

# Redes de Neuronas Recurrentes

**Computación con Inspiración Biológica**

---

Grupo de Computación Evolutiva y Redes Neuronales  
Departamento de Informática  
Universidad Carlos III de Madrid

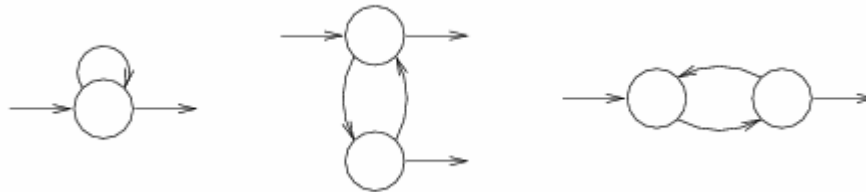
# Redes Recurrentes

- ❑ Introducción
  - ❑ Red de Hopfield
  - ❑ Redes parcialmente recurrentes
    - Red de Jordan
    - Red de Elman
  - ❑ Redes totalmente recurrentes
-

# Introducción

## ■ Características principales

- ❑ Pueden tener ciclos o bucles en las conexiones (conexiones recurrentes)
- ❑ Conexiones recurrentes:
  - de una neurona con ella misma
  - entre neuronas de una misma capa
  - entre neuronas de una capa a una capa anterior



# Introducción

## ■ Características principales (II)

- ❑ Al permitir conexiones recurrentes **aumenta** el número de pesos o de **parámetros ajustables** de la red
    - Aumenta la capacidad de representación
    - Se complica el aprendizaje
  - ❑ Las activaciones no dependen sólo de las activ. de la capa anterior sino también de la activación de cualquier otra neurona conectada a ella e incluso de su propia activación
-

# Introducción

- ❑ Es necesario incluir la **variable tiempo**

$$a_i(t + 1) = f_i\left(\sum_j w_{ji}a_j(t)\right)$$

- ❑ La variable tiempo hace que las redes tengan un **comportamiento dinámico o temporal**
  - ❑ Dos formas de entender el modo de actuación y aprendizaje
    - Evolución de las activaciones de la red hasta alcanzar un punto estable
    - Evolución de las activaciones de la red en modo continuo
-

# Introducción

1. **Evolución hasta alcanzar un punto estable**
    - Evolucionar la red (activaciones de sus neuronas), desde un estado inicial hasta conseguir que las activaciones de todas las neuronas de la red no se modifiquen : **punto estable**
      - ❑ Estado inicial: viene dado por el patrón de entrada
      - ❑ Estado estable: representa al patrón de salida de la red
      - ❑ Destaca la **Red de Hopfield**
-

# Introducción

## 2. Evolución de las activaciones en modo continuo

- En cada instante disponemos de la salida de la red
  - La salida depende en cada instante  $t$  de las entradas en el instante anterior  $t-1$ 
    - Aprendizaje por épocas
    - Aprendizaje en tiempo real o continuo: se aplica la ley de aprendizaje en cada instante (para cada patrón)
  - Redes parcialmente recurrentes: sólo tienen ciertas conexiones recurrentes
  - Redes totalmente recurrentes: no tienen restricciones
  - Todas usan **algoritmos supervisados** para el ajuste de parámetros
-

# Introducción

- ❑ Su comportamiento dinámico facilita el tratamiento de **información temporal** o **patrones dinámicos**:
    - patrones que dependen del tiempo
    - el valor del patrón en un determinado instante depende de sus valores en instantes anteriores de tiempo
  - ❑ Las redes recurrentes son apropiadas para tratar información temporal
  - ❑ También pueden emplearse redes estáticas como MLP o RNBR
    - entrada de la red: secuencia temporal de valores en el pasado
-



# Introducción

## Aplicaciones

- ❑ problemas modelización neurobiológica
  - ❑ tareas lingüísticas
  - ❑ reconocimiento del palabras y fonemas
  - ❑ control de procesos dinámicos
  - ❑ ...
-

# Red de Hopfield



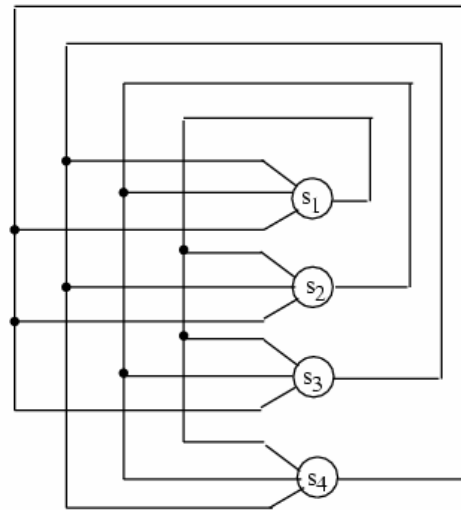
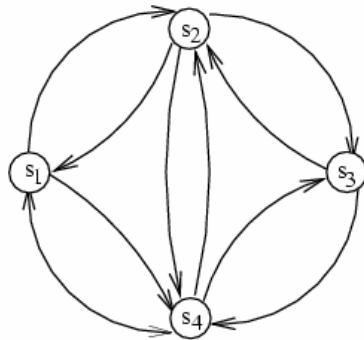
John Hopfield, 1933-

- John Hopfield la propone en 1982  
[Hopfield, 1982]
  - Modelo de memoria asociativa
    - Es capaz de recuperar patrones almacenados a partir de información incompleta e incluso a partir de patrones con ruido
    - Actúa como memoria asociativa procesando patrones estáticos (sin variable tiempo)
    - Todas las neuronas están conectadas con todas las demás
-

# Red de Hopfield

- Cada neurona se conecta con todas las demás

Dos formas diferentes de ver una red de Hopfield con 4 neuronas:



# Red de Hopfield

- Matriz de pesos  $W=(w_{ij})$ , orden  $n \times n$ .

$w_{ij}$  : peso de la conexión de neurona  $i$  a neur  $j$

- Matriz simétrica  $w_{ij} = w_{ji}$

- Los elementos de la diagonal son nulos (no existen conexiones reflexivas)

- Las neuronas poseen dos estados -1 y 1

- Estado de la neurona  $i$  en  $t+1$ :

$$s_i(t+1) = \text{sgn}(v_i(t+1)) \text{ para } i = 1, 2, \dots, n$$

$$\text{sgn}(v_i(t+1)) = \begin{cases} +1 & \text{si } v_i(t+1) > 0 \\ -1 & \text{si } v_i(t+1) < 0 \end{cases}$$

# Red de Hopfield

- Donde  $v_i(t+1)$  es el estado de activación de la neurona  $i$  calculado así:

$$v_i(t+1) = \sum_{j=1}^n w_{ji} s_j(t) - u_i \text{ para } i = 1, 2, \dots, n$$

donde  $s_j(t)$  es el estado de la neurona  $j$  en el instante anterior  $t$  y  $u_i$  es un umbral fijo aplicado a la neurona  $i$ .

- Si el nivel  $v_i(t+1) = 0$ , se considera que el estado de  $i$  no cambia
  - No tiene sentido hablar de entradas o salidas sino del estado de la red en un instante  $t$
-

# Red de Hopfield

- Para una red con  $n$  neuronas, el estado en  $(t+1)$ :

$$s(t+1) = [s_1(t+1), s_2(t+1), \dots, s_n(t+1)]^t$$

- El estado  $s$  representa una palabra binaria con  $n$  bits de información
-

# Aprendizaje y actuación de una red de Hopfield

- Dos fases de operación

- **Fase de almacenamiento**

- Se determinan los valores que tendrán los pesos para almacenar un conjunto de patrones

- **Fase de recuperación**

- mecanismo para recuperar la información almacenada a partir de información incompleta

---

# Red de Hopfield

## Fase de almacenamiento

- Conjunto de patrones que se desea almacenar

$$\{x(k) = (x_1(k), x_2(k), \dots, x_n(k))\}_{k=1, \dots, p}$$

donde cada patrón  $x(k)$  es un vector  $n$ -dimensional de componentes binarias  $(-1, 1)$

- Aplicando la regla de Hebb para almacenar patrones, el peso  $w_{ij}$  será:

$$w_{ji} = \sum_{k=1}^p x_j(k)x_i(k) \quad \forall i \neq j$$

- si las componentes  $x_j(k)$ ,  $x_i(k)$  son iguales,  $w_{ij}$  se incrementa en 1, si son distintos, se decrementa en 1
-



# Red de Hopfield

## Fase de almacenamiento

- Ejemplo: en la red de 4 neuronas almacenar

$$\mathbf{x}(1)=[1, 1, -1, -1] \text{ y } \mathbf{x}(2)=[-1, 1, 1, -1]$$

Inicialmente los pesos son nulos. Al presentar el patrón  $\mathbf{x}(1)$  la matriz de pesos será:

$$W(1) = \begin{pmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{pmatrix}$$

Equivale a la operación matricial  $\mathbf{x}_1^t \cdot \mathbf{x}_1 - \mathbf{I}$

Al presentar el patrón  $\mathbf{x}(2)$  se obtendría una matriz de modificación de los pesos anteriores

$$W(2) = W(1) + \begin{pmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \end{pmatrix}$$

# Red de Hopfield

## Fase de recuperación

- Patrón de prueba  $x = (x_1, x_2, \dots, x_n)$  diferente de los patrones almacenados (versión incompleta o con ruido de algún patrón almacenado)
- La RH va a recuperar el patrón almacenado más parecido al patrón de prueba  $x$
- Procedimiento:
  - Se inicializan los estados de las  $n$  neuronas de la red utilizando dicho patrón  $x$ :  $s_i(0) = x_i$  para  $i = 1, 2, \dots, n$
  - Se calculan los estados de la red en los siguiente instantes de tiempo utilizando las ecuaciones, hasta conseguir un **punto estable**  
(punto en el que los estados de las neuronas permanecen invariantes en el tiempo)
  - El estado estable de la red representa el patrón recuperado

# Red de Hopfield

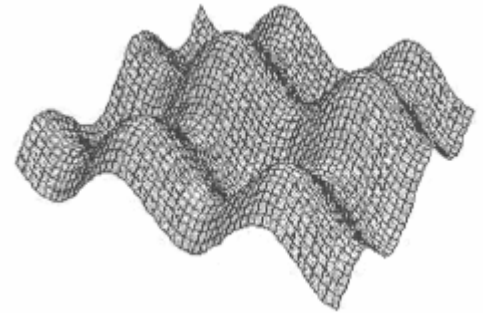
## Fase de recuperación

- Puede ocurrir que en la fase de recuperación la RH converja a estados estables que no correspondan con los patrones almacenados
  - Normalmente ésto ocurre por almacenar un excesivo número de patrones
-

# Red de Hopfield. Función de energía

- En MLP, RNBR, SOM, etc.. existe una **función de error o energía** que describe el funcionamiento de dichas redes
- En RH con  $n$  neuronas existe una función de energía:

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



- Los puntos estables de RH se corresponden con mínimos de la función de energía

# Red de Hopfield. Función de energía

- Cuando los estados de la red cambian siguiendo las ecuaciones,  $\Delta E$  es siempre negativo
  - En la fase de recuperación, cuando se modifican los estados de la red, se está aplicando el descenso del gradiente para encontrar un mínimo local de la función de energía
-

# Capacidad de una red de Hopfield

- Máximo número de patrones que se pueden almacenar con seguridad en una red de dimensión determinada.
- Problema complicado y aún no resuelto de forma satisfactoria.
- La mayor parte de los estudios realizados ofrecen estimaciones asintóticas del máximo valor tolerable de patrones cuando la dimensión de la red es muy grande.
- Si se desea que todos los prototipos sean recuperados con una probabilidad próxima a 1, se ha estimado la relación:

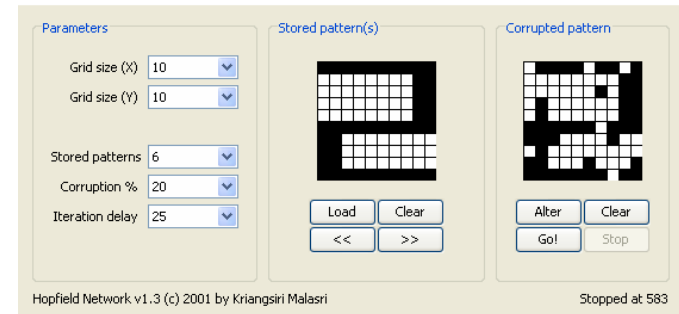
$$p \leq \frac{n}{4 \log n}$$

donde  $n$  es el número de neuronas y  $p$  el número de patrones que se desean recuperar.

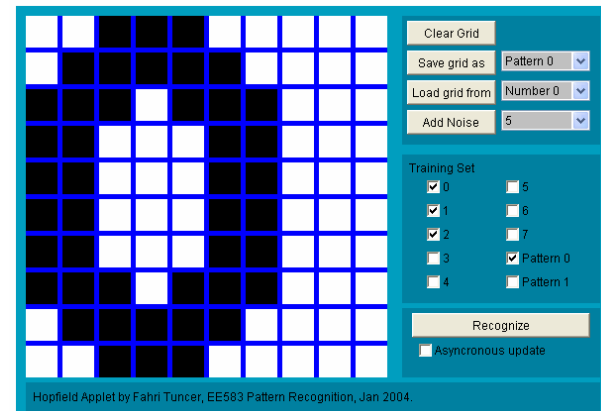
- Si se tolera un cierto margen de error en la recuperación de prototipos, puede admitirse un número de estos proporcional a la dimensión de la red, pudiendo llegar a ser del orden de  $0.138 n$ .
-

# Applets de RH

- <http://www.cbu.edu/~pong/ai/hopfield/hopfieldapplet.html>



- <http://www.eee.metu.edu.tr/~alatan/Courses/Demo/Hopfield.htm>



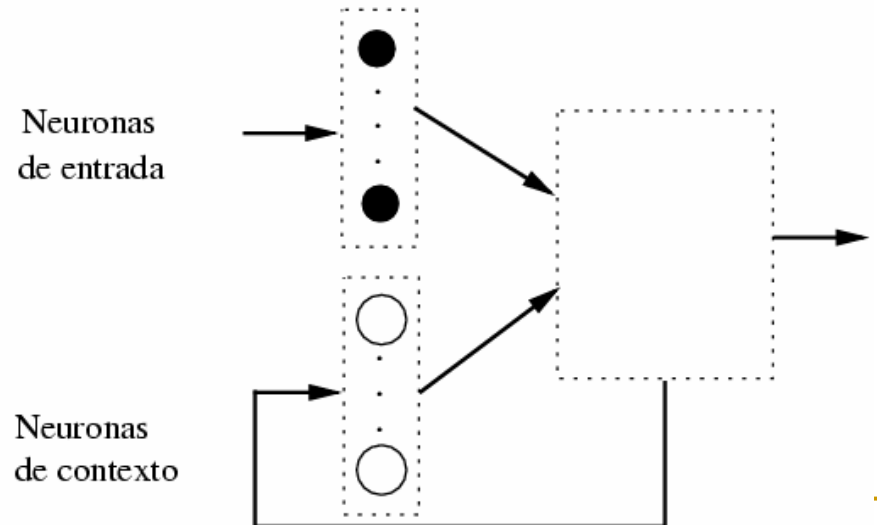
# Redes parcialmente recurrentes

- Son redes multicapa con algunas conexiones recurrentes
    - Estas conexiones permiten recordar el estado anterior de ciertas neuronas de la red
  - Generalmente existen ciertas neuronas especiales en la capa de entrada: **neuronas de contexto**, receptoras de las conexiones recurrentes
    - Funcionan como una memoria de la red, almacenando el estado de las neuronas de una cierta capa en el instante anterior
  - El resto de las neuronas de entrada actúan como receptores de los datos de entrada
-



# Redes parcialmente recurrentes

- Concatenando las activaciones de las neuronas de entrada y de contexto, puede verse como una red multicapa
- El cálculo de las activaciones se realiza como en una red multicapa sin recurrencias

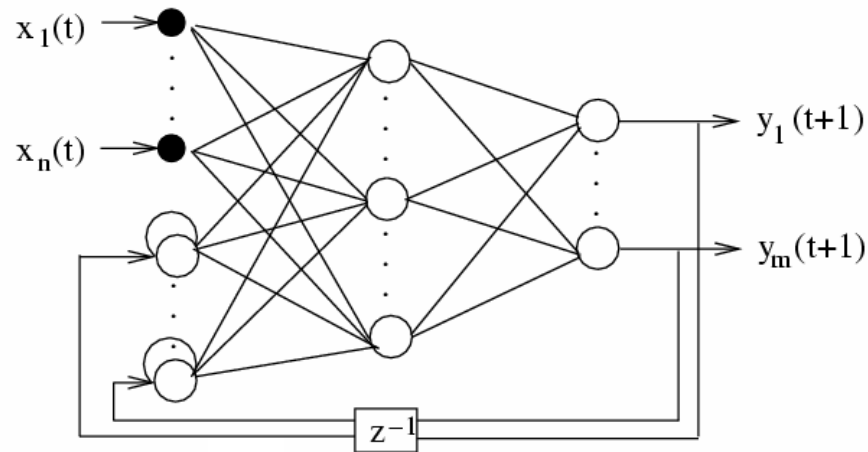


# Redes parcialmente recurrentes

- Los pesos asociados a las conex. recurrentes suelen ser constantes (no aprendizaje)
  - Por tanto, se puede usar el algoritmo de retropropagación, ya que los únicos pesos ajustables son no recurrentes
  - Las más conocidas
    - **Red de Jordan**
    - **Red de Elman**
-

# Red de Jordan

- Propuesto por Jordan en 1986 [Jordan, 1986a][Jordan, 1986b]
- Las neuronas de contexto reciben una copia de las neuronas de salida y de ellas mismas
- Las conexiones recurrentes tienen un parámetro asociado  $\mu$  (generalmente positivo y menor que 1)



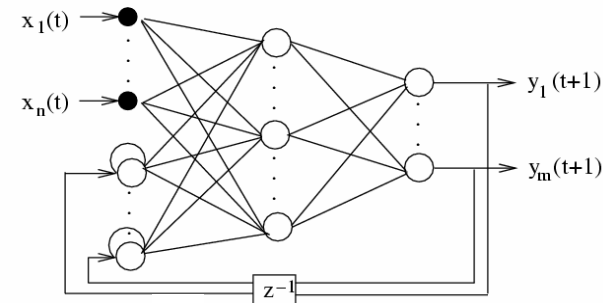
# Red de Jordan

- Cada neurona de contexto recibe una conexión de una neurona de salida y de ella misma
- La activación de las neuronas de contexto en  $t$ :

$$c_i(t) = \mu c_i(t-1) + y_i(t-1) \text{ para } i = 1, 2, \dots, m$$

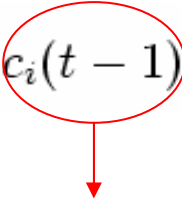
- La entrada total de la red es una concatenación de las activaciones de las neuronas de entrada y de las neuronas de contexto

$$u(t) = (x_1(t), \dots, x_n(t), c_1(t), \dots, c_m(t))$$



# Red de Jordan

- Neuronas ocultas y de salida: activación sigmoideal
- Neuronas de contexto: activación lineal
- Activación de las neuronas de contexto desarrollada en t:



$$c_i(t) = \mu c_i(t-1) + y_i(t-1) \text{ para } i = 1, 2, \dots, m$$


$$c_i(t-1) = \mu c_i(t-2) + y_i(t-2)$$

$$\begin{aligned} c_i(t) &= \mu^2 c_i(t-2) + \mu y_i(t-2) + y_i(t-1) = \dots = \\ &= \mu^{t-2} y_i(1) + \mu^{t-3} y_i(2) + \dots + \mu y_i(t-2) + y_i(t-1) \end{aligned}$$

---

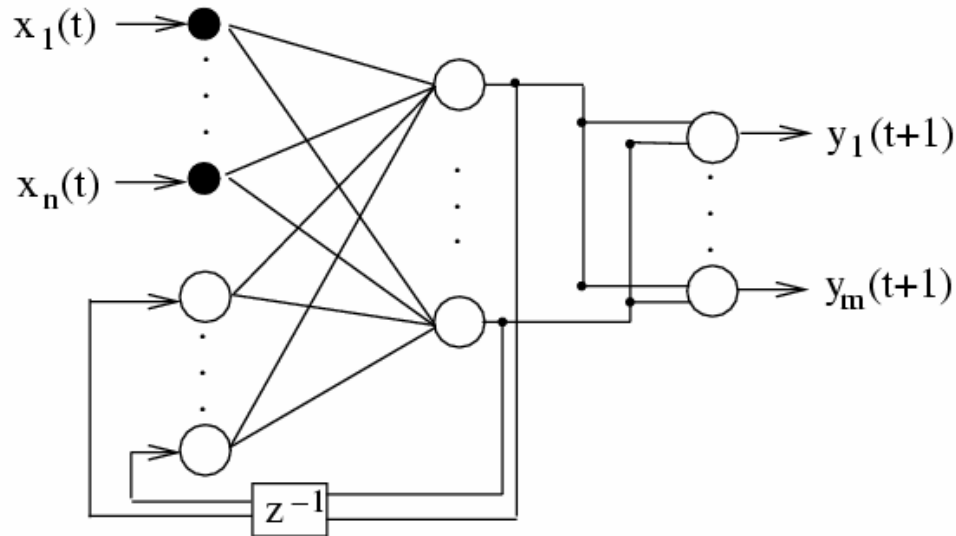
# Red de Jordan

- Las neuronas de contexto acumulan las salidas de la red para todos los instantes anteriores
  - El valor de  $\mu$  determina la sensibilidad de estas neuronas para retener dicha información
  - $\mu$  próximo a 0  estados alejados en el tiempo se olvidan con facilidad
  - $\mu$  próximo a 1  estados alejados en el tiempo se memorizan con facilidad
-

# Red de Elman



- Propuesta por Elman en 1990 [Elman, 1990]
- Las neuronas de contexto reciben una copia de las neuronas ocultas de la red



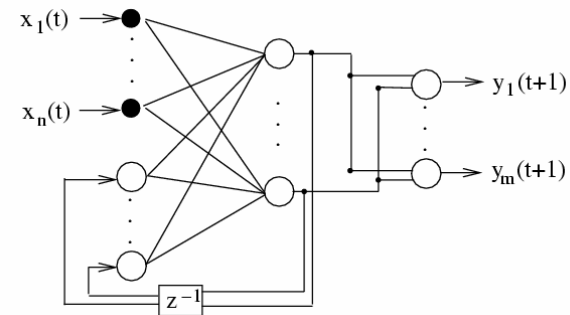
# Red de Elman

- Tantas neuronas de contexto como ocultas
- No existe un parámetro asociado a la conexión recurrente
- La activación de las neuronas de contexto:

$$c_i(t) = a_i(t - 1) \text{ para } i = 1, 2, \dots, r$$

- El resto de las activaciones se calculan como en una red feedforward considerando como entrada el vector total:

$$u(t) = (x_1(t), \dots, x_n(t), c_1(t), \dots, c_m(t))$$





# Red de Elman y de Jordan. Aprendizaje

1. Se inicializan las neuronas de contexto en  $t=0$
2. En el instante  $t$ , se presenta el patrón  $x(t) = (x_1(t), \dots, x_n(t))$  que junto a las activaciones de las neuronas de contexto  $c_i(t)$  forman el vector de entrada total de la red

$$u(t) = (x_1(t), \dots, x_n(t), c_1(t), \dots, c_m(t))$$

3. Se propaga el vector  $u(t)$  hacia la salida de la red, obteniéndose la salida  $y(t)$
  4. Se aplica la regla delta generalizada para modificar los pesos de la red
  5. Se incrementa la variable  $t$  en una unidad ( $t+1$ ) y se vuelve al paso 2
-

# Redes totalmente recurrentes

- No existe restricción de conectividad
- Sus neuronas reciben como entrada la activación del resto de neuronas y su propia activación

A: neuronas de entrada

$$a_i(t) = f_i \left( \sum_{j \in A \cup B} w_{ji} a_j(t-1) \right)$$

B: resto de neuronas

$w_{ji}$ : peso de  $j$  a  $i$

- Los pesos de las conexiones recurrentes se adaptan con aprendizaje
  - Aumenta el poder de representación
  - Se complica el aprendizaje

# Redes totalmente recurrentes

- El algoritmo de retropropagación estándar no puede aplicarse a estas redes
  - Existen dos algoritmos basados en retropropagación, adaptados para redes recurrentes:
    - Algoritmo de retropropagación a través del tiempo (Backpropagation through time)
    - Algoritmo de aprendizaje recurrente en tiempo real (Real time recurrent learning)
-

# Bibliografía

- **[Jordan, 1986a]** Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. of the Eighth Annual Conference of the Cognitive Science Society*, 531--546
  - **[Jordan, 1986b]** Jordan, M. (1986). Serial Order: a parallel distributed processing approach. Technical Report, Institute for Cognitive Science. University of California
  - **[Elman, 1990]** Elman J. (1990). Finding structure in time. *Cognitive Science*, 14: 179--211
  - **[Hopfield, 1982]** Hopfield, J. (1982). Neural Networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Science*, vol 81, p 3088—3092. National Academy of Sciences
-