Mapas auto-organizativos

Diego Milone

Inteligencia Computacional
Departamento de Informática

FICH-UNL

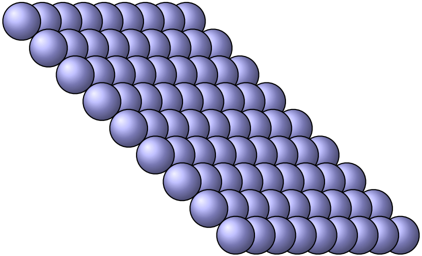
Auto-organización

- Autoorganización: es el proceso en el cual, por medio de interacciones locales, se obtiene ordenamiento global (Turing, 1952).

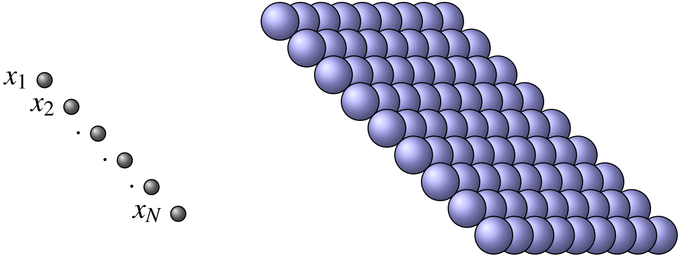
Auto-organización

- Autoorganización: es el proceso en el cual, por medio de interacciones locales, se obtiene ordenamiento global (Turing, 1952).
    - Ejemplo 1: evolución de la ubicación de los alumnos asisten a un curso...
    - Ejemplo 2: las celulas del cerebro se auto-organizan en grupos de acuerdo a la información que reciben...

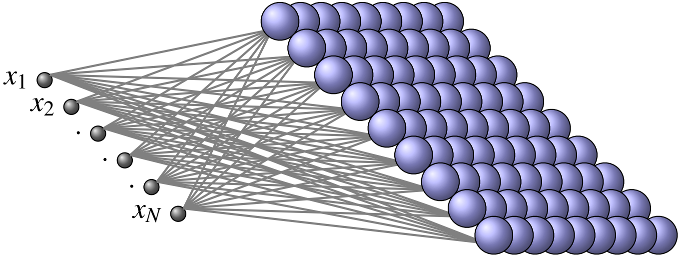
SOM: Arquitectura



SOM: Arquitectura



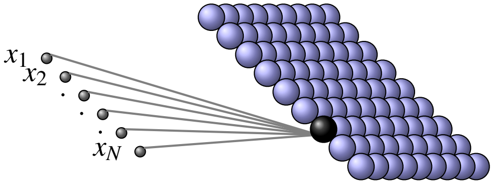
SOM: Arquitectura



SOM: Arquitectura

- Arquitectura SOM 2D

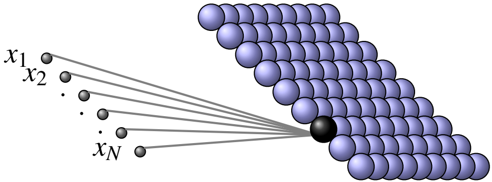
$$\mathbf{x} \longrightarrow \mathbf{w}_j \longrightarrow s_j$$



SOM: Arquitectura

- Arquitectura SOM 2D

$$\mathbf{x} \rightarrow \mathbf{w}_j \rightarrow s_j$$

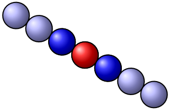


- Salidas: se activa sólo la ganadora j^*

$$j^*(n) = G(\mathbf{x}(n)) = \arg\min_j \{\|\mathbf{x}(n) - \mathbf{w}_j(n)\|\}$$

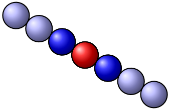
Entornos de influencia o vecindades

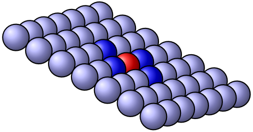
- Formas básicas:



Entornos de influencia o vecindades

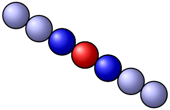
- Formas básicas:

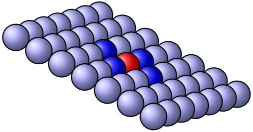




Entornos de influencia o vecindades

- Formas básicas:





- Alcance Λ_G:
  - Entornos fijos
  - Entornos variables
- Excitación asociadas al entorno:
  - Excitatorias puras
  - Inhibitorias puras
  - Funciones generales de excitación/inhibición

Funciones de excitación/inhibición lateral

- Uniforme (entorno simple):

$$\Lambda_G(n) = 2 \Rightarrow \begin{cases} h_{G,i} = \beta(n) & \text{si } |G - i| \leq 2 \\ h_{G,i} = 0 & \text{si } |G - i| > 2 \end{cases}$$

siendo $\beta(n)$ adaptable en función de las iteraciones n.

Funciones de excitación/inhibición lateral

- Uniforme (entorno simple):

$$\Lambda_G(n) = 2 \Rightarrow \begin{cases} h_{G,i} = \beta(n) & \text{si } |G-i| \leq 2 \\ h_{G,i} = 0 & \text{si } |G-i| > 2 \end{cases}$$

siendo $\beta(n)$ adaptable en función de las iteraciones n.

- Gaussiana:

$$h_{G,i} = \beta(n) e^{-\frac{|G-i|^2}{2\sigma^2(n)}}$$

Funciones de excitación/inhibición lateral

- Uniforme (entorno simple):

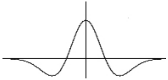
$$\Lambda_G(n) = 2 \Rightarrow \begin{cases} h_{G,i} = \beta(n) & \text{si } |G-i| \leq 2 \\ h_{G,i} = 0 & \text{si } |G-i| > 2 \end{cases}$$

siendo $\beta(n)$ adaptable en función de las iteraciones n.

- Gaussiana:

$$h_{G,i} = \beta(n) e^{-\frac{|G-i|^2}{2\sigma^2(n)}}$$

- Sombrero Mejicano: campos receptivos retina, inhibición lateral



Entrenamiento de un SOM

Diego Milone

Inteligencia Computacional
Departamento de Informática

FICH-UNL

Entrenamiento de un SOM

Características principales:

- Entrenamiento NO-supervisado

Entrenamiento de un SOM

Características principales:

- Entrenamiento NO-supervisado
- Aprendizaje competitivo

Algoritmo de entrenamiento

1. Inicialización:
    - pequeños valores aleatorios con $w_{ji} \in [-0{,}5, \cdots, +0{,}5]$ o también
    - eligiendo aleatoriamente $\mathbf{w}_j(0) = \mathbf{x}_\ell$ ($\ell \in [1, \cdots, L]$)

Algoritmo de entrenamiento

1. Inicialización:
   - pequeños valores aleatorios con $w_{ji} \in [-0{,}5, \cdots, +0{,}5]$ o también
   - eligiendo aleatoriamente $\mathbf{w}_j(0) = \mathbf{x}_\ell$ ($\ell \in [1, \cdots, L]$)
2. Selección del ganador:

$$G(\mathbf{x}(n)) = \arg\min_{\forall j} \{\|\mathbf{x}(n) - \mathbf{w}_j(n)\|\}$$

Algoritmo de entrenamiento

1. Inicialización:
   - pequeños valores aleatorios con $w_{ji} \in [-0{,}5, \cdots, +0{,}5]$ o también
   - eligiendo aleatoriamente $\mathbf{w}_j(0) = \mathbf{x}_\ell$ ($\ell \in [1, \cdots, L]$)
2. Selección del ganador:

$$G(\mathbf{x}(n)) = \arg\min_{\forall j} \{\|\mathbf{x}(n) - \mathbf{w}_j(n)\|\}$$

3. Adaptación de los pesos:

$$\mathbf{w}_j(n+1) = \begin{cases} \mathbf{w}_j(n) + \eta(n)\left(\mathbf{x}(n) - \mathbf{w}_j(n)\right) & \text{si } y_j \in \Lambda_G(n) \\ \mathbf{w}_j(n) & \text{si } y_j \notin \Lambda_G(n) \end{cases}$$

Algoritmo de entrenamiento

1. Inicialización:
   - pequeños valores aleatorios con $w_{ji} \in [-0{,}5, \cdots, +0{,}5]$ o también
   - eligiendo aleatoriamente $\mathbf{w}_j(0) = \mathbf{x}_\ell$ ($\ell \in [1, \cdots, L]$)
2. Selección del ganador:

$$G(\mathbf{x}(n)) = \arg\min_{\forall j} \{\|\mathbf{x}(n) - \mathbf{w}_j(n)\|\}$$

3. Adaptación de los pesos:

$$\mathbf{w}_j(n+1) = \begin{cases} \mathbf{w}_j(n) + \eta(n)\left(\mathbf{x}(n) - \mathbf{w}_j(n)\right) & \text{si } y_j \in \Lambda_G(n) \\ \mathbf{w}_j(n) & \text{si } y_j \notin \Lambda_G(n) \end{cases}$$

4. Volver a 2 hasta no observar cambios significativos.

Entrenamiento de un SOM

Consideraciones prácticas:

- $\Lambda_G(n)$ es generalmente cuadrado

Entrenamiento de un SOM

Consideraciones prácticas:

- $\Lambda_G(n)$ es generalmente cuadrado
- $h_{G,i}(n)$ uniforme en i, decreciente con n

Entrenamiento de un SOM

Consideraciones prácticas:

- $\Lambda_G(n)$ es generalmente cuadrado
- $h_{G,i}(n)$ uniforme en i, decreciente con n
- $0 < \eta(n) < 1$ decreciente con n

Entrenamiento de un SOM

Consideraciones prácticas:

- $\Lambda_G(n)$ es generalmente cuadrado
- $h_{G,i}(n)$ uniforme en i, decreciente con n
- $0 < \eta(n) < 1$ decreciente con n

¿Cómo varían $\Lambda_G(n)$ y $\eta(n)$?

Etapas del entrenamiento

1. Ordenamiento global (o topológico)
   - $\Lambda_G(n)$ grande ($\approx$ medio mapa)
   - $\eta(n)$ grande (entre 0.9 y 0.7)
   - Duración 500 a 1000 épocas

Etapas del entrenamiento

1. Ordenamiento global (o topológico)
   - $\Lambda_G(n)$ grande ($\approx$ medio mapa)
   - $\eta(n)$ grande (entre 0.9 y 0.7)
   - Duración 500 a 1000 épocas
2. Transición
   - $\Lambda_G(n)$ se reduce linealmente hasta 1
   - $\eta(n)$ se reduce lineal o exponencialmente hasta 0.1
   - Duración $\approx$1000 épocas

Etapas del entrenamiento

1. Ordenamiento global (o topológico)
   - $\Lambda_G(n)$ grande ($\approx$ medio mapa)
   - $\eta(n)$ grande (entre 0.9 y 0.7)
   - Duración 500 a 1000 épocas
2. Transición
   - $\Lambda_G(n)$ se reduce linealmente hasta 1
   - $\eta(n)$ se reduce lineal o exponencialmente hasta 0.1
   - Duración $\approx$1000 épocas
3. Ajuste fino (o convergencia)
   - $\Lambda_G(n) = 0$ (sólo se actualiza la ganadora)
   - $\eta(n) = cte$ (entre 0.1 y 0.01)
   - Duración: hasta convergencia ($\approx$ 3000 épocas)

Formación de mapas topológicos

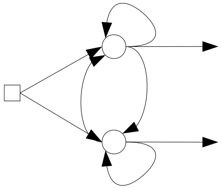
Diego Milone

Inteligencia Computacional
Departamento de Informática

FICH-UNL

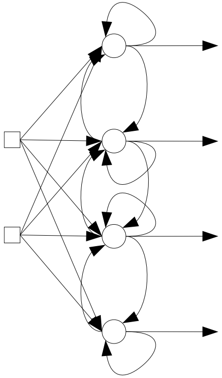
Formación de mapas topológicos

- Ejemplo 1: $\mathbb{R}^1 \rightarrow \mathbb{R}^1$, 2 neuronas



Formación de mapas topológicos

- Ejemplo 1: $\mathbb{R}^1 \to \mathbb{R}^1$, 2 neuronas
- Ejemplo 2: $\mathbb{R}^2 \to \mathbb{R}^1$, 4 neuronas



Formación de mapas topológicos

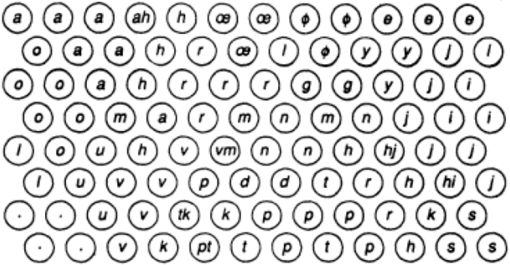
- Ejemplo 1: $\mathbb{R}^1 \rightarrow \mathbb{R}^1$, 2 neuronas
- Ejemplo 2: $\mathbb{R}^2 \rightarrow \mathbb{R}^1$, 4 neuronas
- Ejemplo 3: $\mathbb{R}^2 \rightarrow \mathbb{R}^2$, 4 neuronas

Formación de mapas topológicos

- Ejemplo 4: $\mathbb{R}^N \rightarrow \mathbb{R}^2$, M neuronas

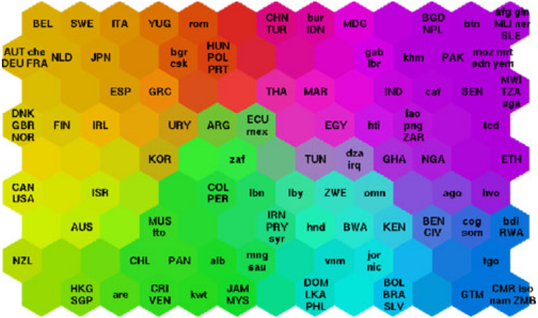
Formación de mapas topológicos

- Ejemplo 4: $\mathbb{R}^N \to \mathbb{R}^2$, M neuronas
- Ejemplo 5: el “phonetic typewriter” (Kohonen)



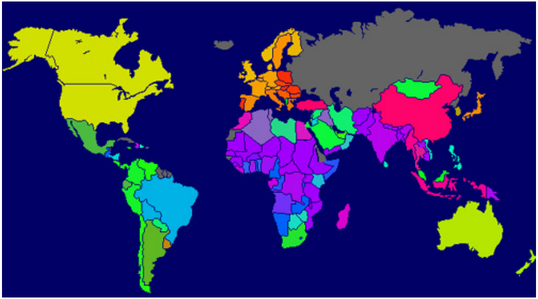
Formación de mapas topológicos

- Ejemplo 4: $\mathbb{R}^N \to \mathbb{R}^2$, M neuronas
- Ejemplo 5: el “phonetic typewriter” (Kohonen)
- Ejemplo 6: el experimento de los países (Kohonen)



Formación de mapas topológicos

- Ejemplo 4: $\mathbb{R}^N \to \mathbb{R}^2$, M neuronas
- Ejemplo 5: el “phonetic typewriter” (Kohonen)
- Ejemplo 6: el experimento de los países (Kohonen)



Formación de mapas topológicos

- Ejemplo 4: $\mathbb{R}^N \rightarrow \mathbb{R}^2$, M neuronas
- Ejemplo 5: el “phonetic typewriter” (Kohonen)
- Ejemplo 6: el experimento de los países (Kohonen)
- Ejemplo 7: ejemplos demo Matlab

Agrupamiento y clasificación

¿Cómo utilizar un SOM para clasificar patrones?

Agrupamiento y clasificación

¿Cómo utilizar un SOM para clasificar patrones?

- Entrenamiento no-supervisado

Agrupamiento y clasificación

¿Cómo utilizar un SOM para clasificar patrones?

- Entrenamiento no-supervisado
- Etiquetado de neuronas

Agrupamiento y clasificación

¿Cómo utilizar un SOM para clasificar patrones?

- Entrenamiento no-supervisado
- Etiquetado de neuronas
- Clasificación por mínima distancia

Demostraciones online...

- Caracteres: http://fbim.fh-regensburg.de/~saj39122/begrolu/kohonen.html
- 1D: http://www.cis.hut.fi/research/javasomdemo/demo1.html
- 2D: http://www.cis.hut.fi/research/javasomdemo/demo2.html
- Otro 2D: http://www.neuroinformatik.ruhr-uni-bochum.de/VDM/research/gsn/DemoGNG/GNG.html
- 3D: http://www.sund.de/netze/applets/som/som1/index.htm
- Buscador WEB: http://www.gnod.net/

Cuantización vectorial con aprendizaje

Diego Milone

Inteligencia Computacional
Departamento de Informática

FICH-UNL

Cuantización vectorial con aprendizaje (LVQ)

Conceptos básicos:

- Cuantizador escalar: señales cuantizadas

Cuantización vectorial con aprendizaje (LVQ)

Conceptos básicos:

- Cuantizador escalar: señales cuantizadas
- Cuantizador vectorial:
  - centroides o prototipos
  - diccionario o code-book
  - el proceso de cuantización: de vectores a números enteros

Cuantización vectorial con aprendizaje (LVQ)

Conceptos básicos:

- Cuantizador escalar: señales cuantizadas
- Cuantizador vectorial:
  - centroides o prototipos
  - diccionario o code-book
  - el proceso de cuantización: de vectores a números enteros
- Ideas de cómo entrenarlo:
  - algoritmo k-means etiquetado para clasificación
  - SOM etiquetado como clasificador...
  - algoritmo supervizado LVQ

Algoritmo LVQ1

1. Inicialización aleatoria

Algoritmo LVQ1

1. Inicialización aleatoria
2. Selección:

$$c(n) = \arg\min_i \{\|\mathbf{x}(n) - \mathbf{m}_i(n)\|\}$$

Algoritmo LVQ1

1. Inicialización aleatoria
2. Selección:

$$c(n) = \arg\min_i \{\|\mathbf{x}(n) - \mathbf{m}_i(n)\|\}$$

3. Adaptación:

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(c,d,n)\alpha\left[\mathbf{x}(n) - \mathbf{m}_c(n)\right]$$

$$s(c,d,n) = \begin{cases} +1 & \text{si } c(n) = d(n) \\ -1 & \text{si } c(n) \neq d(n) \end{cases}$$

Algoritmo LVQ1

1. Inicialización aleatoria
2. Selección:

$$c(n) = \arg\min_i \{\|\mathbf{x}(n) - \mathbf{m}_i(n)\|\}$$

3. Adaptación:

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(c,d,n)\alpha\left[\mathbf{x}(n) - \mathbf{m}_c(n)\right]$$

$$s(c,d,n) = \begin{cases} +1 & \text{si } c(n) = d(n) \\ -1 & \text{si } c(n) \neq d(n) \end{cases}$$

4. Volver a 2 hasta satisfacer error de clasificación

LVQ1: observaciones

- Interpretación gráfica
  - Caso de clasificación correcta
  - Caso de clasificación incorrecta

LVQ1: observaciones

- Interpretación gráfica
  - Caso de clasificación correcta
  - Caso de clasificación incorrecta
- No hay arquitectura neuronal

LVQ1: observaciones

- Interpretación gráfica
  - Caso de clasificación correcta
  - Caso de clasificación incorrecta
- No hay arquitectura neuronal
- Se puede ver al cuantizador como SOM lineal, sin entorno y supervisado

LVQ1: observaciones

- Interpretación gráfica
  - Caso de clasificación correcta
  - Caso de clasificación incorrecta
- No hay arquitectura neuronal
- Se puede ver al cuantizador como SOM lineal, sin entorno y supervisado
- Velocidad de aprendizaje
  - ¿Existe un α_c óptimo para cada centroide?
  - ¿Se debe considerar un $\alpha_c(n)$ óptimo para cada instante de tiempo?

LVQ1: velocidad de aprendizaje

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(n)\alpha(n)\left[\mathbf{x}(n) - \mathbf{m}_c(n)\right]$$

LVQ1: velocidad de aprendizaje

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(n)\alpha(n)\left[\mathbf{x}(n) - \mathbf{m}_c(n)\right]$$

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(n)\alpha(n)\mathbf{x}(n) - s(n)\alpha(n)\mathbf{m}_c(n)$$

LVQ1: velocidad de aprendizaje

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(n)\alpha(n)\left[\mathbf{x}(n) - \mathbf{m}_c(n)\right]$$

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(n)\alpha(n)\mathbf{x}(n) - s(n)\alpha(n)\mathbf{m}_c(n)$$

$$\mathbf{m}_c(n+1) = \left[1 - s(n)\alpha(n)\right]\mathbf{m}_c(n) + s(n)\alpha(n)\mathbf{x}(n)$$

LVQ1: velocidad de aprendizaje

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(n)\alpha(n)\left[\mathbf{x}(n) - \mathbf{m}_c(n)\right]$$

$$\mathbf{m}_c(n+1) = \mathbf{m}_c(n) + s(n)\alpha(n)\mathbf{x}(n) - s(n)\alpha(n)\mathbf{m}_c(n)$$

$$\mathbf{m}_c(n+1) = \left[1 - s(n)\alpha(n)\right]\mathbf{m}_c(n) + s(n)\alpha(n)\mathbf{x}(n)$$

$$\mathbf{m}_c(n+1) =$$

$$\begin{aligned} = \quad & [1 - s(n)\alpha(n)] \\ & \{\mathbf{m}_c(n-1) + s(n-1)\alpha(n-1)\left[\mathbf{x}(n-1) - \mathbf{m}_c(n-1)\right]\} \\ & + s(n)\alpha(n)\mathbf{x}(n) \end{aligned}$$

$\mathbf{x}(n-1)$ es afectado dos veces por α

LVQ1: velocidad de aprendizaje

Siendo $\alpha < 1$ la importancia relativa de los primeros patrones de entrenamiento siempre será menor que la de los últimos.

LVQ1: velocidad de aprendizaje

Siendo $\alpha < 1$ la importancia relativa de los primeros patrones de entrenamiento siempre será menor que la de los últimos.

Si queremos que α afecte por igual a todos los patrones deberemos hacerlo decrecer con el tiempo.

LVQ1: velocidad de aprendizaje

Siendo $\alpha < 1$ la importancia relativa de los primeros patrones de entrenamiento siempre será menor que la de los últimos.

Si queremos que α afecte por igual a todos los patrones deberemos hacerlo decrecer con el tiempo.

Se debe cumplir que:

$$\alpha_c(n) = [1 - s(n)\alpha_c(n)]\,\alpha_c(n-1)$$

LVQ1-O: velocidad de aprendizaje

Demostrar que:

$$\alpha_c(n) = \frac{\alpha_c(n-1)}{1 + s(n)\alpha_c(n-1)}$$

(no sobrepasar $\alpha > 1$)