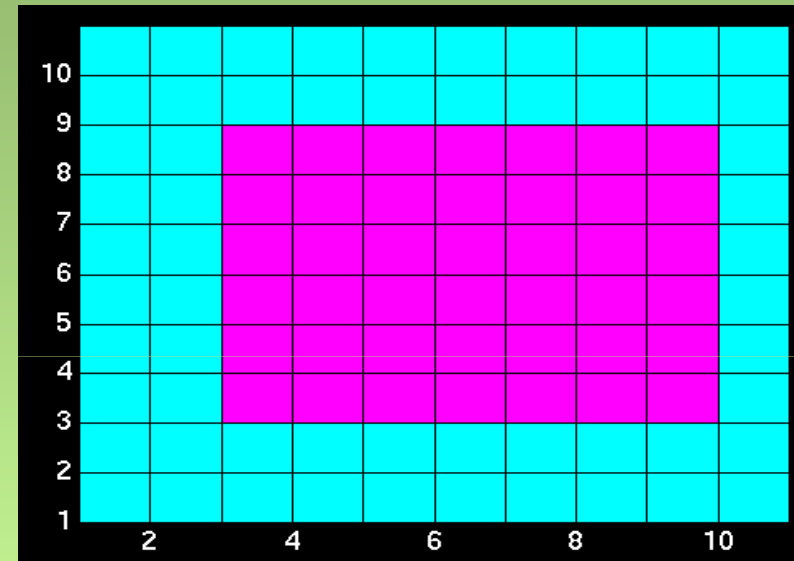
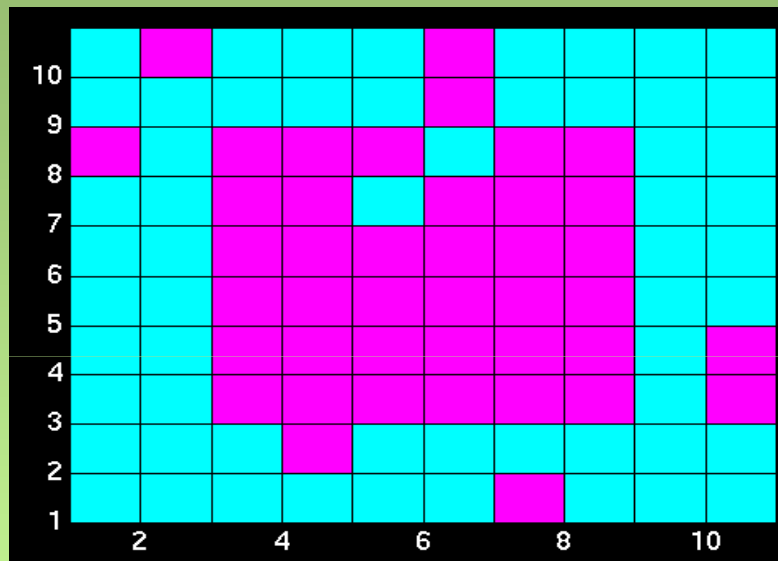


## El concepto





# Completar un patrón en una red de Hopfield



**Mínimo local (“attractor”)  
de la función de energía  
del patrón almacenado**



## Ejemplo 1



Desactivada

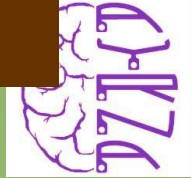


Activada

Estado actual	→ Seleccionamos una unidad a partir del estado actual	Correspondiente nuevo estado
<p>Aquí, se calcula la suma de pesos de las unidades activas de la unidad seleccionada</p>		<p><math>\text{Suma} = 3 - 2 = 1 &gt; 0;</math> activada</p>
		<p><math>\text{Suma} = -2 &lt; 0;</math> desactivada</p>



## Redes Estables: Ejemplo 2



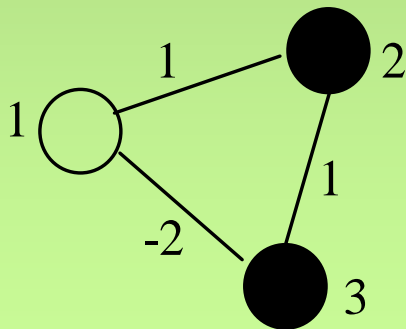
Para la unidad elegida, se calcula la suma de los pesos de las conexiones sólo a los vecinos activos, si los hay.

Si la suma es  $> 0$ , entonces la unidad elegida se convierte en activa, de lo contrario, se vuelve inactiva

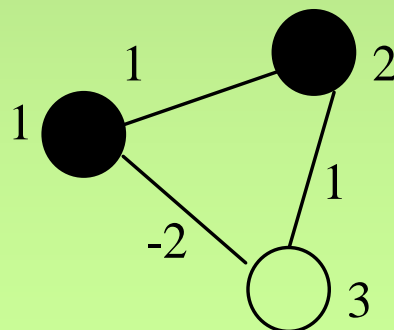
Si la unidad elegida no tiene vecinos activos entonces se ignora, y el estado sigue siendo igual.

○ Desactivada      ● Activada

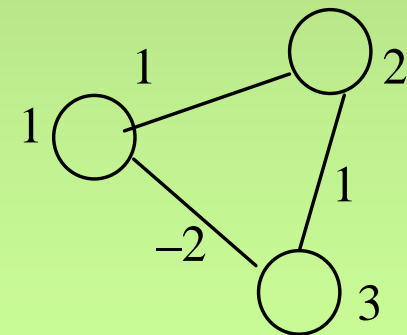
### 3 Estados estables



$X = [0 \ 1 \ 1]$ , para los nodos 1, 2 y 3



$X = [1 \ 1 \ 0]$



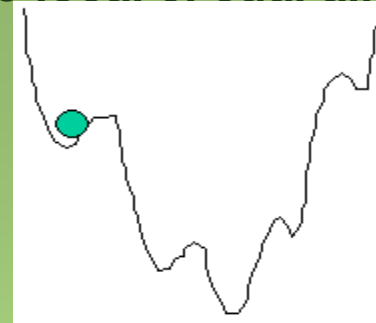
$X = [0 \ 0 \ 0]$



## Red de Hopfield: Método de calculo de los pesos



- **Nota:** La red converge a un mínimo local el cual almacena diferentes patrones.



**Almacena** M vectores N-dimensionales de patrones de entrada, a saber,  $\mathbf{x}_1, \dots, \mathbf{x}_M$

**Así, M es el n° de patrones almacenados, y utilizamos la regla de aprendizaje de Hebb para obtener los pesos de la red, para N nodos de la red:**

$$w_{ij} = \begin{cases} \frac{1}{N} \sum_{m=1}^M x_{mj} x_{mi} & \text{para todo } j \neq i, \text{ donde } i, j = 1, \dots, N \\ 0 & \text{para } j = i \end{cases}$$

$$\mathbf{W} = \frac{1}{N} \sum_{m=1}^M \mathbf{x}_m \mathbf{x}_m^T - \mathbf{M}\mathbf{I}, \text{ donde T es la transpuesta del vector}$$



## Metodo de computación de pesos



- Los pesos son determinados utilizando los  $M$  patrones de entrenamiento.

$$\mathbf{W} = \sum_{i=1}^M \mathbf{x}_i \mathbf{x}_i^T - M\mathbf{I}$$

Aquí

- $\mathbf{W}$  es la matriz de pesos
- $\mathbf{x}_i$  es un patrón de entrada representado por un vector de dimensión  $N$  con coordenadas en el conjunto  $\{-1, 1\}$ .
  - $N$  es el número de unidades (nodos) de la red, **que coincide con la dimensión de los vectores de entrada.** Los valores 1 y -1 representan las unidades (nodos) activas e inactivas respectivamente.
- $(\mathbf{x}_i)^T$  es el vector traspuesto del vector de entrada  $\mathbf{x}_i$ ,
- $M$  denota el **número de patrones** del conjunto de entrenamiento
- $\mathbf{I}$  es la matriz identidad de dimensión  $N \times N$



## Ejemplo 3



- Consideremos ahora una red de Hopfield con cuatro unidades y tres vectores del conjunto de entrenamiento que han de ser aprendidos por la red. Consideremos tres vectores del conjunto de entrenamiento entrada, a saber,  $\mathbf{x}_1$ ,  $\mathbf{x}_2$  y  $\mathbf{x}_3$  y que se definen de la siguiente manera:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}; \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}; \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}; \text{ los pesos de la red se calculan como } \mathbf{W} = \sum_{i=1}^M \mathbf{x}_i \mathbf{x}_i^T - M\mathbf{I}$$

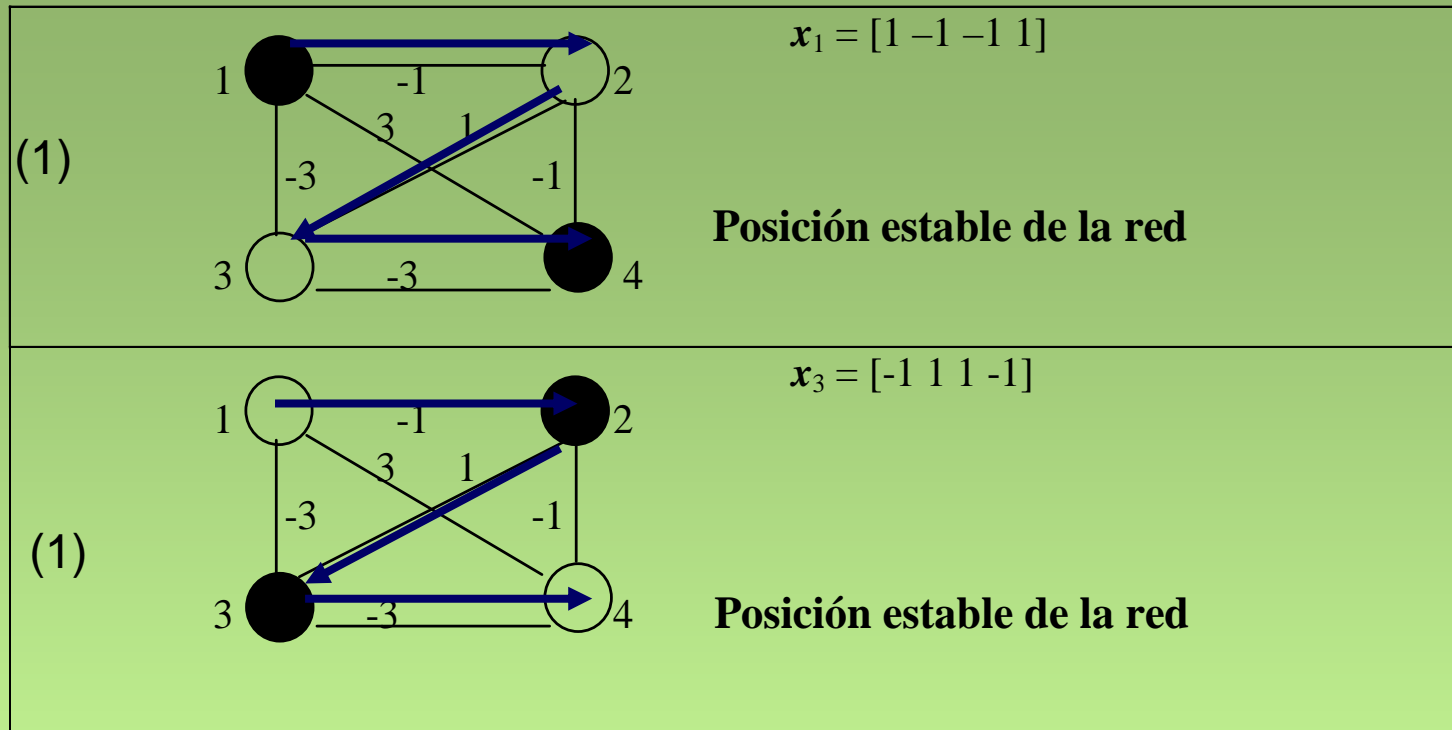
$$\mathbf{x}_1 \mathbf{x}_1^T = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}; \mathbf{x}_2 \mathbf{x}_2^T = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 \end{bmatrix}; \mathbf{x}_3 \mathbf{x}_3^T = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} - 3\mathbf{I} = \begin{bmatrix} 3 & -1 & -3 & 3 \\ -1 & 3 & 1 & -3 \\ -3 & 1 & 3 & -3 \\ 3 & -3 & -3 & 3 \end{bmatrix} - 3\mathbf{I}$$





## Ejemplo 3

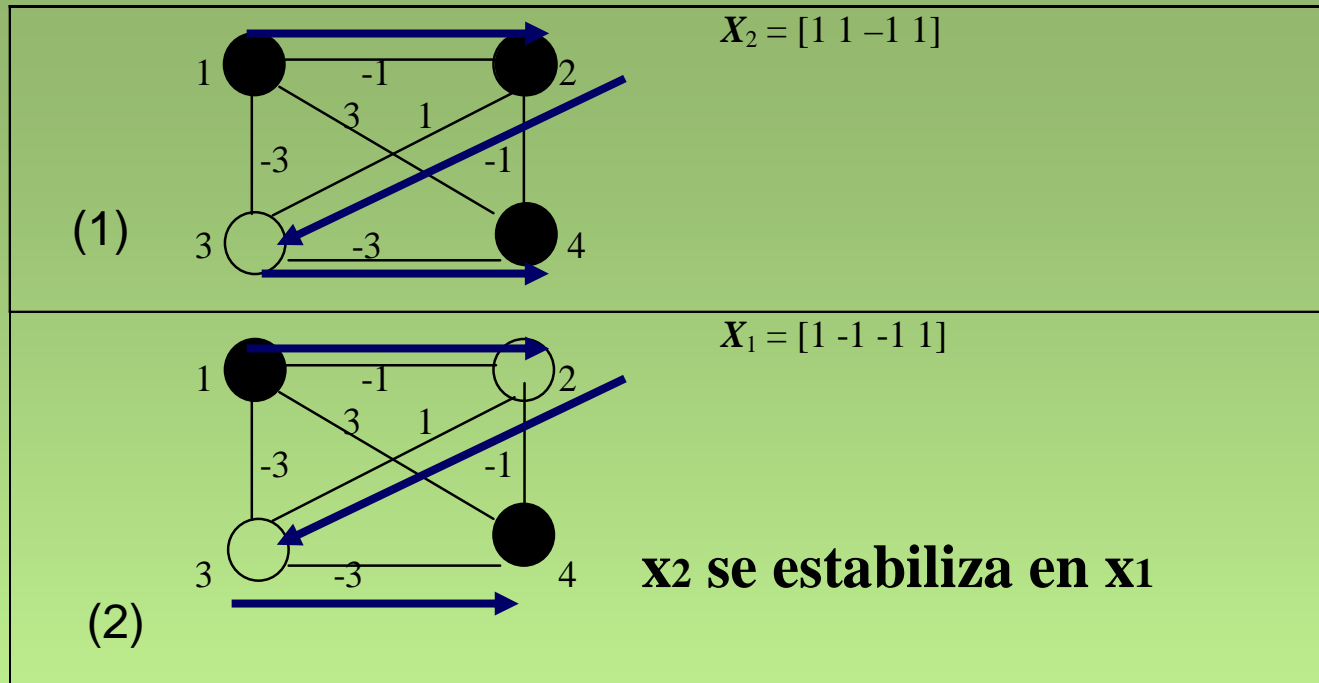


$$\mathbf{W} = \sum_{i=1}^M \mathbf{x}_i \mathbf{x}_i^T - M \mathbf{I} \quad (1)$$

$$\mathbf{W} = \begin{pmatrix} 3 & -1 & -3 & 3 \\ -1 & 3 & 1 & -1 \\ -3 & 1 & 3 & -3 \\ 3 & -1 & -3 & 3 \end{pmatrix} - \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 0 & -1 & -3 & 3 \\ -1 & 0 & 1 & -1 \\ -3 & 1 & 0 & -3 \\ 3 & -1 & -3 & 0 \end{pmatrix}$$



## Ejemplo 3



$$\begin{bmatrix} 0 & -1 & -3 & 3 \\ -1 & 0 & 1 & -3 \\ -3 & 1 & 0 & -3 \\ 3 & -3 & -3 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1+3+3 \\ -1-1-3 \\ -3+1-3 \\ 3-3+3 \end{bmatrix} \text{ pasa al } \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -3 & 3 \\ -1 & 0 & 1 & -3 \\ -3 & 1 & 0 & -3 \\ 3 & -3 & -3 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ -5 \\ -7 \\ 9 \end{bmatrix} \text{ pasa al } \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$



## Ejemplo 3



- Las redes generadas utilizando estos pesos y los vectores de entrada son estables, a excepción de

$$\mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

- $\mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$  se estabiliza en  $\mathbf{x}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$  (el cual está a una distancia de Hamming de 1).

- Así, con los pesos obtenidos y los estados estables ( $\mathbf{x}_1$  y  $\mathbf{x}_3$ ), podemos estabilizar cualquier nuevo patrón (parcialmente) a uno de ellos

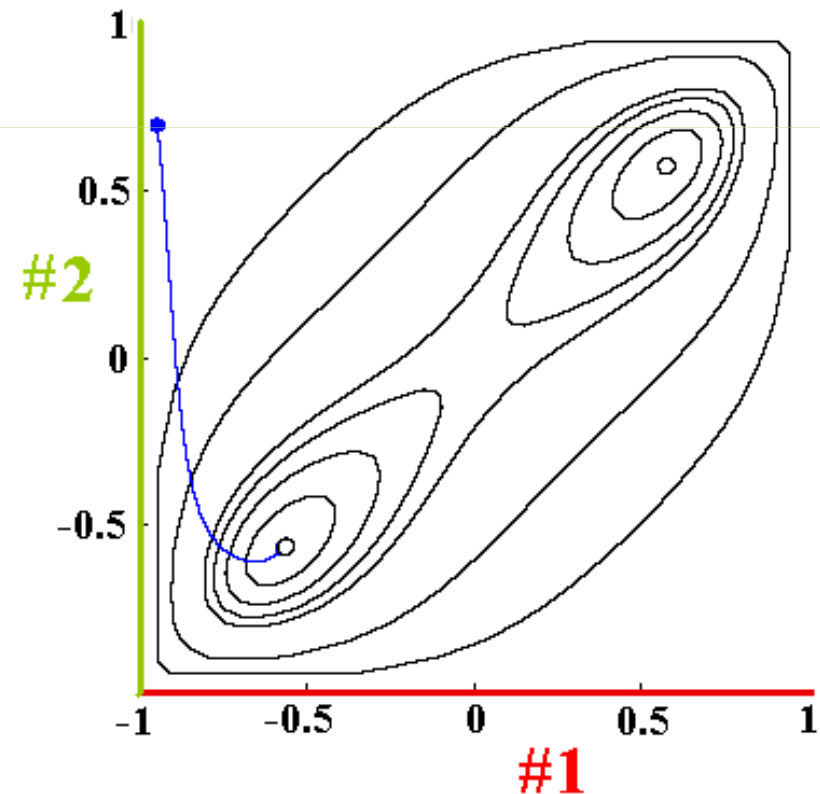
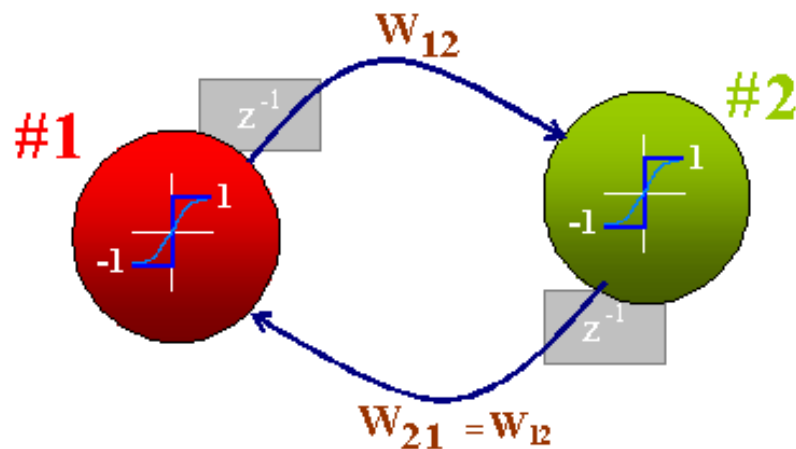


## Ejemplo 4



Red de Hopfield con 2-neuronas de estados  
contínuos caracterizada por dos estados estables

Dibujo del contorno

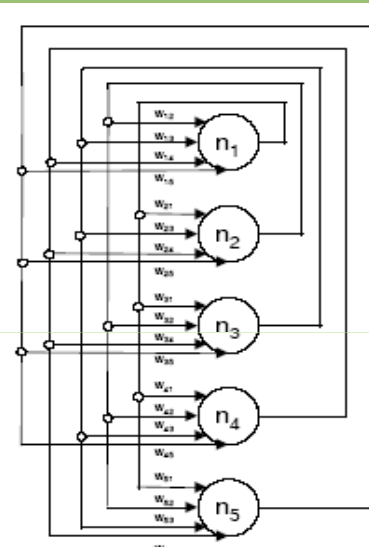
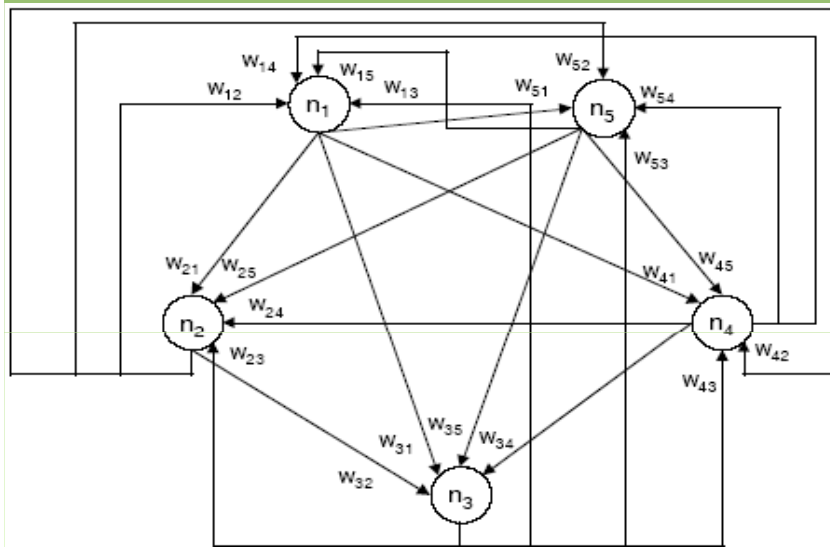




# Sistema dinámico



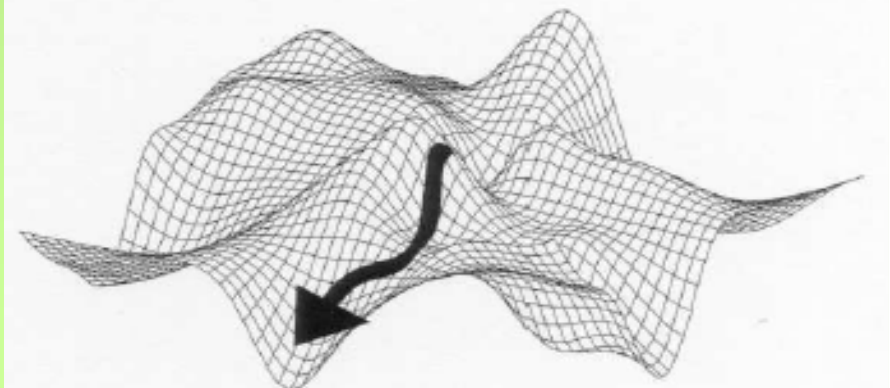
El comportamiento de un sistema dinámico de este tipo está totalmente determinado por los pesos sinápticos



$$W_{ij} = W_{ji}$$

0	$W_{12}$	$W_{13}$	$W_{14}$	$W_{15}$
$W_{21}$	0	$W_{23}$	$W_{24}$	$W_{25}$
$W_{31}$	$W_{32}$	0	$W_{34}$	$W_{35}$
$W_{41}$	$W_{42}$	$W_{43}$	0	$W_{45}$
$W_{51}$	$W_{52}$	$W_{53}$	$W_{54}$	0

Se puede pensar como un proceso de minimización de la energía. El sistema dinámico correspondiente evoluciona hacia estados de baja energía

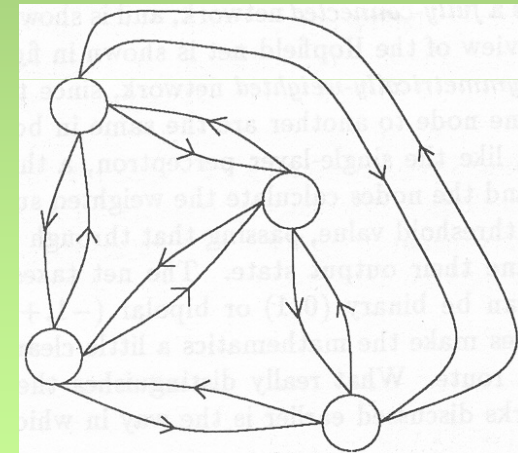
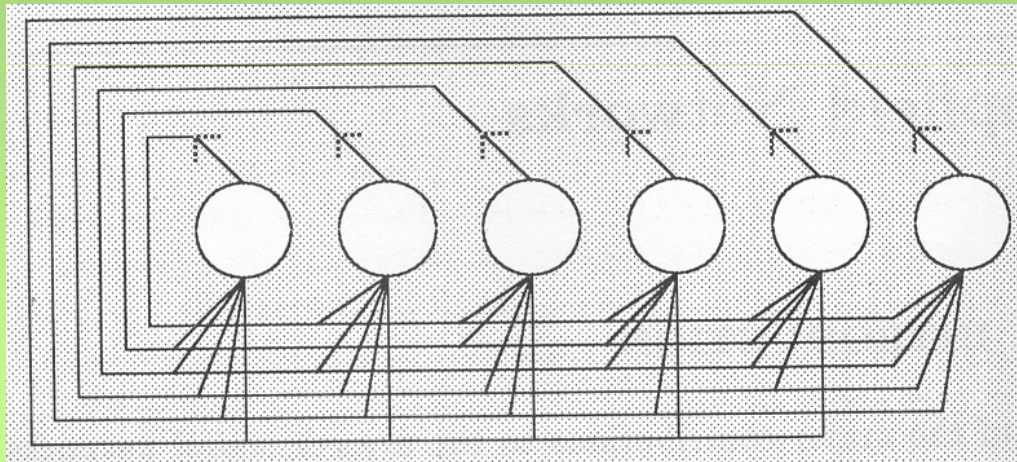




## Sin autoconexiones



Las redes de Hopfield están completamente conectadas, son redes simétricamente ponderadas que extienden las ideas de memorias lineales asociativas mediante la adición de conexiones cíclicas





# Funcionamiento de la red



Después de la "etapa de enseñanza-aprendizaje", en la que se definen los pesos, se establece el estado inicial de la red (patrón de entrada) y una regla sencilla recurrente se itera hasta que se alcance la convergencia a un estado estable (patrón de salida)

En cuanto al entrenamiento de una red Hopfield como una memoria de contenido direccionable se utiliza la regla del producto externo,  $\mathbf{xx}^T$ , para el almacenamiento de patrones.

Hay dos modos principales de funcionamiento del entrenamiento:

Actualización síncrona versus asíncrona



## Hopfield Network Algorithm



### 1. Assign connection weights

$$w_{ij} = \begin{cases} \frac{1}{N} \sum_{s=0}^{M-1} x_i^s x_j^s & i \neq j \\ 0 & i = j, \end{cases}$$

**Aprendizaje Hebbiano**

where  $w_{ij}$  is the connection weight between node  $i$  and node  $j$ , and  $x_i^s$  is element  $i$  of the exemplar pattern  $s$ , and is either  $+1$  or  $-1$ . There are  $M$  patterns, from  $0$  to  $M - 1$ , in total.

### 2. Initialise with unknown pattern

$$\mu_i(0) = x_i \quad 0 \leq i \leq N - 1$$

**Patrón de prueba**

where  $\mu_i(t)$  is the output of node  $i$  at time  $t$ .

### 3. Iterate until convergence

$$\mu_i(t + 1) = f_h \left[ \sum_{j=0}^{N-1} w_{ij} \mu_j(t) \right] \quad 0 \leq j \leq N - 1$$

**Evolución dinámica**

The function  $f_h$  is the hard-limiting non-linearity, the step function,  
Repeat the iteration until the outputs from the node  
remain unchanged.

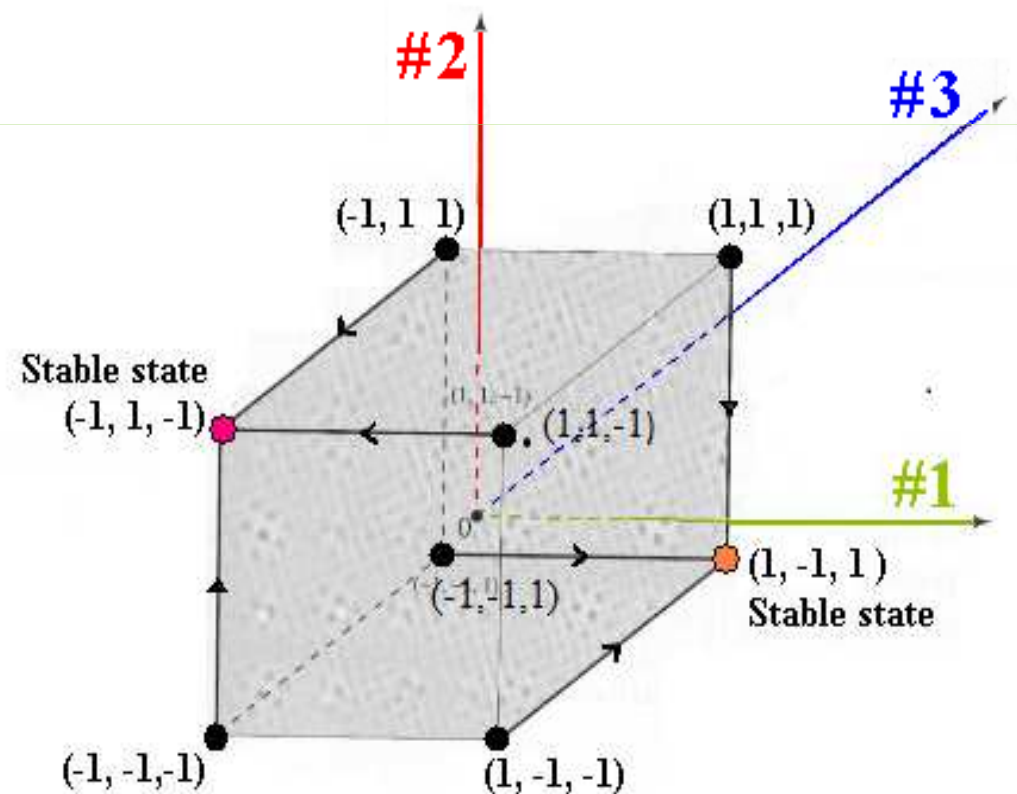
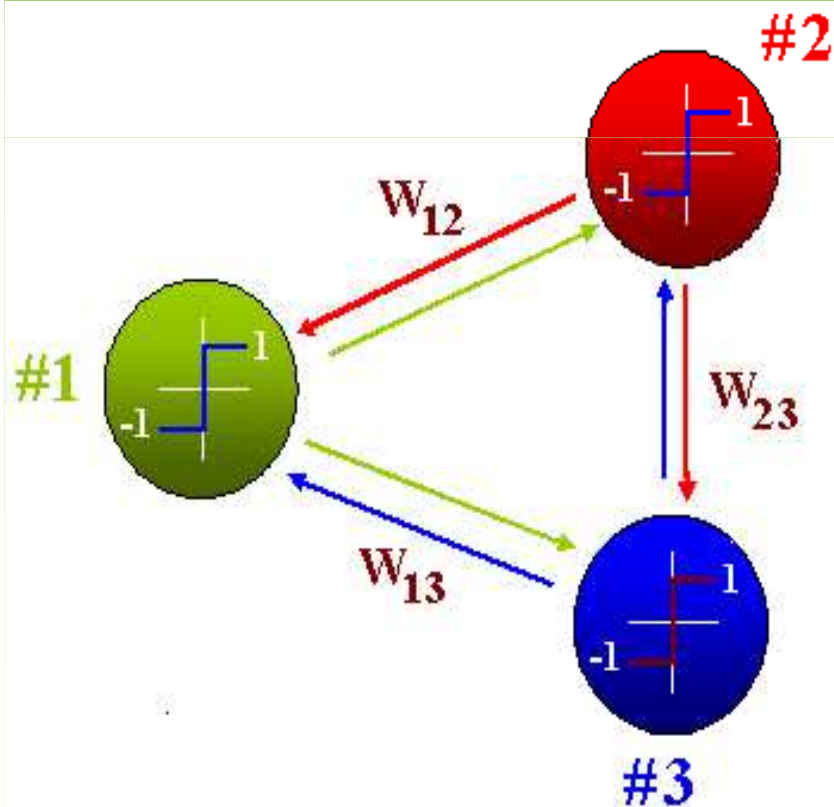




## Nuevo Ejemplo



Red de Hopfield con 3-neuronas o nodos de  
8 estados caracterizada por dos estados  
estables





# Un ejemplo sencillo



## Etapa 1.

**Diseñar una red con patrones (vectores) memorizados  
[ 1, -1, 1 ] y [ -1, 1, -1 ] y estables, el resto inestables**

- Compute the outer products, sum, normalize, and set diagonals to 0:

$$(1, -1, 1)^T * (1, -1, 1) + (-1, 1, -1)^T * (-1, 1, -1) =$$

$$\mathbf{W} = \sum_{i=1}^M \mathbf{x}_i \mathbf{x}_i^T - M \mathbf{I}, \text{ o } \mathbf{W} = \frac{1}{N} \sum_{i=1}^M \mathbf{x}_i \mathbf{x}_i^T - \frac{M}{N} \mathbf{I}$$

$$\begin{bmatrix} 2 & -2 & 2 \\ -2 & 2 & -2 \\ 2 & -2 & 2 \end{bmatrix} \xrightarrow{\frac{1}{3}} \begin{bmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{bmatrix}$$

$\frac{1}{3} = 1/(\text{number of neurons})$

Force Diagonal to 0

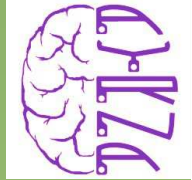
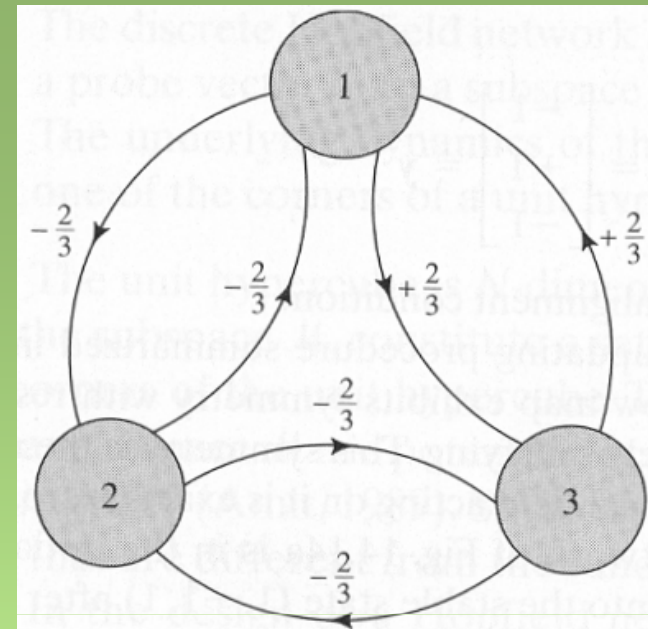
**O lo que es igual**

$$\mathbf{W} = \frac{1}{3} \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} [+1, -1, +1] + \frac{1}{3} \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} [-1, +1, -1] - \frac{2}{3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



## Etapa 2. Inicialización

$$\mathbf{W} = \frac{1}{3} \begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}$$



Hay 8 estados diferentes que pueden ser alcanzados por la red y por lo tanto, cualquiera se puede utilizar como su estado inicial #1:  $y_1$

#2:  $y_2$

#3:  $y_3$

$$\begin{bmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix}$$

↑ ↑  
Patrones iniciales, estados estables

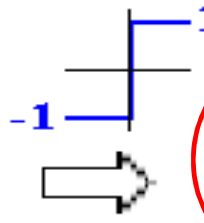


## Etapa 3.- Iterar hasta la convergencia

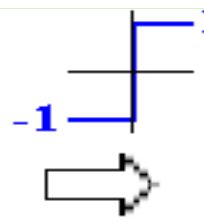


### - Cambio Síncrono-

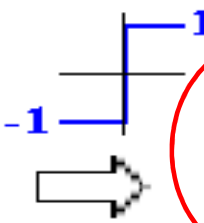
3 ejemplos diferentes del flujo de la red

$$\frac{1}{3} \begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 0 \\ -4 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$$

Cambia sólo  
el del medio,  
pasa a estado estable

$$\frac{1}{3} \begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} -4 \\ 4 \\ -4 \end{pmatrix}$$

$$\begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix}$$

No cambia,  
estado estable

$$\frac{1}{3} \begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$$

Cambia sólo  
el primero,  
pasa a estado estable

**Converge inmediatamente**



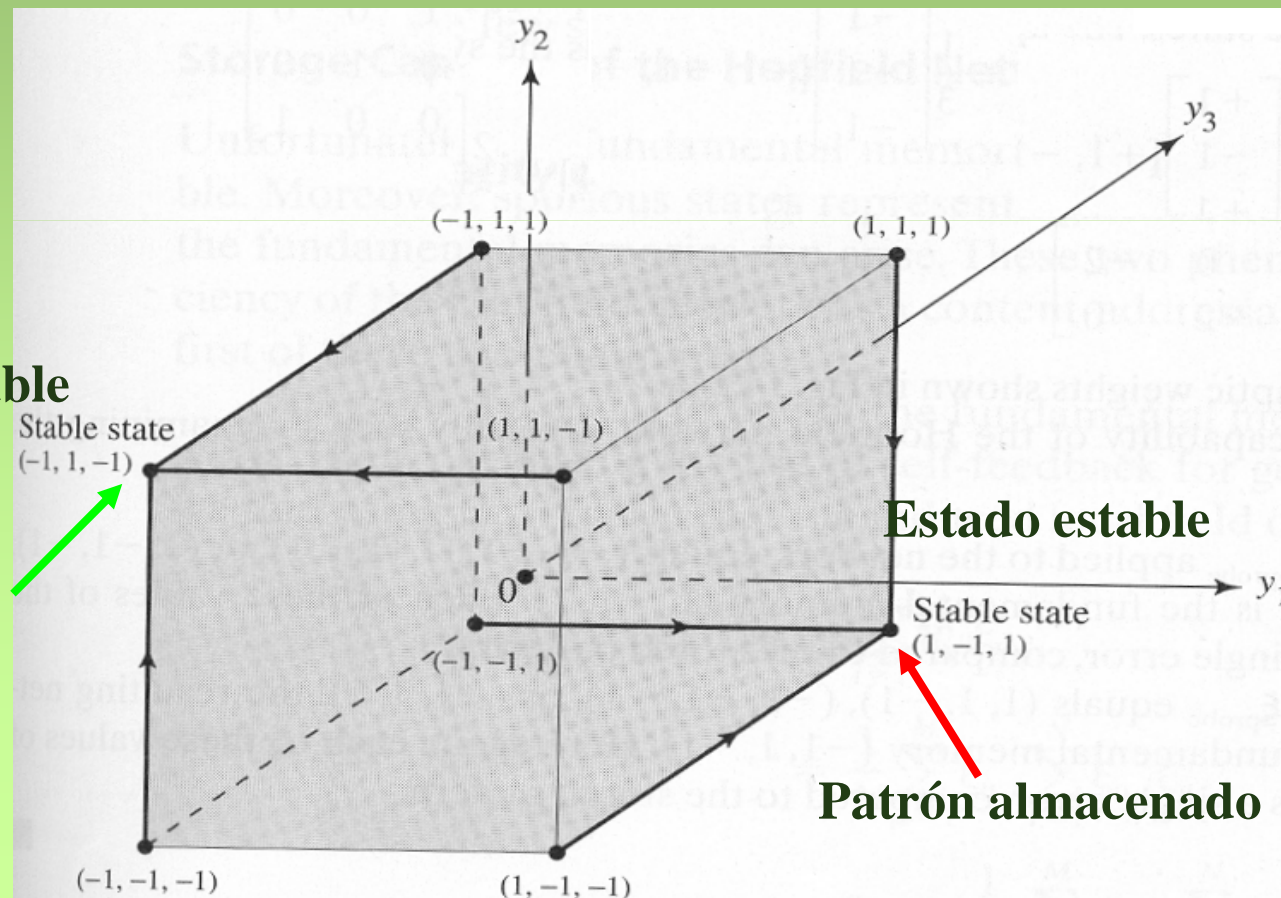
## Etapa 3.- Iterar hasta la convergencia



### - Cambio Síncrono-

Diagrama esquemático de todas las trayectorias dinámicas que corresponden a la red diseñada.

**Estado estable**





## Etapa 3.- Iterar hasta la convergencia



### - Cambio Asíncrono-

Cada vez, seleccione una neurona al azar y actualice su estado con la regla anterior

Y el convenio general de que si la entrada total a la neurona es 0 su estado permanece sin cambios

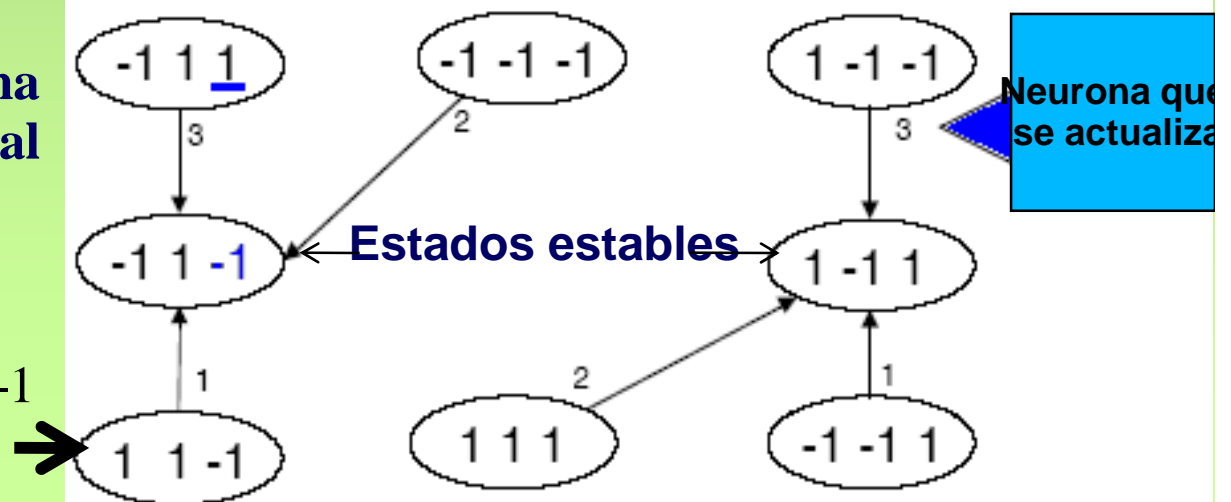
Ahora si seleccionamos la neurona primera en (1,1,-1) pasamos al estado estable (-1,1,-1)

$$\frac{1}{3}(0, -2, 2) \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = \frac{1}{3}(-4), \text{ cambia 1 por -1}$$

possible initial states

$$\frac{1}{3} \begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix} \begin{pmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & \underline{1} & -1 & 1 & -1 & 1 \end{pmatrix}$$

$$\frac{1}{3} \begin{pmatrix} 2 & -2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ \underline{1} \end{pmatrix} = \frac{1}{3}(-4) \xrightarrow{\text{thresholding}} -1$$





**MODELOS COMPUTACIONALES: CUARTO CURSO  
DEL GRADO  
DE ING. INFORMÁTICA EN COMPUTACION**



# **MODELOS DE REDES NEURONALES**

**César Hervás-Martínez  
Pedro A. Gutiérrez Peña  
Grupo de Investigación AYRNA**

**Departamento de Informática y Análisis  
Numérico  
Universidad de Córdoba  
Campus de Rabanales. Edificio Einstein.  
Email: [chervas@uco.es](mailto:chervas@uco.es)  
[pagutierrez@uco.es](mailto:pagutierrez@uco.es)**

**2021-2022**