

# Introducción a otros modelos de redes neuronales

(REDES COMPETITIVAS Y REDES AUTO-ORGANIZADAS)

~~ cognitio ~~

---

V. Robles-Bykbaev (vrobles@ups.edu.ec)

<http://catedraunescoinclusion.org/>

<http://giiata.blog.ups.edu.ec/>

**Universidad Politécnica Salesiana,  
Sede Matriz Cuenca, 2019**



**GIATA**

# Contenidos

- 1 Redes Competitivas
- 2 Ejemplo práctico de implementación de una RNA Hopfield

# Contenidos

1 Redes Competitivas

2 Ejemplo práctico de implementación de una RNA Hopfield

# Introducción

## Conceptos básicos

Como se mencionó anteriormente, las redes neuronales concurrentes son aquellas en las que las salidas una capa pueden emplearse para realimentar las entradas de dicha red [Nunes da Silva et al., 2017].

A continuación se describen algunos aspectos de interés relacionados con la historia y evolución de este tipo de redes [Demuth et al., 2014]:

- A finales de 1960 e inicios de 1970, Stephen Grossberg introdujo varios tipos de redes neuronales competitivas que usaban inhibición lateral<sup>1</sup> con un buen efecto.
- En 1973, Christoph von der Malsburg introdujo la regla de aprendizaje auto-organizado que permitió a la red clasificar entradas de modo que las neuronas vecinas respondían a entradas similares. La estructura de esta red buscaba imitar de cierto modo a las estructuras previamente descubiertas en el córtex visual por David Hubel y Torsten Wiesel.

# Introducción (cont'd)

## Conceptos básicos

- La propuesta de von der Malsburg generó gran interés, pero dado que su algoritmo de aprendizaje usaba cálculos no-locales para asegurar que los pesos se normalicen, perdía plausibilidad.
- Grossberg extendió el trabajo de von der Malsburg al redescubrir la regla *instar* (introducida previamente por Nils Nilsson en 1965). Con ello, Grossberg demostró que esta regla permite evitar la re-normalización de pesos que necesitaba von der Malsburg. Por lo tanto, **los vectores de pesos que aprenden a reconocer vectores de entradas normalizadas automáticamente se normalizan a sí mismos.**

# Introducción (cont'd)

## Conceptos básicos

- Otro trabajo interesante de redes neuronales competitivas fue desarrollado por Teuvo Kohonen. Su propuesta se enfocó en aplicaciones de ingeniería y descripciones matemáticas eficientes de las RNA. En los años 70s Kohonen desarrolló una versión simplificada de la regla *instar* e inspirado en los desarrollos de von der Malsburg y Grossberg, encontró una forma eficiente de incorporar la topología a una red competitiva.

---

<sup>1</sup>Es la capacidad que tiene una neurona excitada para reducir la actividad de sus vecinas. Esto crea un contraste en la estimulación que permite incrementar la percepción sensorial

# Red Hopfield

## Conceptos básicos y estructura

Esta red neuronal fue desarrollada John Hopfield en 1982 y cuenta con retroalimentación global. Sus principales características son las que se detallan a continuación [Nunes da Silva et al., 2017]:

- Comportamiento dinámico.
- Habilidad para memorizar relaciones.
- Posibilidad de almacenar información.
- Facilidad de implementación en *hardware* analógico.

# Red Hopfield (cont'd)

## Conceptos básicos y estructura

El principal aporte de Hopfield fue **demostrar que las redes recurrentes con una sola capa pueden ser caracterizadas por una función de energía**, que está relacionada con los estados dinámicos de su comportamiento dinámico [Nunes da Silva et al., 2017].

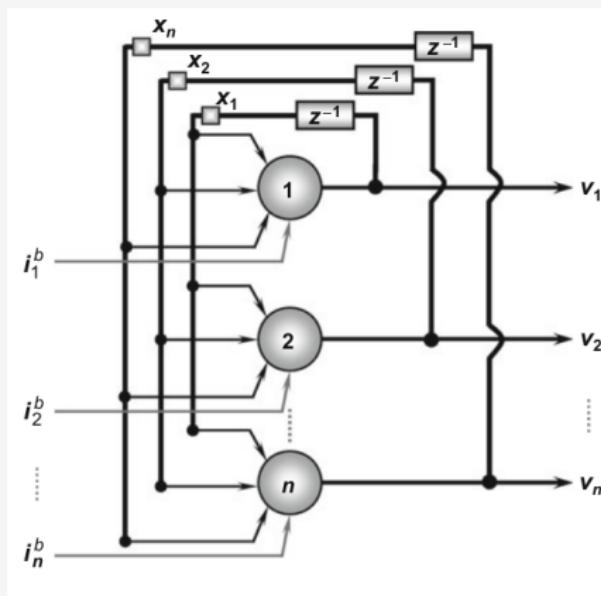
La red está constituida por una sola capa, donde todas las neuronas están conectadas a todas las demás y a sí mismas. Todas las salidas de la red son realimentadas a la todas las entradas.

En la Figura 1 podemos observar la arquitectura de la red Hopfield.

# Red Hopfield (cont'd)

## Conceptos básicos y estructura

Figure: Estructura de una RNA Hopfield [Nunes da Silva et al., 2017].



# Red Hopfield (cont'd)

## Conceptos básicos y estructura

La red Hopfield puede funcionar de diversos modos, y en nuestro caso presentaremos una breve descripción de su modo de operación como **memoria asociativa**. Para ello, de acuerdo a una cierta cantidad  $p$  de patrones  $\{z\}$  a ser almacenados en la memoria, donde cada uno tiene  $n$  elementos, la red se define como:

$$\mathbf{W} = \frac{1}{n} \sum_{k=1}^p z^{(k)} \cdot \left(z^{(k)}\right)^T \quad (1)$$
$$\mathbf{i}^b = \mathbf{0},$$

Donde:

- $\mathbf{i}^b$  son los bías de la RNA

# Red Hopfield (cont'd)

## Conceptos básicos y estructura

- La función de activación empleada es el escalón bipolar:

$$f(n) = \begin{cases} 1, & \text{si } n \geq 0 \\ -1, & \text{si } n < 0 \end{cases}$$

En el caso de las memorias asociativas, la diagonal de la matriz de pesos debe tener valores nulos (que indican la ausencia de auto-realimentación). Con ello, la expresión quedaría como:

$$\mathbf{W} = \frac{1}{n} \sum_{k=1}^p \underbrace{\mathbf{z}^{(k)} \cdot \left(\mathbf{z}^{(k)}\right)^T}_{(i)} - \underbrace{\frac{p}{n} \cdot \mathbf{I}}_{(ii)}$$

# Red Hopfield (cont'd)

## Conceptos básicos y estructura

Donde:

- $\mathbf{I} \in \mathbb{R}^{n \times n}$  y es la matriz identidad.
- La parcela  $(i)$  obtiene el producto de Kronecker.
- La parcela  $(ii)$  simplemente neutraliza los elementos en la diagonal principal ( $W_{kk} = 0$ ).

# Red Hopfield (cont'd)

## Conceptos básicos y estructura

### Producto de Kronecker (1)

- En esta sección se repasa de forma breve cómo se efectúa el producto de Kronecker.
- En primer lugar establecemos dos matrices que se multiplicarán, en nuestro caso hay que recordar que en la red Hopfield se multiplica la matriz de patrones  $x$  por sí misma.
- El producto de Kronecker establece lo siguiente [Vargas, 1996]:

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$$

# Red Hopfield (cont'd)

## Conceptos básicos y estructura

### Producto de Kronecker (1)

Supongamos que se desean multiplicar las siguientes matrices:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Aplicamos el producto de Kronecker:

$$A \otimes B = \begin{bmatrix} 1 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 2 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \\ 3 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 4 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \end{bmatrix}$$

# Red Hopfield (cont'd)

## Conceptos básicos y estructura

### Producto de Kronecker (2)

Con lo que el resultado sería el siguiente:

$$A \otimes B = \begin{bmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{bmatrix}$$

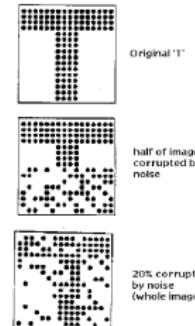
# Red Hopfield (cont'd)

## Conceptos básicos y estructura

### Regla de aprendizaje de Hebb para redes Hopfield (1)

La red Hopfield es una red asociativa en el sentido de que puede usarse para almacenar patrones y recuperarlos cuando se le proporciona incluso versiones incompletas de esos patrones [Gurney, 1997] (ver Figura 2).

Figure: Ejemplo de patrones que puede reconocer una red Hopfield [Gurney, 1997].



# Regla de aprendizaje de Hebb

## Redes Hopfield

Dado un conjunto de patrones  $z_k = (z_{k,1}, \dots, z_{k,n})^T$ ,  $k = 1 \dots p$ , la regla de aprendizaje de Hebb para la red Hopfield se define como sigue [Gurney, 1997]:

$$\mathbf{W} = \sum_{k=1}^m z_k z_k^T - p \cdot \mathbf{I} \quad (2)$$

# Regla de aprendizaje de Hebb (cont'd)

## Redes Hopfield

### Ejemplo de Cálculo (1)

Suponga que desea implementar una red Hopfield con un sólo patrón sencillo de 4 elementos (los cuadros en azul se consideran 1)

[Shevchuk, 2017]:

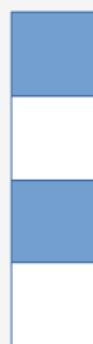
$$z_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

# Regla de aprendizaje de Hebb (cont'd)

## Redes Hopfield

Gráficamente, el patrón se vería así:

Figure: Ejemplo de patrones a entrenar.



# Regla de aprendizaje de Hebb (cont'd)

## Redes Hopfield

### Ejemplo de Cálculo (2)

Para realizar el cálculo nos ayudaremos de **Python** y la librería **NumPy**. El valor de  $p$  es 1, dado que tenemos 1 solo patrón. Si tuviésemos más patrones, rerealizamos el mismo cálculo y sumamos el nuevo resultado al anterior:

Figure: Ejemplo de cálculo de los pesos de la red Hopfield.

```
import numpy as np

# Creamos el patron
z1=np.array([1,-1,1,-1])

# Calculamos el producto de Kronecker:
zz=np.kron(z1,z1.transpose())

# Reorganizamos el resultado como una matriz de 4x4:
zz=zz.reshape(4,4)

# Restamos de la matriz identidad
w=zz-np.identity(4)
```

# Regla de aprendizaje de Hebb (cont'd)

## Redes Hopfield

### Ejemplo de Cálculo (3)

Si ejecutamos el código anterior, obtendremos el siguiente resultado:

Figure: Resultado de cálculo de los pesos de la red Hopfield.

```
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>>
>>> # Creamos el patron
... z1=np.array([1,-1,1,-1])
>>>
>>> # Calculamos el producto de Kronecker:
... zz=np.kron(z1,z1.transpose())
>>>
>>> # Reorganizamos el resultado como una matriz de 4x4:
... zz=zz.reshape(4,4)
>>>
>>> # Restamos de la matriz identidad
... w=zz-np.identity(4)
>>> w
array([[ 0., -1.,  1., -1.],
       [-1.,  0., -1.,  1.],
       [ 1., -1.,  0., -1.],
       [-1.,  1., -1.,  0.]])
>>> ■
```

# Contenidos

- 1 Redes Competitivas
- 2 Ejemplo práctico de implementación de una RNA Hopfield

# Aplicación práctica de la red Hopfield

## Ejemplo en Python

### Ejemplo 1.

Para desarrollar el siguiente ejemplo previamente debemos instalar la librería **NeuPy** que posee diversos métodos para trabajar con redes neuronales y aprendizaje profundo:

```
sudo pip install neupy
```

De igual forma, es importante considerar que se debe contar con los siguientes prerequisitos instalados:

- Python
- NumPy
- Matplotlib

## Contexte :

### ■ Avertir Drupal

```
// Gestion du contexte ete2013
// Utilisation d'un nouveau template
if (%variables['ctpage'] == "ete2013") {
variables['template_files']=array('page-ete');
}
```

## Práctica en clase

Determine el valor de salida de la red neuronal presentada en el ejemplo anterior para el caso en el que los pesos tomen un valor aleatorio en el rango  $[0, 1]$ , los mismos valores de bías ( $\theta$ ) y se trabaje con la función sigmodal.

## Contexte :

### ■ Avertir Drupal

```
// Gestion du contexte ete2013
// Utilisation d'un nouveau template
if (%variables['ctpage'] == "ete2013") {
variables['template_files']=array('page-ete');
}
```

## Referencias

-  Demuth, H. B., Beale, M. H., De Jess, O., and Hagan, M. T. (2014).  
*Neural Network Design.*  
Martin Hagan, USA, 2nd edition.
-  Gurney, K. (1997).  
*An introduction to neural networks.*  
CRC press.
-  Nunes da Silva, I., Hernane Spatti, D., Andrade Flauzino, R., Bartocci Liboni, L. H., and dos Reis Alves, S. F. (2017).  
*Artificial Neural Networks : A Practical Course.*  
Springer International Publishing, 1 edition.
-  Shevchuk, Y. (2017).  
*Discrete Hopfield Network.*

## Referencias (cont'd)

-  Vargas, J. P. (1996).  
productos de kronecker.  
*Revista de Matemática: Teoría y Aplicaciones*, 3(1):45–60.