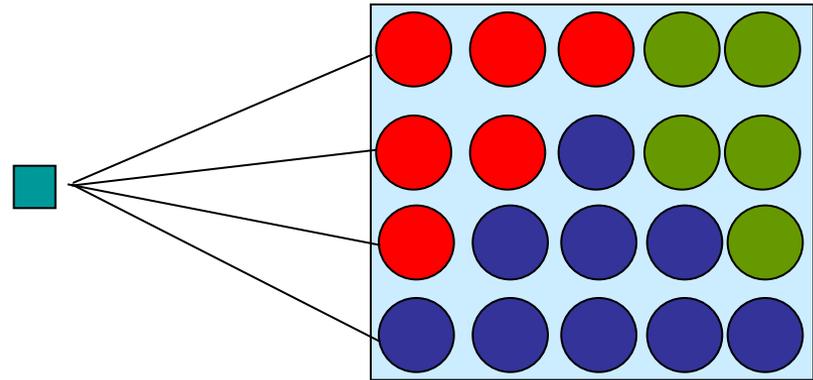




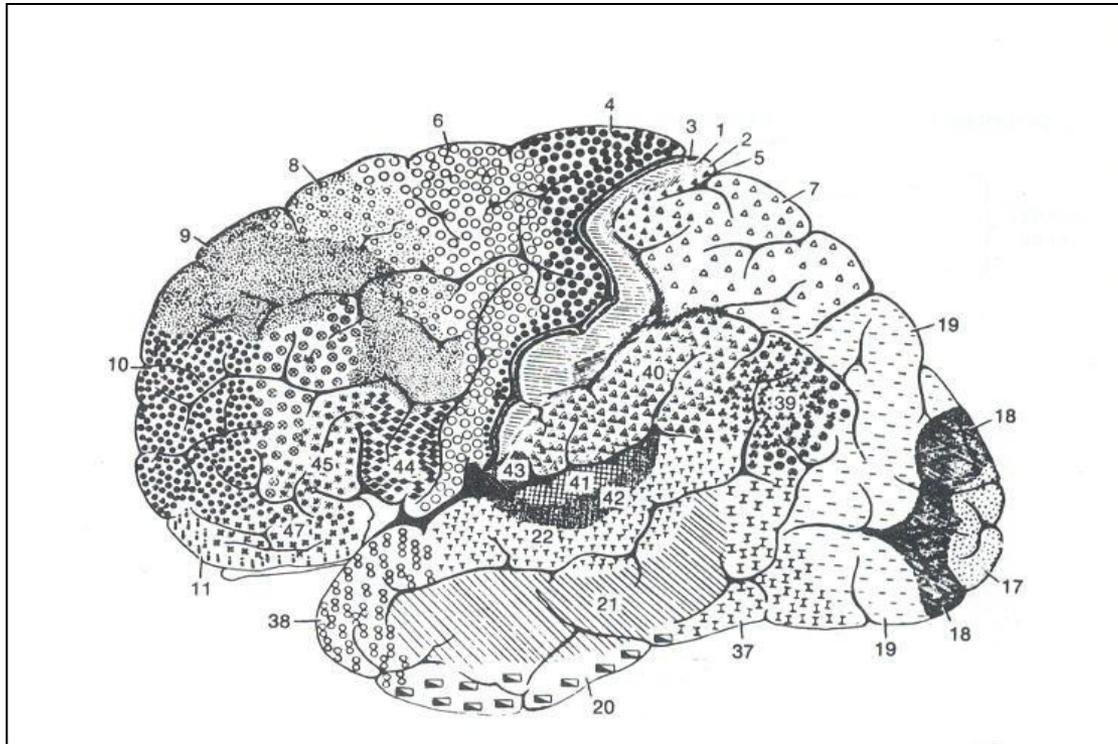
REDES NEURONALES NO SUPERVISADAS





El cerebro tiene la capacidad de auto-organizarse a partir de los estímulos recibidos y esto lo logra de manera no supervisada.

Diferentes estímulos son enviados y procesados en partes específicas del cerebro. Por ejemplo, un estímulo visual y otro auditivo son dirigidos a regiones distintas en el cerebro.





Los algoritmos de entrenamiento no supervisados pueden ser divididos en dos grandes categorías:

Hard Competitive learning

Las neuronas en la red compiten entre si y sólo una de ellas (la neurona ganadora) produce la respuesta al estímulo.

Durante el entrenamiento sólo los pesos sinápticos de la neurona ganadora se actualizan

No existe interacción entre neuronas vecinas

Soft Competitive learning

Las neuronas en la red compiten entre si y sólo una de ellas (la neurona ganadora) produce la respuesta al estímulo.

Durante el entrenamiento los pesos sinápticos de la neurona ganadora y sus vecinas se actualizan

Existe interacción entre neuronas vecinas



Mecanismo de Competencia

Consiste en considerar una medida de similitud y buscar la neurona cuyo vector de pesos sinápticos se asemeja más al estímulo presentado a la red.

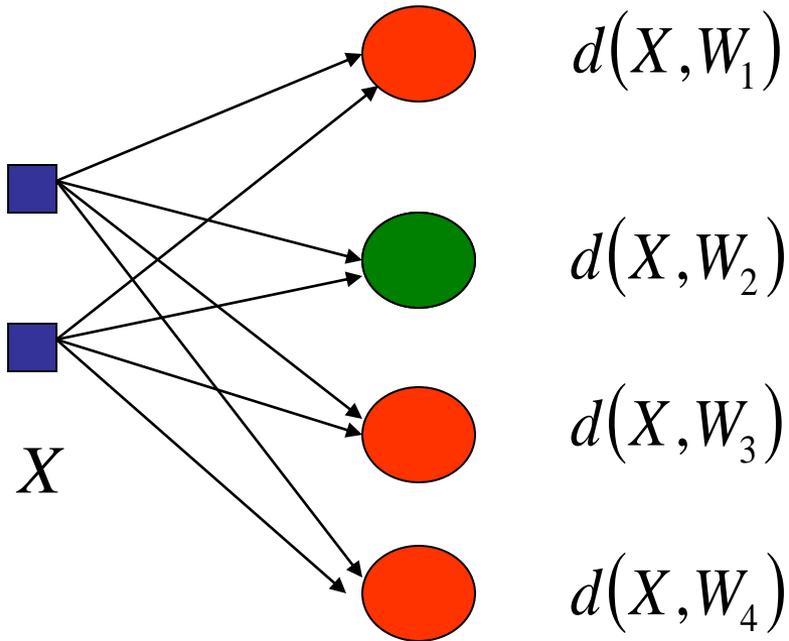
La medida de similitud usada varía dependiendo de las aplicaciones y entre las más usadas se encuentran:

La distancia Euclídea usual: $d(X, W_j) = \|X - W_j\| \quad j = 1, 2, \dots, N$

En este caso la neurona ganadora es aquella para la cual la distancia es la más pequeña

Producto interno usual: $d(X, W_j) = W_j^t X \quad j = 1, 2, \dots, N$

En esta caso la neurona ganadora es aquella para la cual la distancia es la más grande



Neurona Ganadora

Este proceso puede ser bastante lento cuando la cantidad de neuronas es grande !



Mecanismo de Actualización

Consiste en mover en cierto grado el vector de pesos sinápticos para que se asemeje al patrón o estímulo de entrada.

Para ello se considera:

$$W(n+1) = W(n) + \Delta W(n)$$

$$\Delta W(n) = \eta (X(n) - W(n)) \quad 0 < \eta < 1$$

La tasa o constante de aprendizaje puede ser constante o función del número de épocas. Esto es,

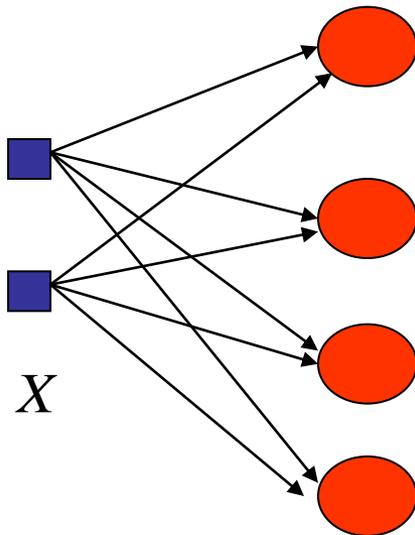
$$\eta = \eta(n)$$



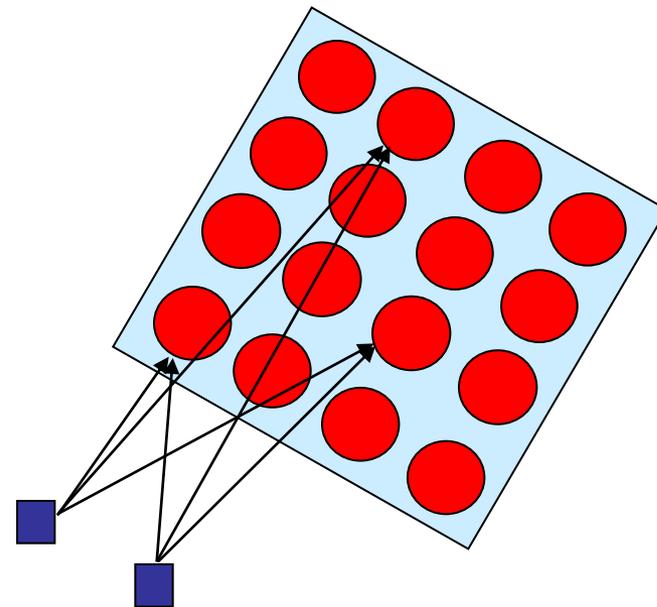
Configuración de la red

Generalmente se utiliza una red con una sola capa de neuronas o un arreglo 2D de neuronas, dependiendo de la aplicación. Esta configuración puede ser estática(fija) o dinámica (cambiante)

Para efectos de visualización, es preferible trabajar con un arreglo de neuronas en 2D.



1D



2D



Respuesta de la red

Dependiendo de la aplicación, la respuesta puede ser:

- **El Vector de pesos sinápticos de la neurona ganadora**

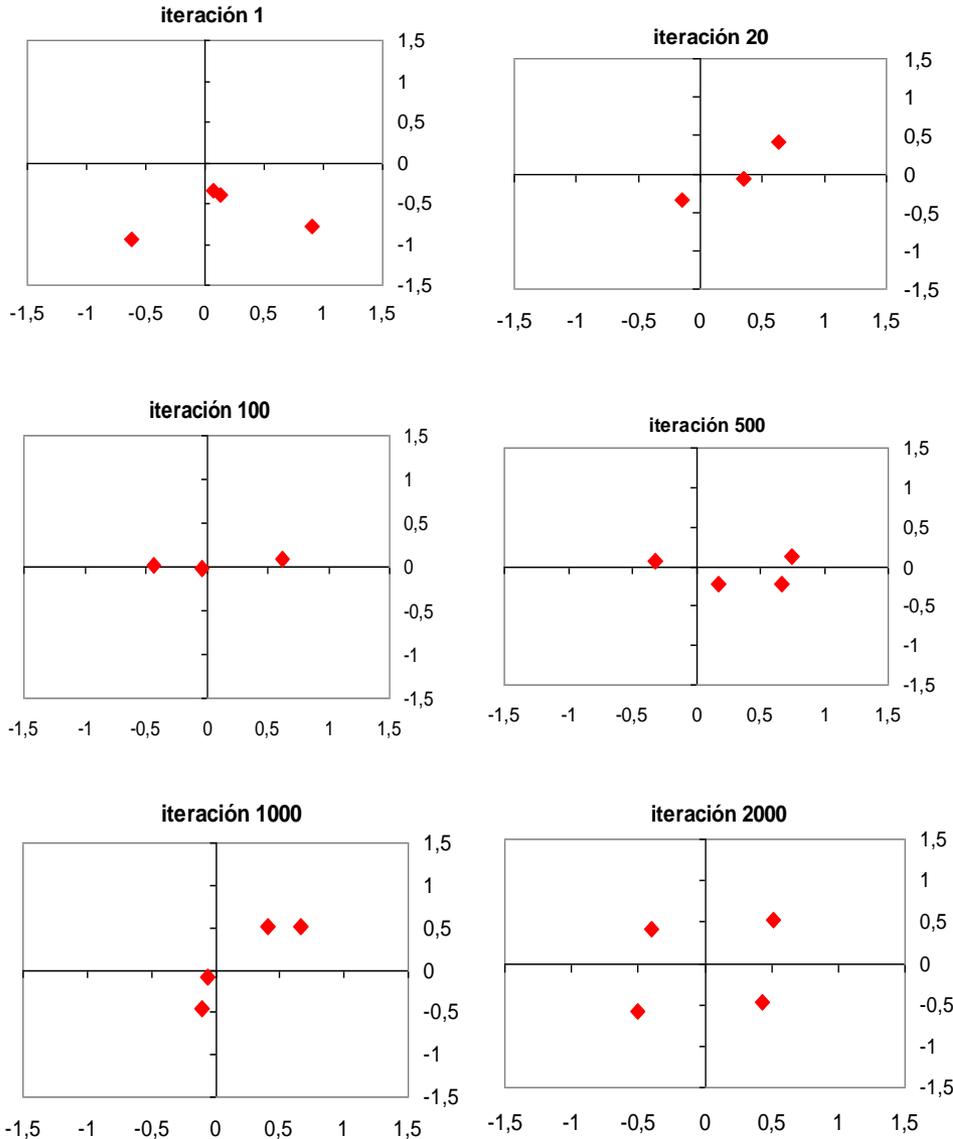
Permite detectar cluster en los estímulos presentados a la red.

- **El índice de la neurona ganadora**

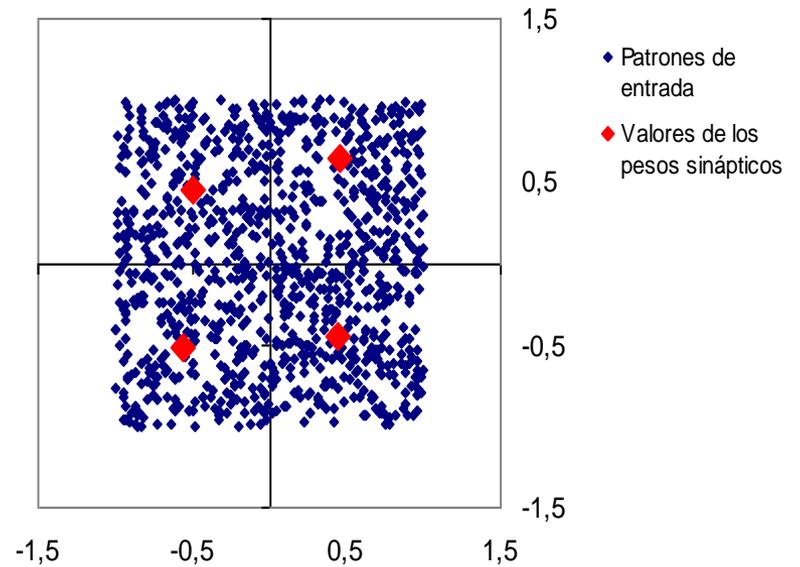
Permite detectar cluster en los estímulos presentados a la red.

Permite visualizar agrupamiento de los estímulos o patrones cuando estos se encuentran en dimensiones superiores

Esto se logra asignando a cada estímulo el índice de la neurona ganadora que lo representa



Red 1D, 4 Neuronas



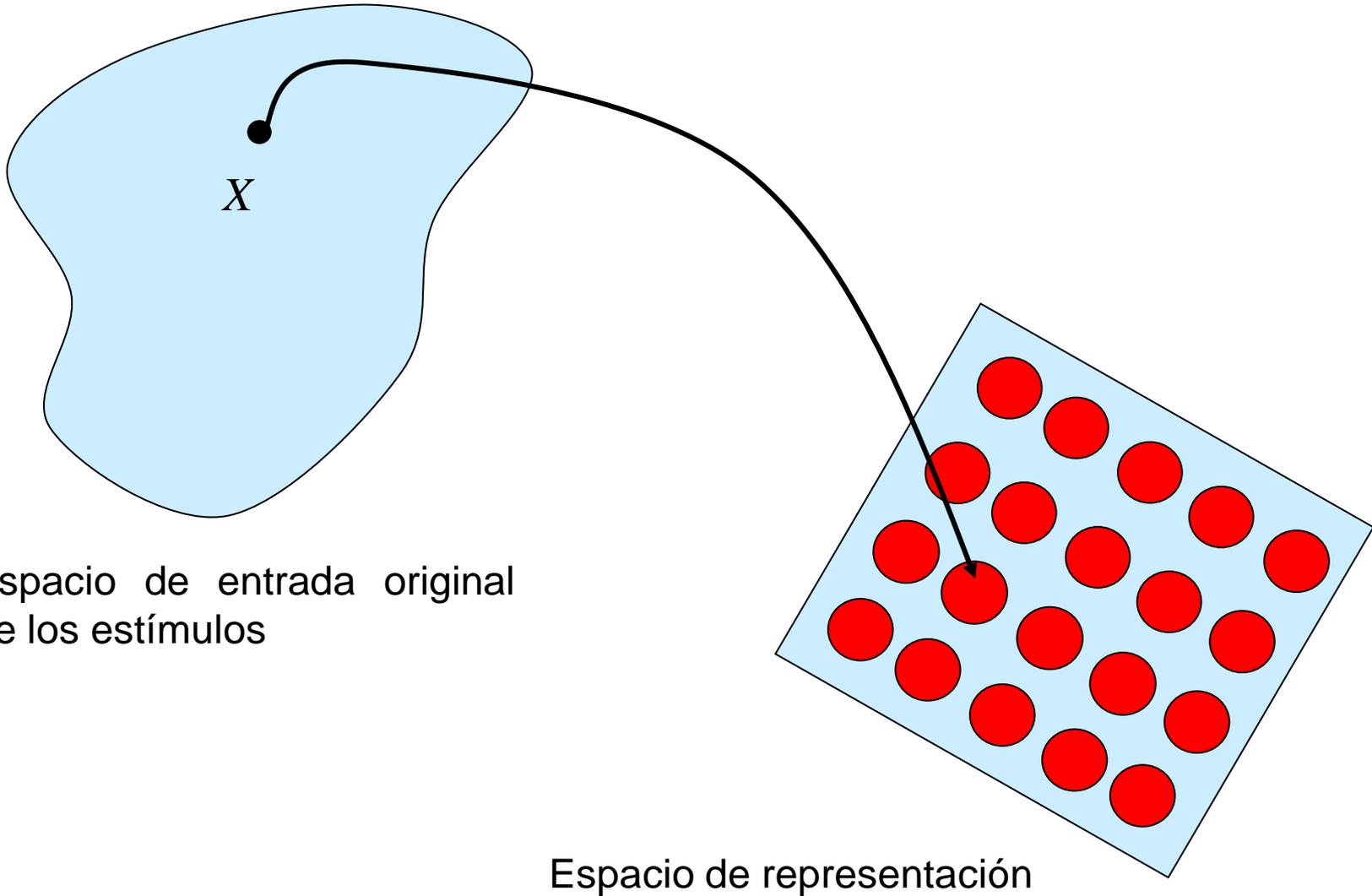
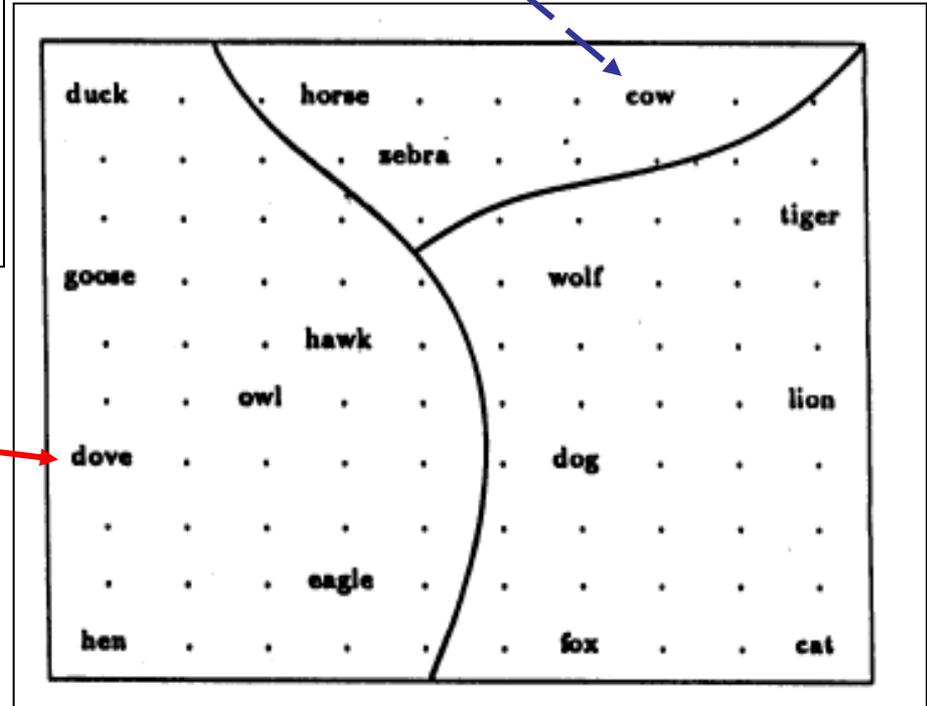




Table 3.4. Animal names and their attributes

		dove	hen	duck	goose	owl	hawk	eagle	fox	dog	wolf	cat	tiger	lion	horse	zebra	cow
is	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes to	hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	fly	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0





HARD COMPETITIVE LEARNING



El algoritmo básico del HCL es:

- 1. Escoger un estímulo o patrón de entrada $X(n)$**
- 2. Determinar la neurona ganadora i^***

Esta corresponde al índice i^* para el cual

$$\|X - W_{i^*}\| \leq \|X - W_j\| \quad \forall j$$

- 3. Actualizar el vector de pesos sinápticos de la neurona i^***

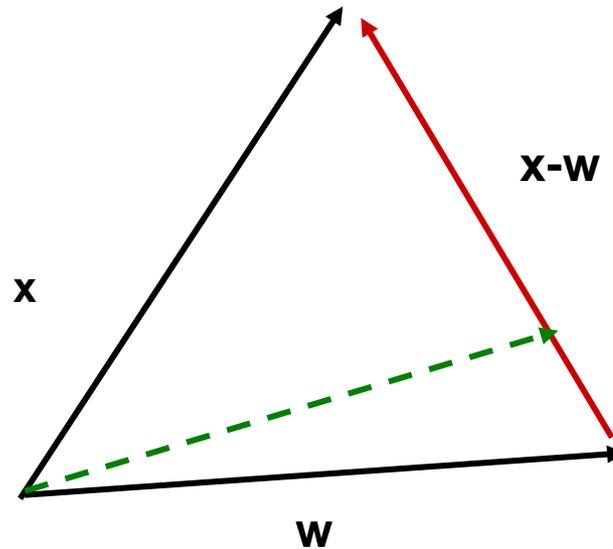
$$W_{i^*}(n+1) = W_{i^*}(n) + \eta(n)(X(n) - W_{i^*}(n))$$

- 4. Repetir hasta satisfacer la condición de parada**



Geoméricamente este algoritmo acerca a los pesos hacia los patrones.

$$W(n+1) = W(n) + \eta(n)(X(n) - W(n))$$



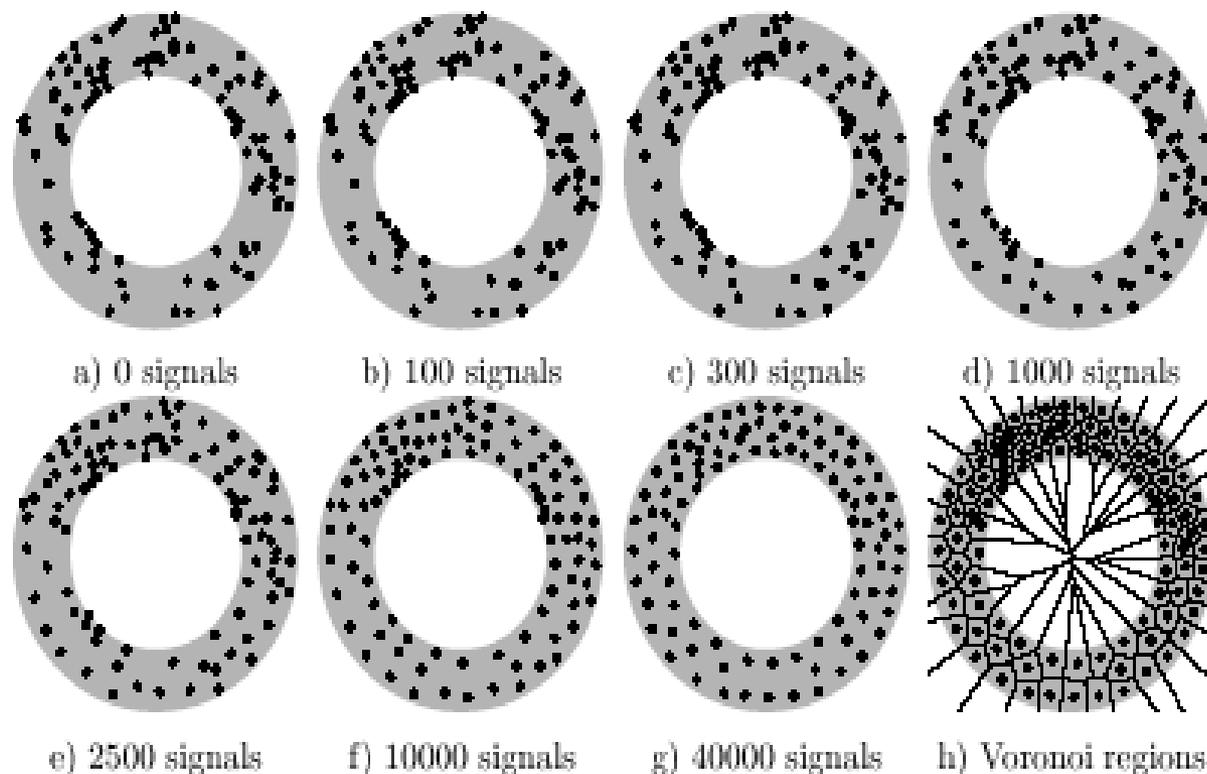
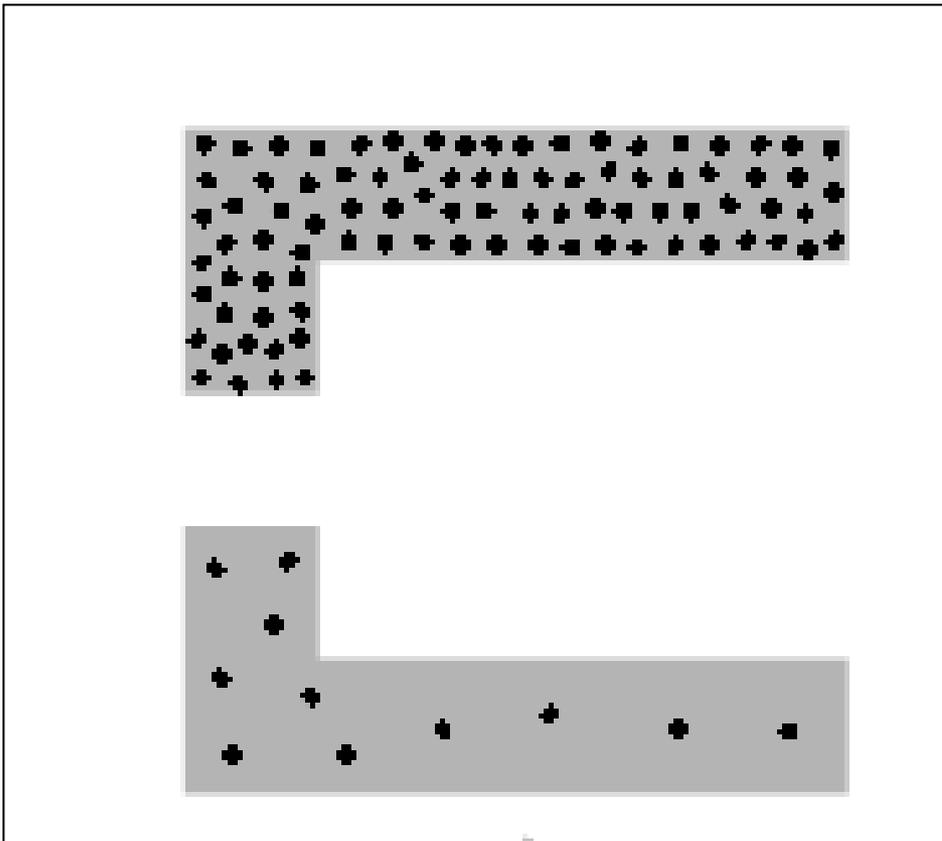


Figure 4.3: *Hard competitive learning* simulation sequence for a ring-shaped uniform probability distribution. A constant adaptation rate was used. a) Initial state. b-f) Intermediate states. g) Final state. h) Voronoi tessellation corresponding to the final state.



La desventaja del algoritmo es que algunas neuronas pudieran ser actualizadas menos frecuentemente que otras o incluso no ser actualizadas.



Los estímulos tienen distribución uniforme en la región y sin embargo la densidad de puntos es 10 veces mayor en la parte superior



REDES DE KOHONEN



Las redes de kohonen son redes no supervisadas en 1D o 2D entrenadas con un algoritmo de entrenamiento del tipo *soft competitive learning*

Tienen la ventaja de que todas las neuronas son actualizadas en cierta medida. Esto se logra utilizando una vecindad cambiante con el tiempo

El uso de la vecindad hace que la red respete la topología de los datos o estímulos. Es decir, estímulos similares son *mapeados* en neuronas próximas y estímulos completamente similares son *mapeados* en neuronas distantes

Esto permite utilizar estas redes para extraer patrones y características resaltantes en los estímulos



El algoritmo básico entrenamiento de una red de kohonen es:

- 1. Escoger un estímulo o patrón de entrada $X(n)$**
- 2. Determinar la neurona ganadora i^***

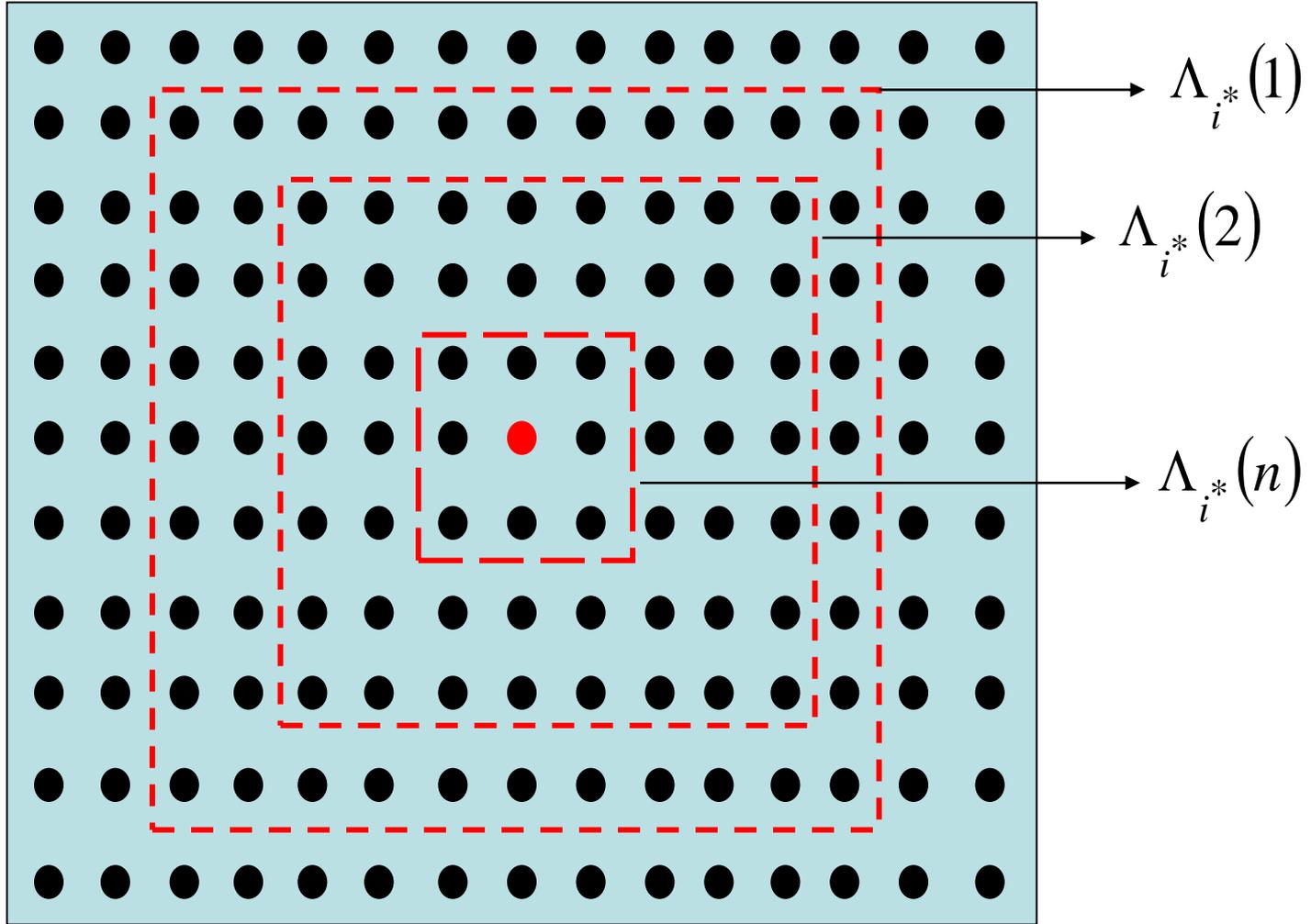
Esta corresponde al índice i^* para el cual

$$\|X - W_{i^*}\| \leq \|X - W_j\| \quad \forall j$$

- 3. Actualizar el vector de pesos sinápticos de la neurona i^* y de todas las neuronas en la vecindad $\Lambda_{i^*}(n)$ utilizando la ecuación**

$$W(n+1) = W(n) + \eta(n)(X(n) - W(n))$$

- 4. Repetir hasta satisfacer la condición de parada**



Esta característica evita el *overfitting* de la red !



La vecindad se puede definir considerando:

$$h_{i^*,j}(n) = \exp\left(-d_{i^*,j}^2 / 2\sigma^2(n)\right)$$

Y de esta forma la corrección de los pesos sinápticos se puede resumir como

$$W_j(n+1) = W_j(n) + \eta(n)h_{i^*,j}(n)(X(n) - W(n))$$



El entrenamiento se divide en dos fases:

fase de ordenamiento:

Durante esta fase ocurre el ordenamiento topológico inicial. Esta fase puede llegar a durar unas **1000 épocas**. Aquí debemos cuidar de elegir convenientemente:

- **La tasa de aprendizaje:** debe estar cercana a 0.1 inicialmente y decrecer hasta más o menos 0.01.
- **Tamaño de la vecindad:** inicialmente debería abarcar casi la totalidad de las neuronas y gradualmente decrecer.

fase de convergencia:

es una etapa para afinar la red, hacen falta unas **500 veces el número de neuronas** en épocas para entrenar.

- **La tasa de aprendizaje** debe mantenerse cercano a 0.01 (no permitir decrecer hasta 0).
- **Tamaño de la vecindad:** seguir decreciendo hasta algunas pocas o solo abarcar la neurona ganadora.



Para la fase de ordenamiento, una forma usual de escoger la tasa de aprendizaje es considerar

$$\eta(n) = \eta_0 \exp(-n / \tau_2)$$

Esta escogencia también evita el *overfitting* de la red y las constantes se escogen para satisfacer ciertos requerimientos durante la fase de organización de la red. Algunos valores sugeridos:

$$\eta_0 = 0.1$$

$$\tau_2 = 1000$$

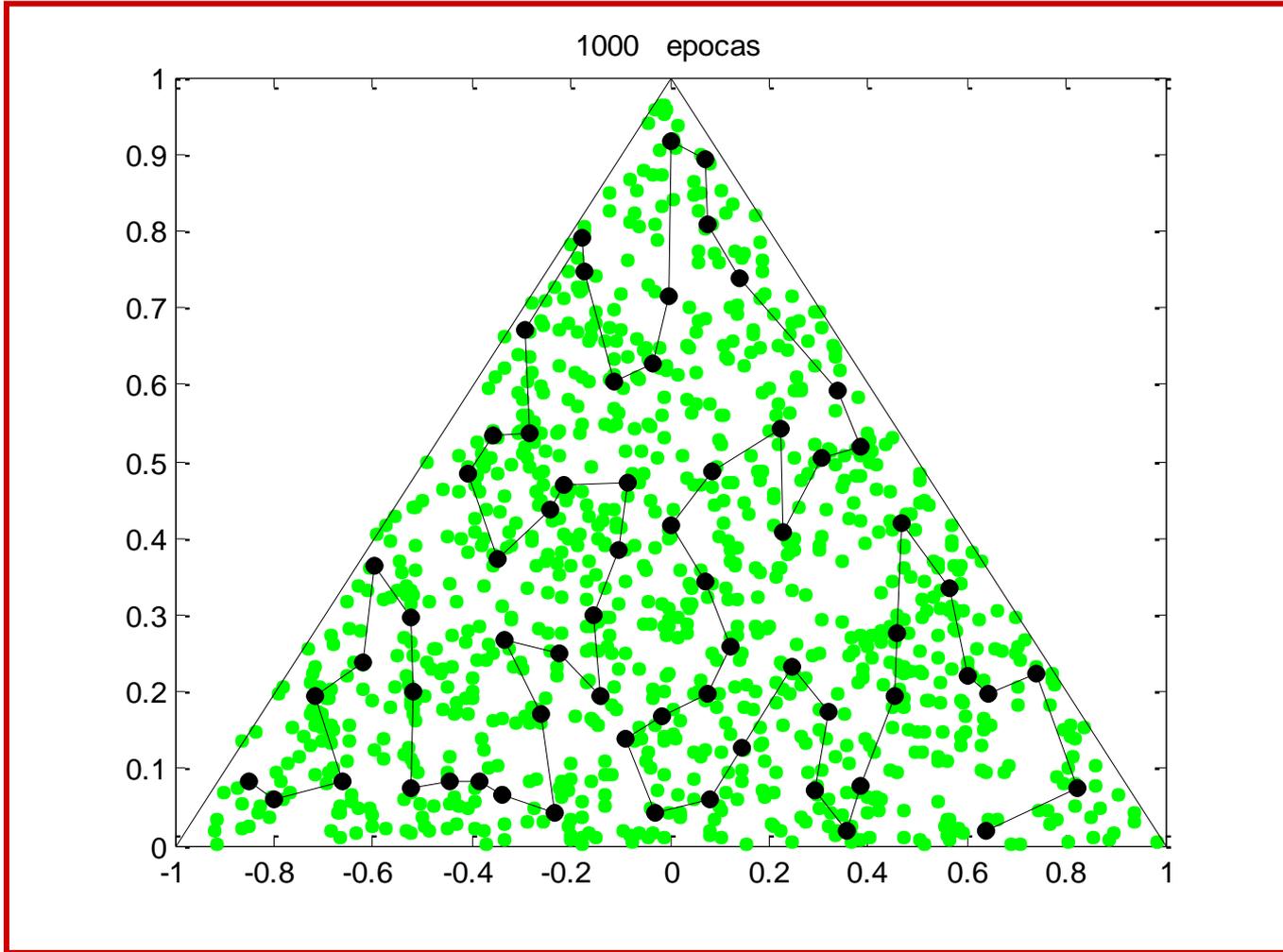
La vecindad se puede definir con radios cada vez más pequeños (decrecen con las épocas).

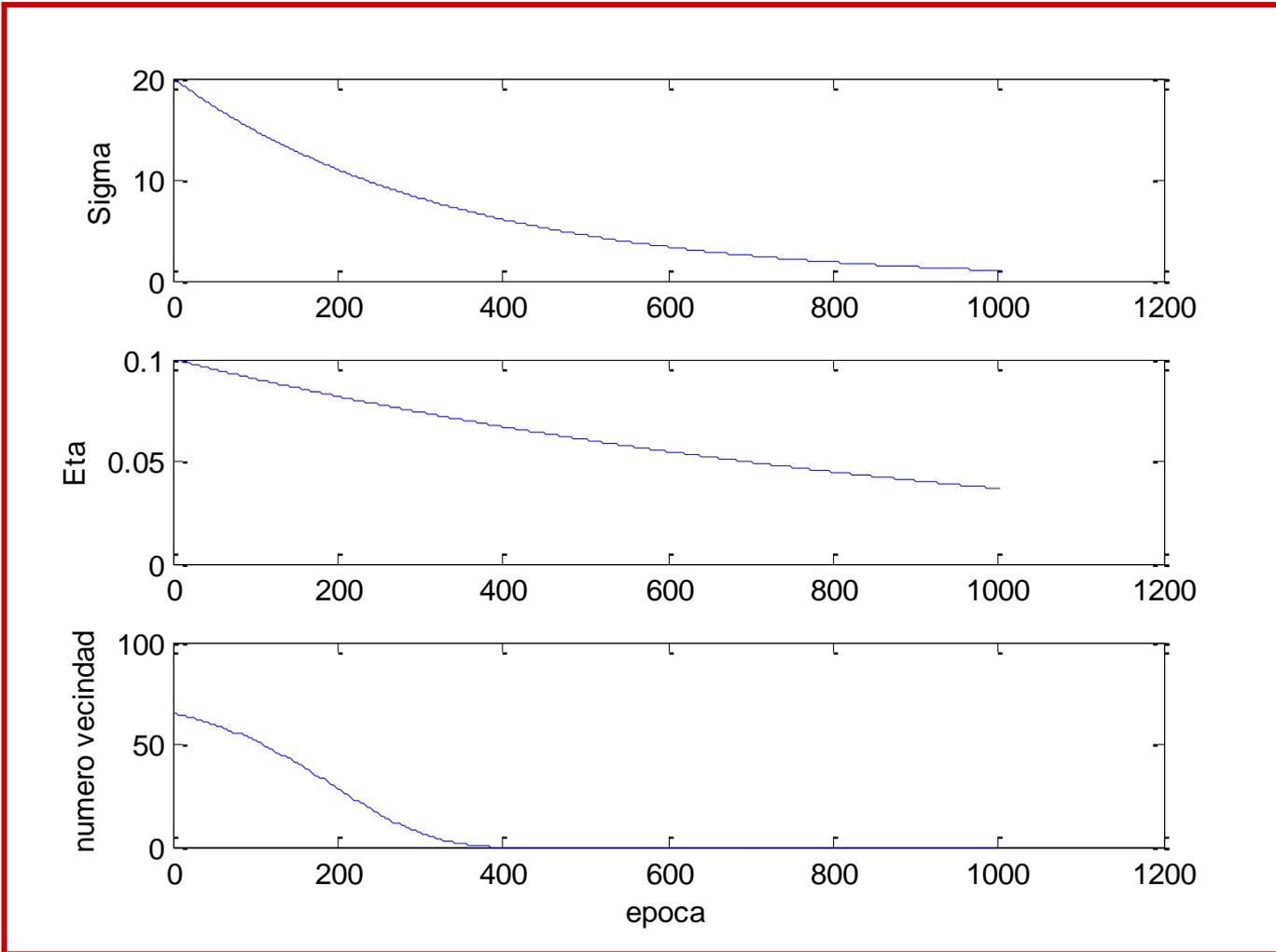
$$\sigma(n) = \sigma_0 \exp(-n / \tau_1)$$

$$\tau_1 = 1000 / \log \sigma_0$$



Un decrecimiento muy rápido de la vecindad podría afectar el “despliegue” del mapa.

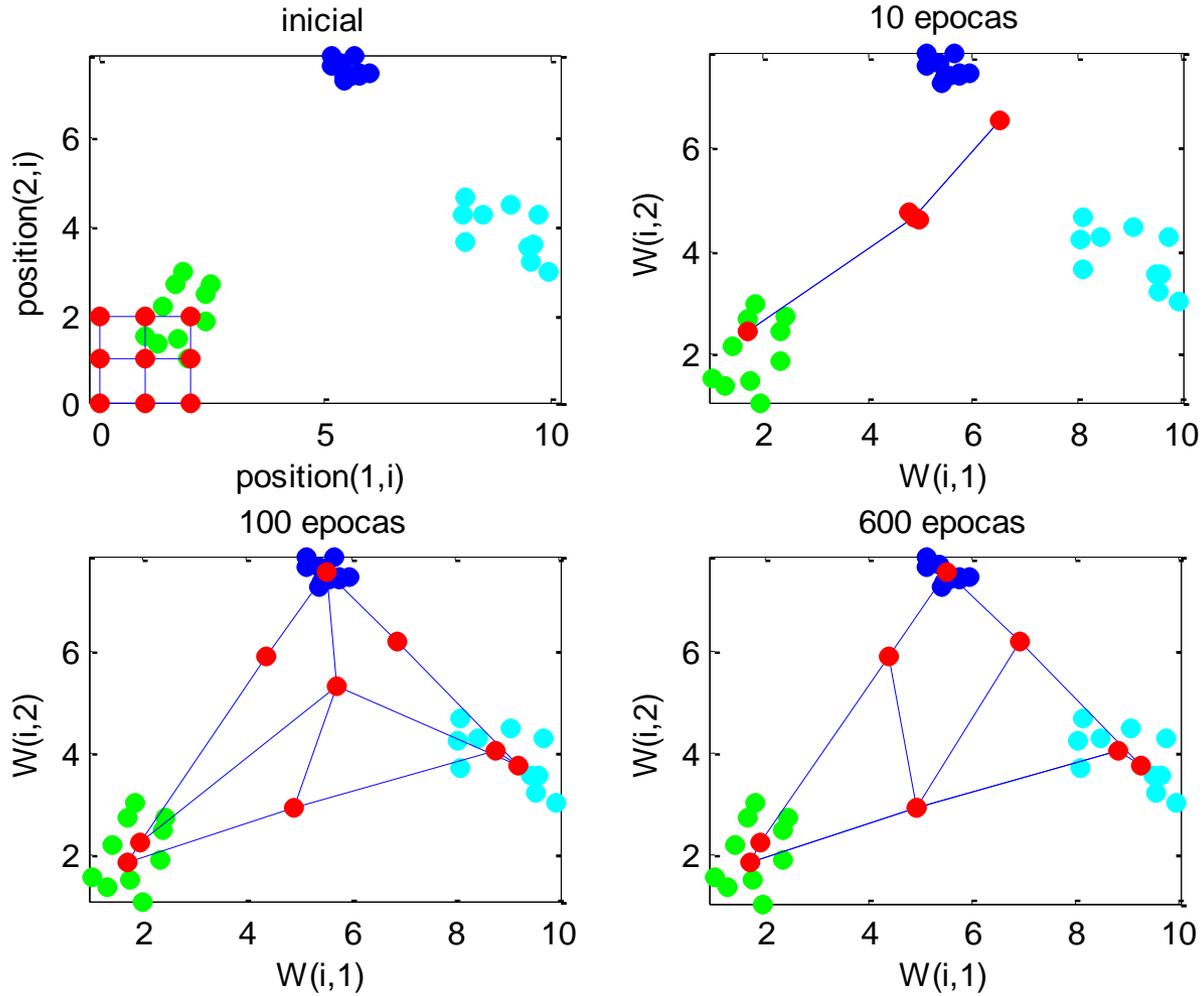


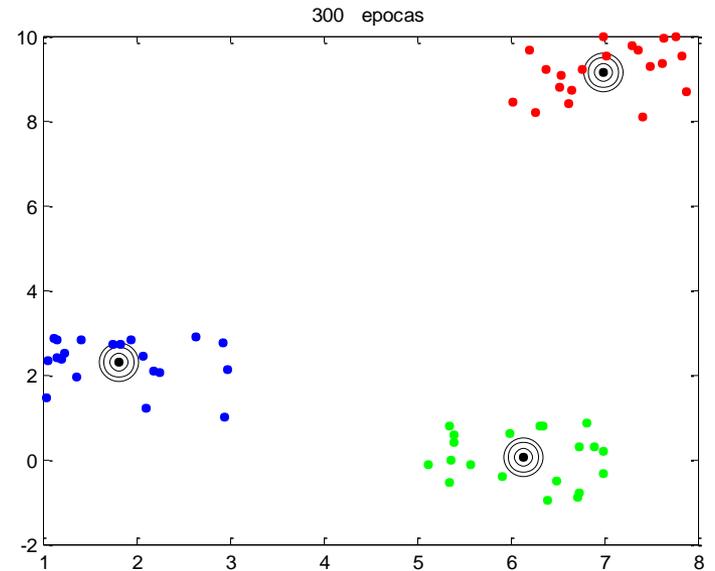
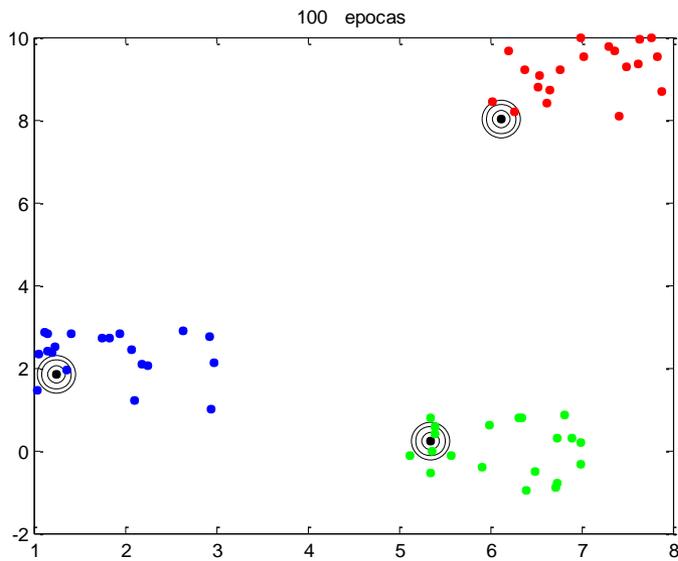
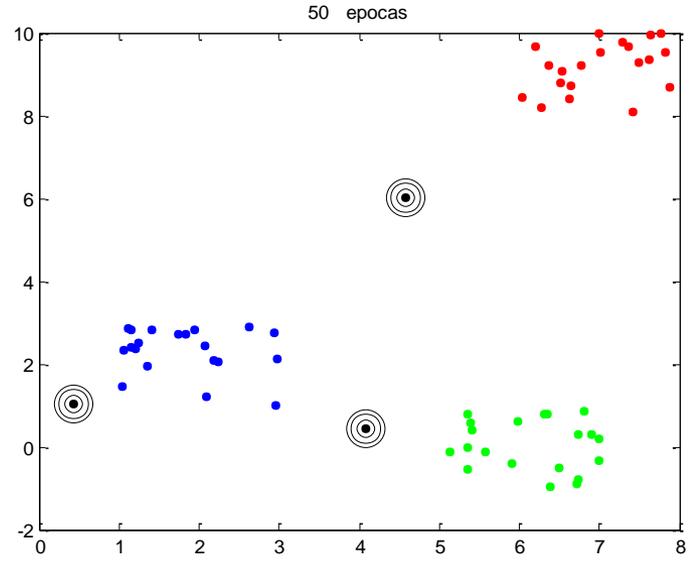
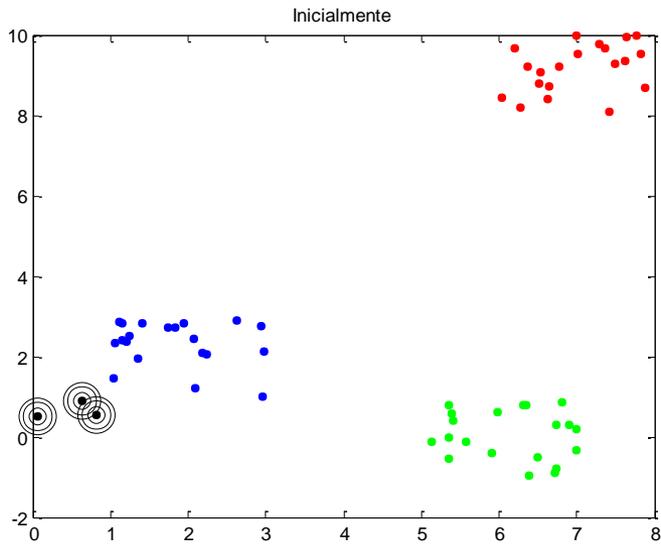




Casos de estudio:

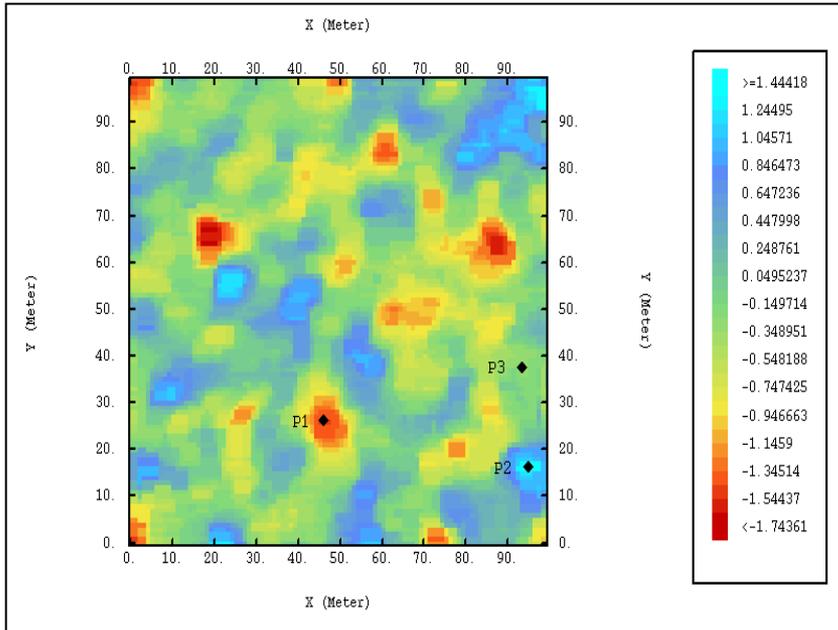
- **Distribución (TriangLat.m)**
- **Clasificación (SOMClass.m)**
- **PDVSA (cortesía Prof. O. Rondón)**
- **Mapas contextuales**
- **Señales ultrasónicas (cortesía Ing. C Correia)**



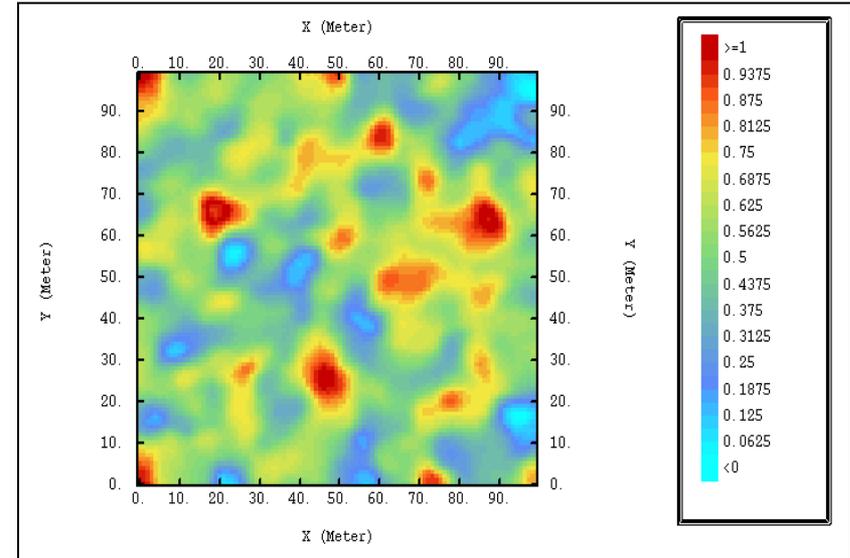




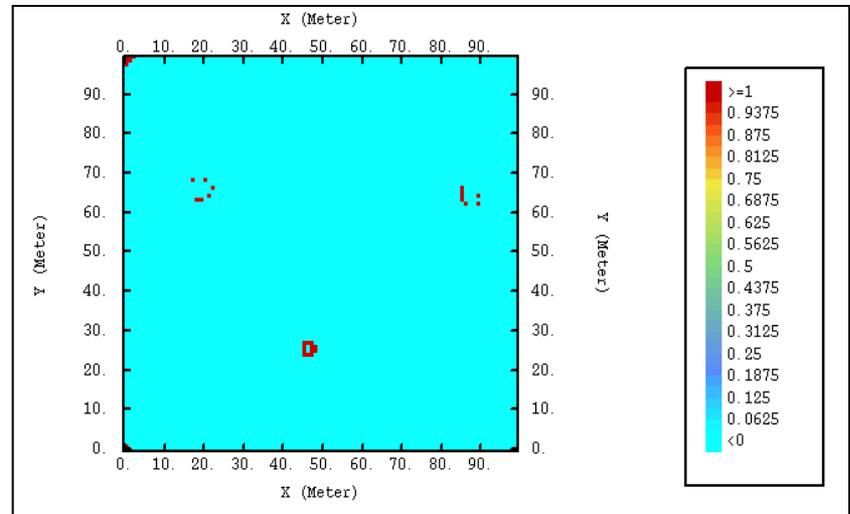
Se quiere encontrar zonas similares

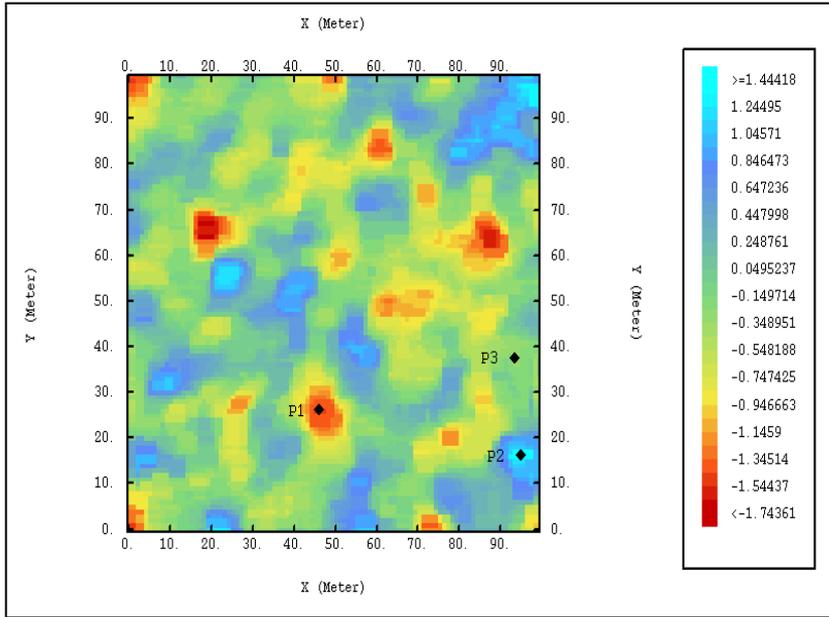


P1

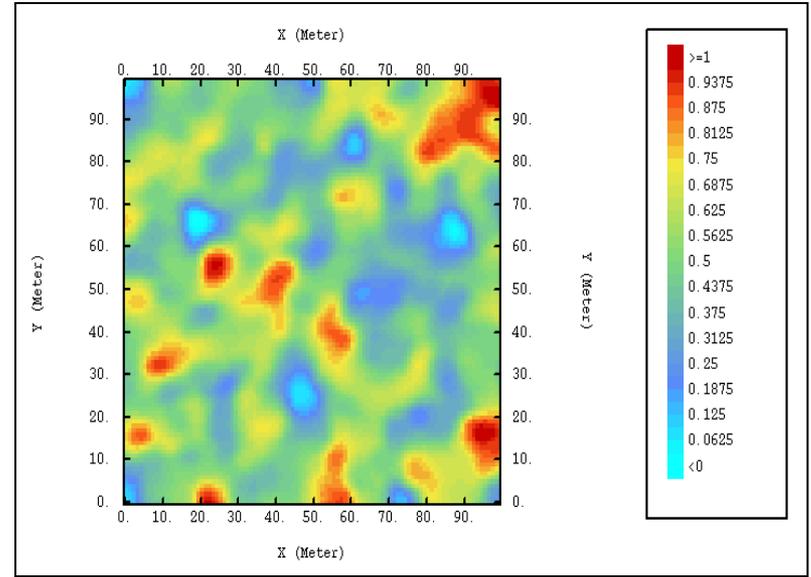


P1

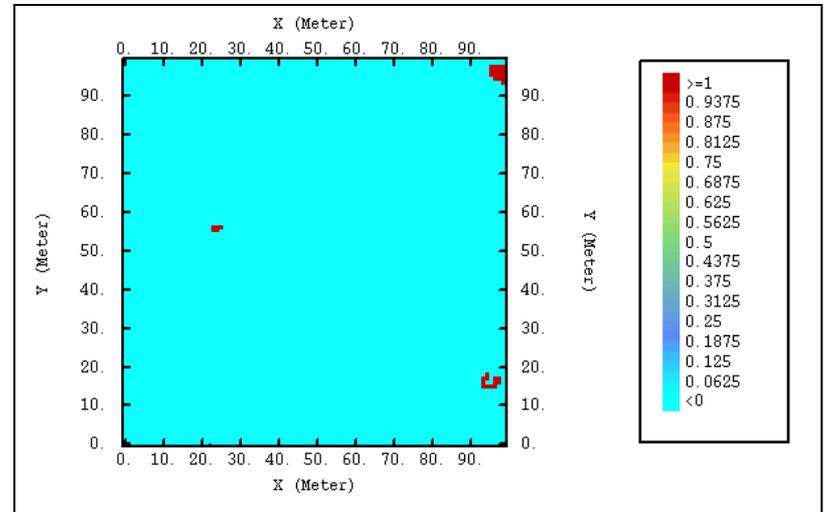


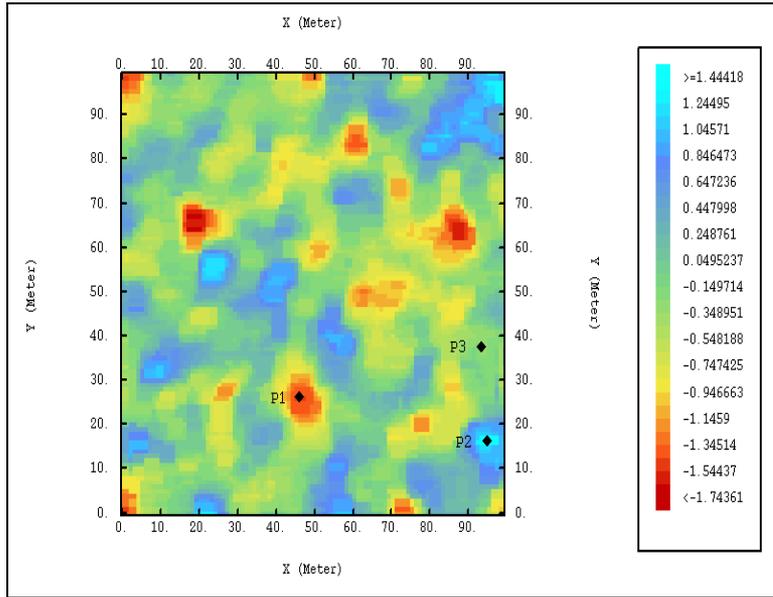


P2

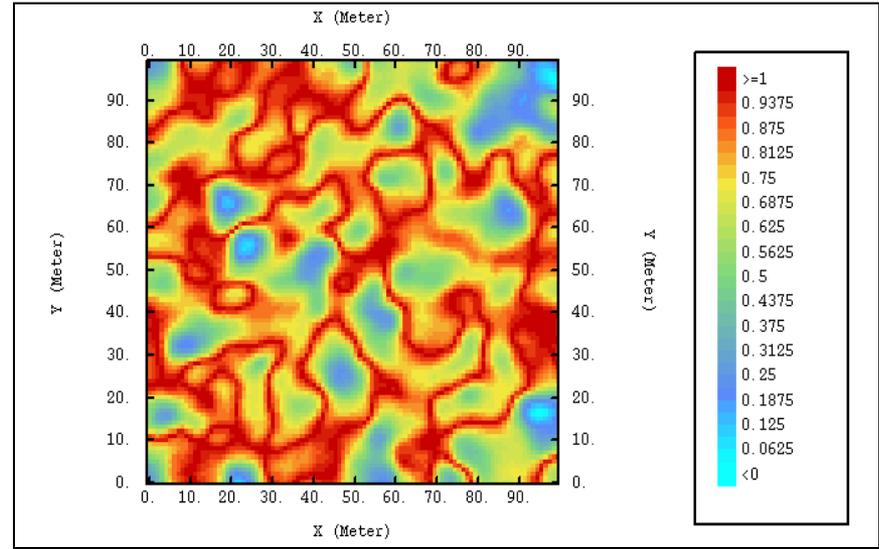


P2

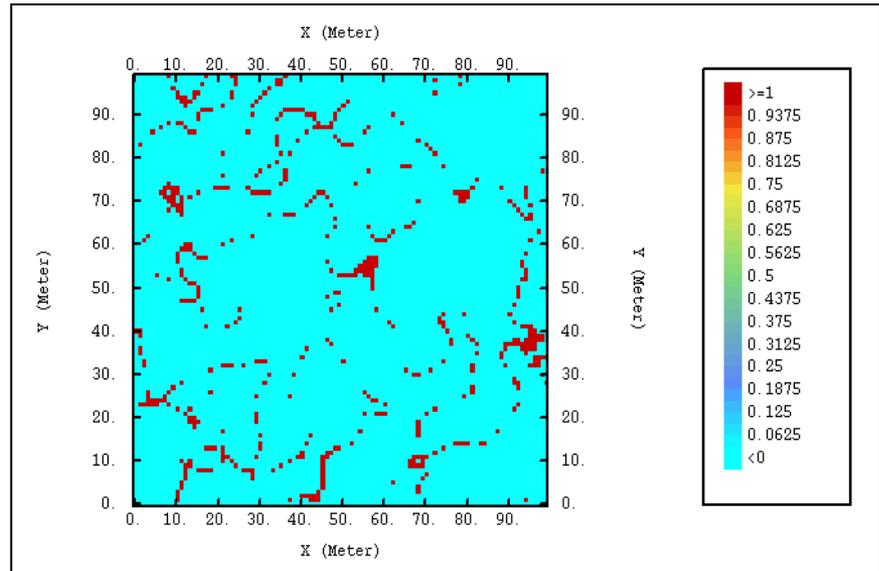


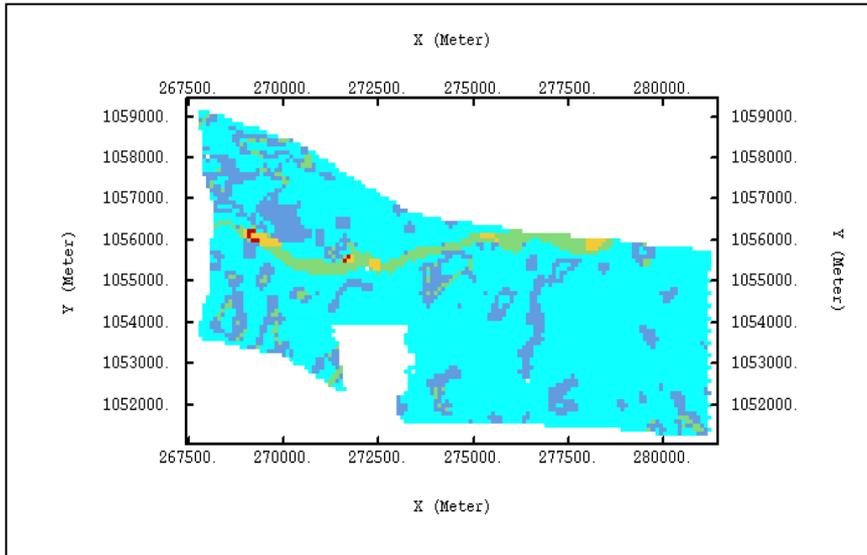
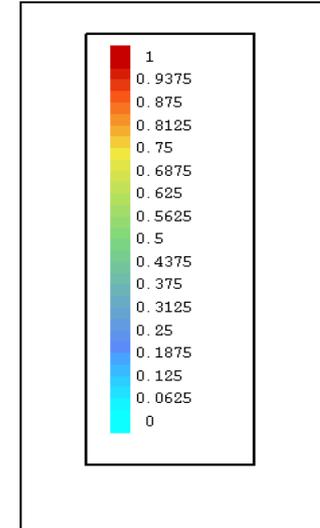
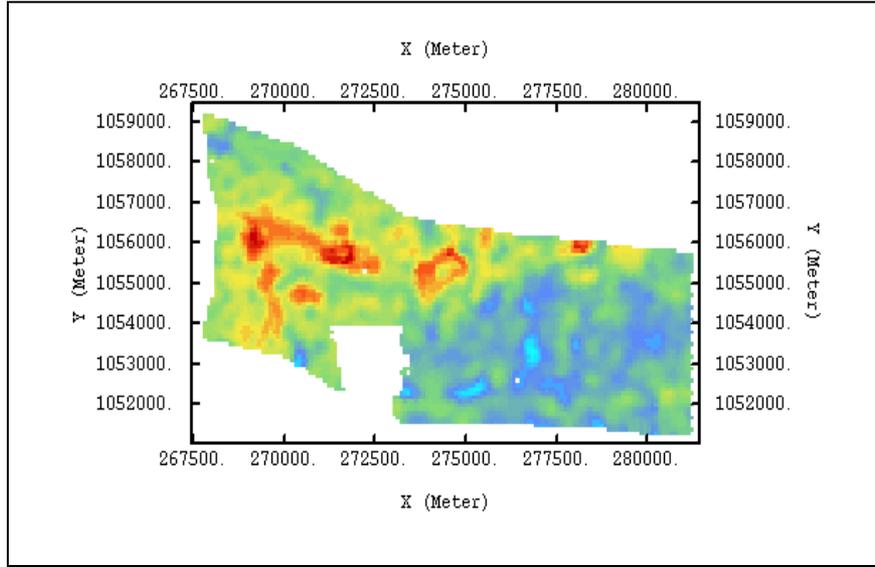


P3

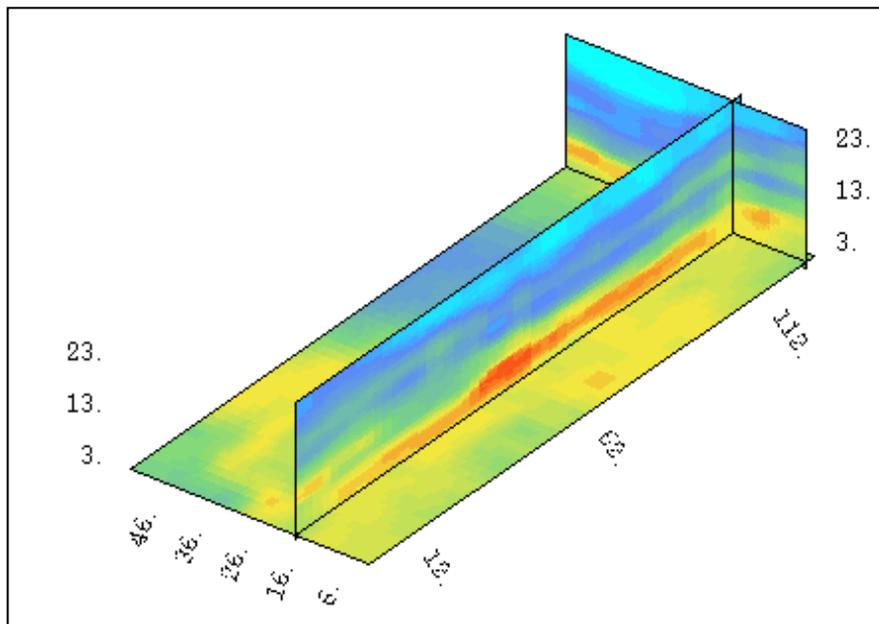


P3

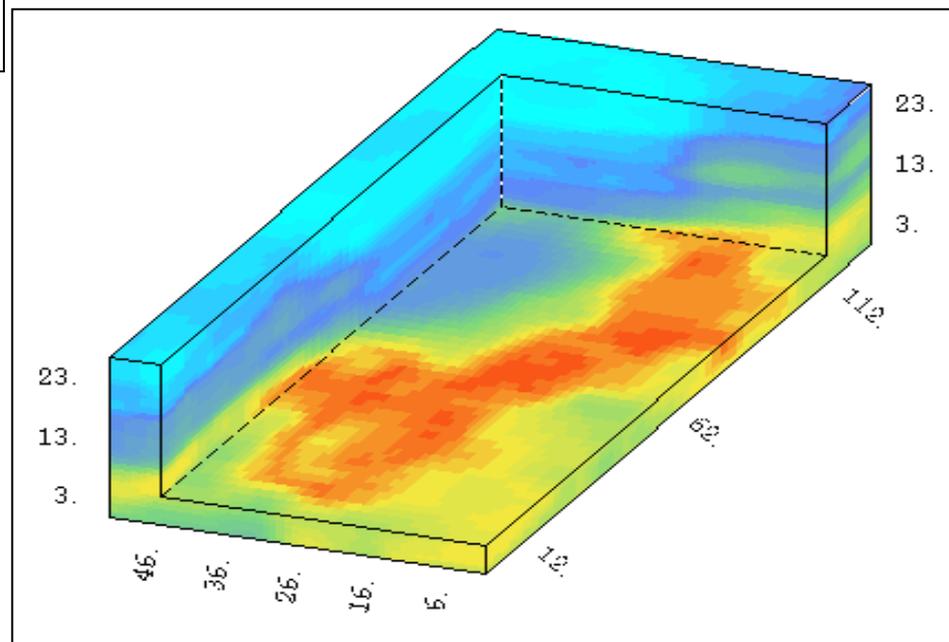




Aplicación con datos reales

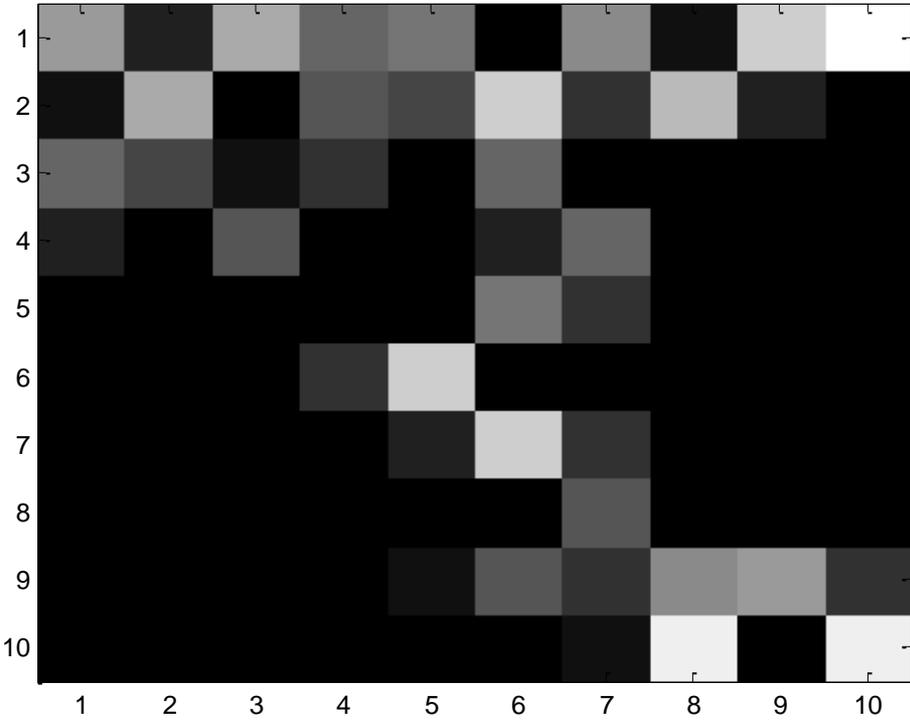


Aplicación con datos reales

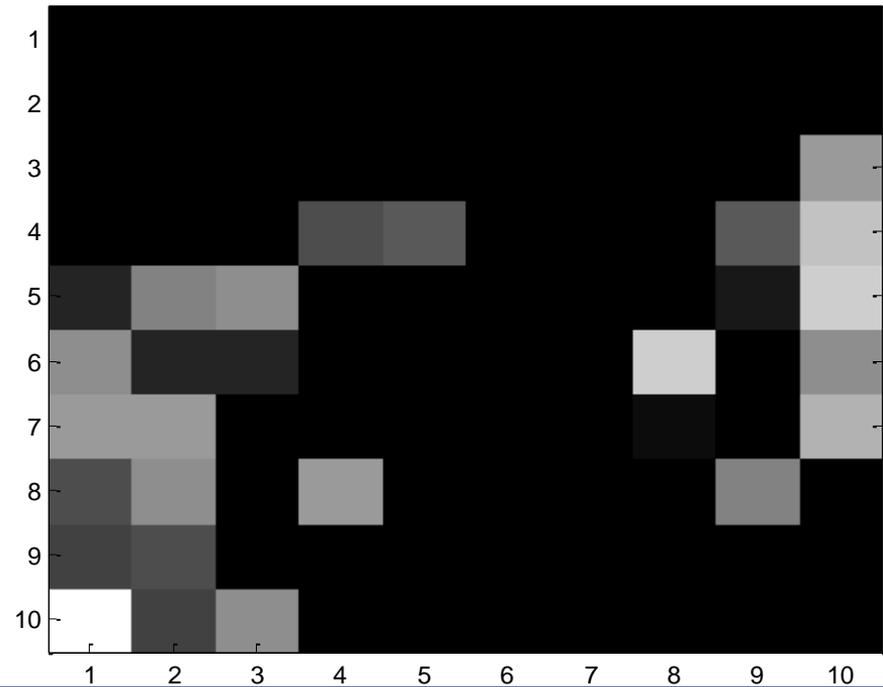


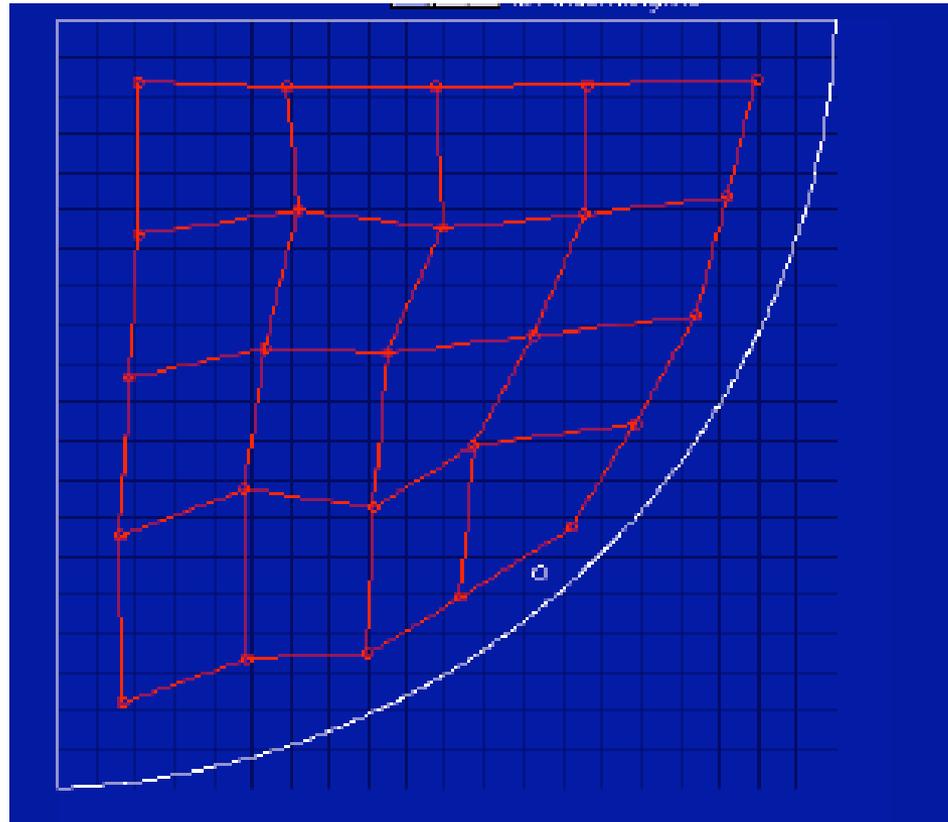


Grietas



Falta de Fusión





<http://www.patol.com/java/fill/index.html>

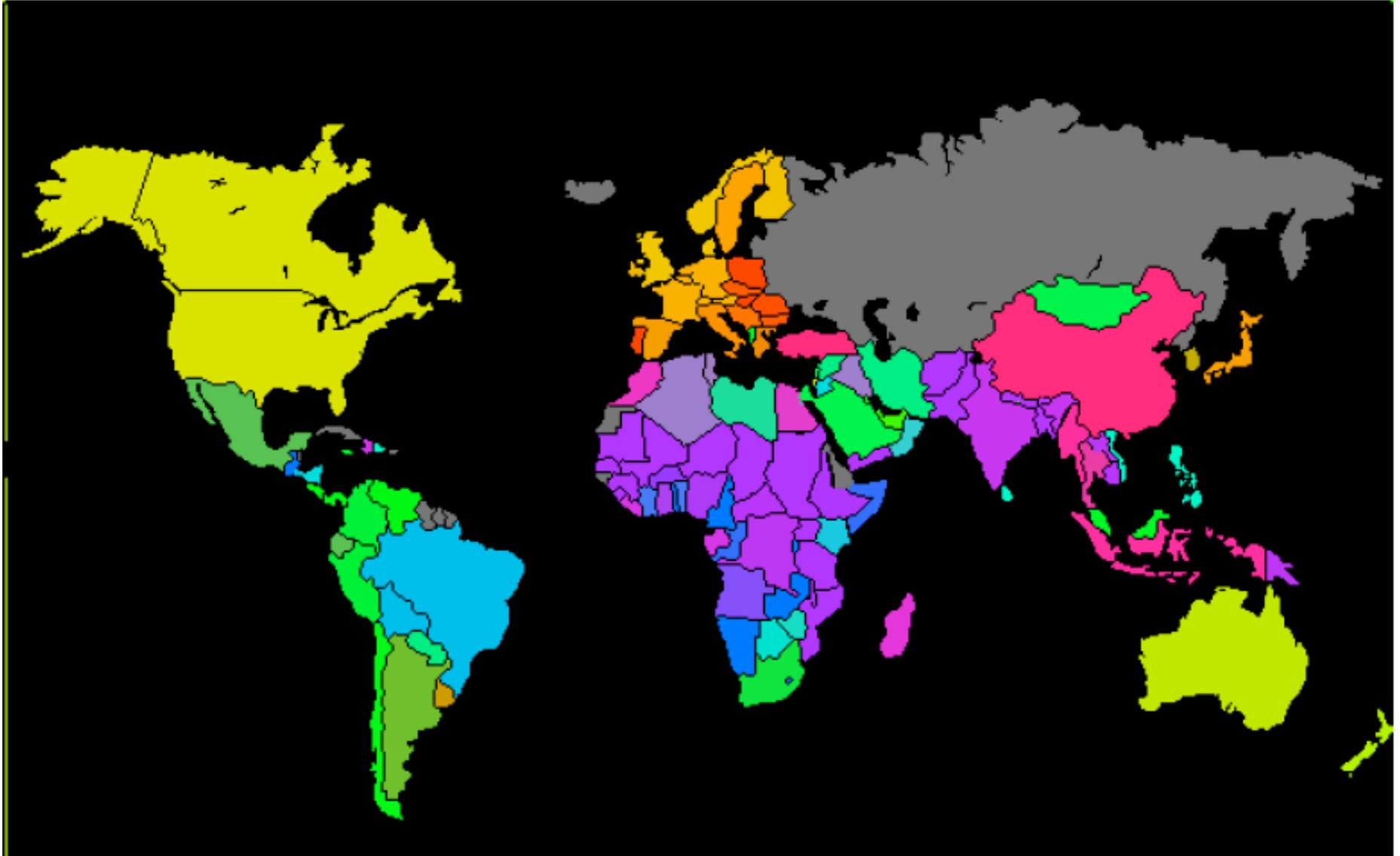


- Gas neural y Variantes:

<http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper/>

http://www.sund.de/netze/applets/gng/full/GNG_0.html

Sugerencia para un proyecto



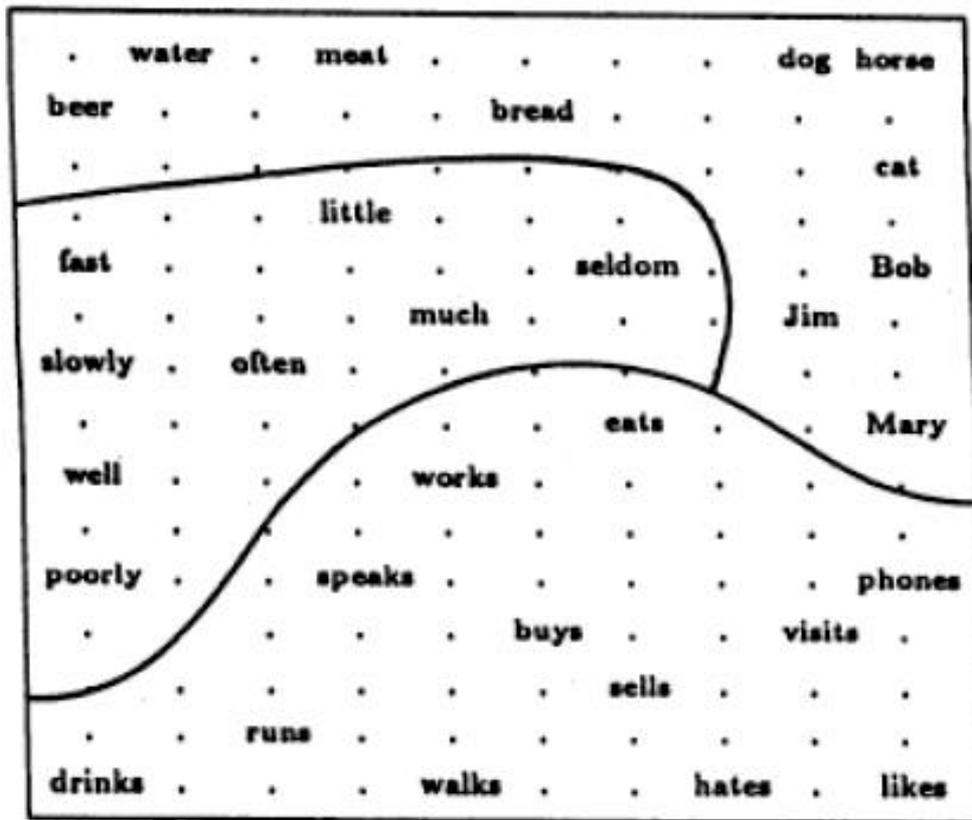


Fig. 7.11. "Semantic map" obtained on a network of 10×15 cells after 2000 presentations of word-context-pairs derived from 10,000 random sentences of the kind shown in Fig. 7.8. Nouns, verbs and adverbs are segregated into different domains. Within each domain a further grouping according to aspects of meaning is discernible