Redes Neuronales Recurrentes Dinámicas.

Mª Carmen Pegalajar

mcarmen@decsai.ugr.es

Depto de Ciencias de la Computación e IA

Universidad de Granada

Redes Neuronales Recurrentes Dinámicas

1. Redes Neuronales Recurrentes Dinámicas (RNRD).
2. Identificación de Sistemas.
3. Reconocimiento de Voz mediante RNRSO.
4. Inferencia Gramatical (Crisp/Difusa) mediante RNRD.
5. Clasificación de ADN mediante RNRD.
6. Series Temporales y RNRD.

Redes Neuronales Recurrentes Dinámicas

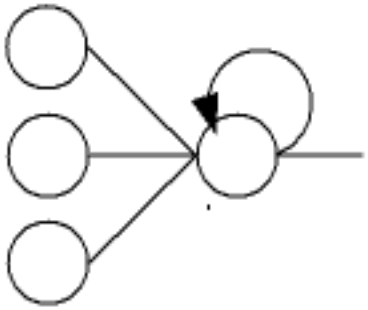
1. Introducción
2. Arquitecturas de R.N. Recurrentes
3. Algoritmos de Aprendizaje
4. Bibliografía

1. Introducción

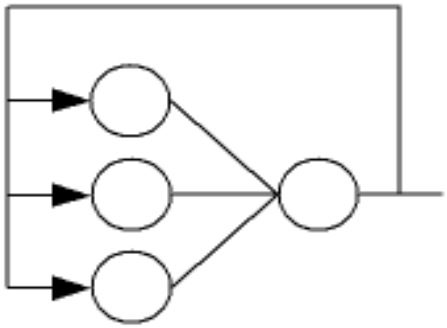
- RNN nacieron en la década de los años 80.
- Dificultad de entrenamiento
- Requerimientos computacionales
- Trabajan con Series Temporales.

Tipos de Recurrencia

Recurrencia Local



- Recurrencia Global



- Recurrencia Mixta: Cuando se dan recurrencia local y global al mismo tiempo

Tipos de Retroalimentación

- ✓ Tomando como base un Perceptron multicapa, la retroalimentación puede ser:
  - De las neuronas de salida a la capa de entrada.
  - De las neuronas ocultas de la red a la capa de entrada.
  - Si tiene dos o más capas ocultas, las posibles formas de retroalimentación global se amplian.

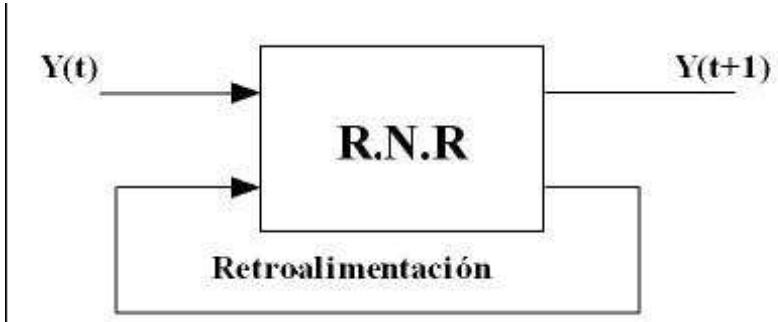
Usos Funcionales

- ✓ Básicamente, hay dos usos funcionales:
  - Memorias asociativas
  - Redes de aplicación de entrada-salida.

Redes de Aplicación de Entrada-Salida

- ✓ Dos Formas de Utilización:
  - Predicción
  - Clasificación

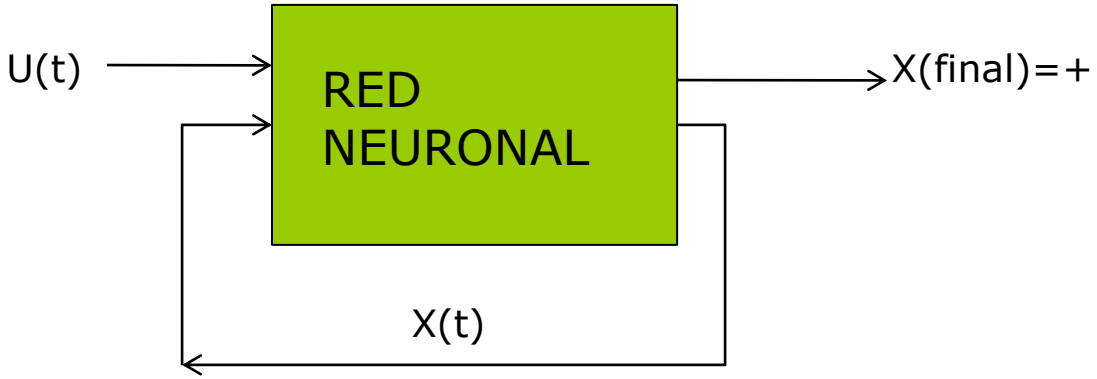
Redes de Aplicación de Entrada-Salida: Predicción



Y(0)=9.58,Y(1)= 9.62, Y(2)=9.57,Y(3)=9.70, …etc

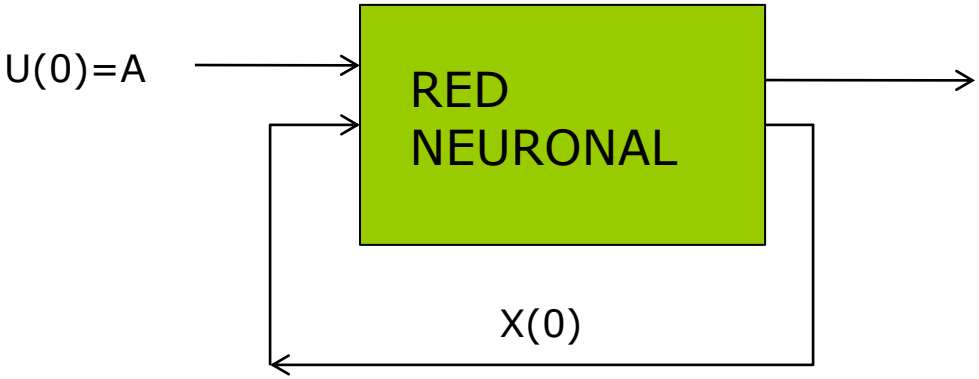
Redes de Aplicación de Entrada-Salida: Clasificación

Secuencia	Clase
ATTCG	POSITIVO
ATTCGG	POSITIVO
CCGATTG	NEGATIVO
GATTCGATA	POSITIVO
TTA	NEGATIVO



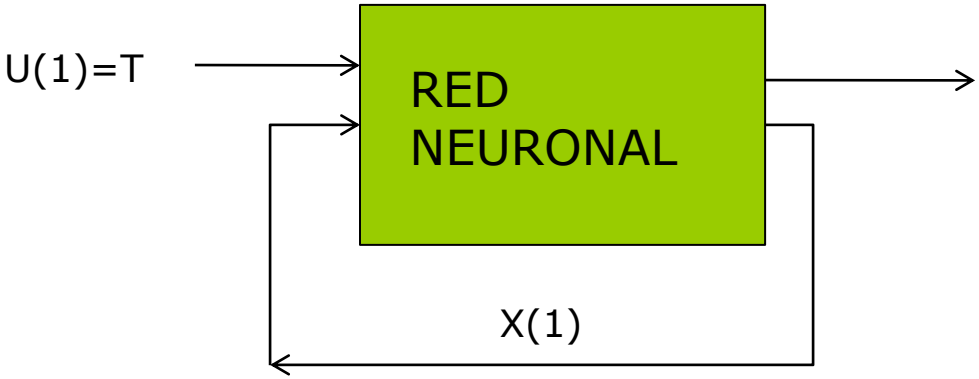
Redes de Aplicación de Entrada-Salida: Clasificación

Secuencia	Clase
ATT	POSITIVO



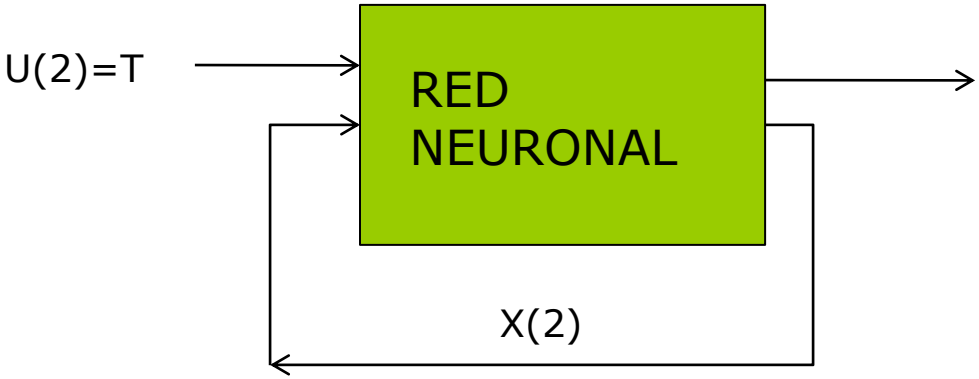
Redes de Aplicación de Entrada-Salida: Clasificación

Secuencia	Clase
ATT	POSITIVO



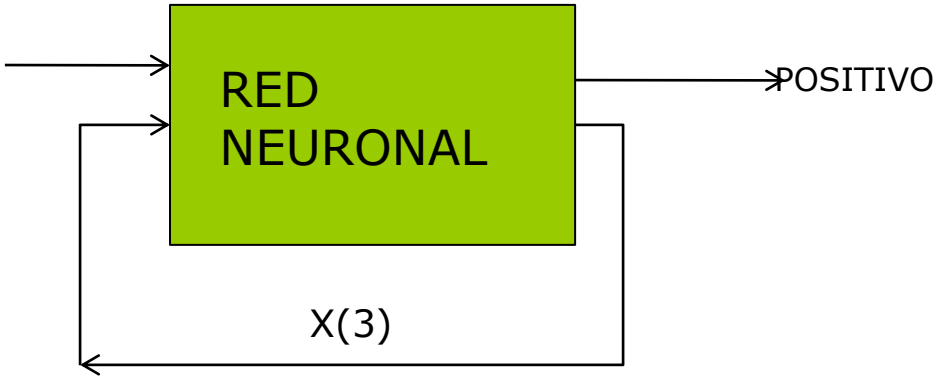
Redes de Aplicación de Entrada-Salida: Clasificación

Secuencia	Clase
ATT	POSITIVO



Redes de Aplicación de Entrada-Salida: Clasificación

Secuencia	Clase
ATT	POSITIVO



Aplicaciones

- ✓ Adquieren representaciones de estados, las hacen apropiadas para aplicaciones diversas como:

  - Predicción Series Temporales
  - Identificación de Sistemas
  - Ecualización Adaptativa de Canales de Comunicación
  - Procesamiento del Habla
  - Control de Plantas
  - Diagnóstico de Automóviles
  - Medicina
  - Reconocimiento de Patrones

Comportamiento

- Las Redes Recurrentes ofrecen una alternativa a las Redes Feedforward Dinámicas
- Actualmente se comportan mejor en estas aplicaciones
- El uso de retroalimentación global y mixta tiene el potencial de reducir significativamente los requerimientos de memoria

2. Arquitecturas de Redes Recurrentes Dinámicas

- Describiremos varias arquitecturas generales y concretas
- Tienen las siguientes características:
  - Tienen de base: Perceptron Multicapa Estático
  - Explotan la capacidad de aplicación no lineal del Perceptron Multicapa.

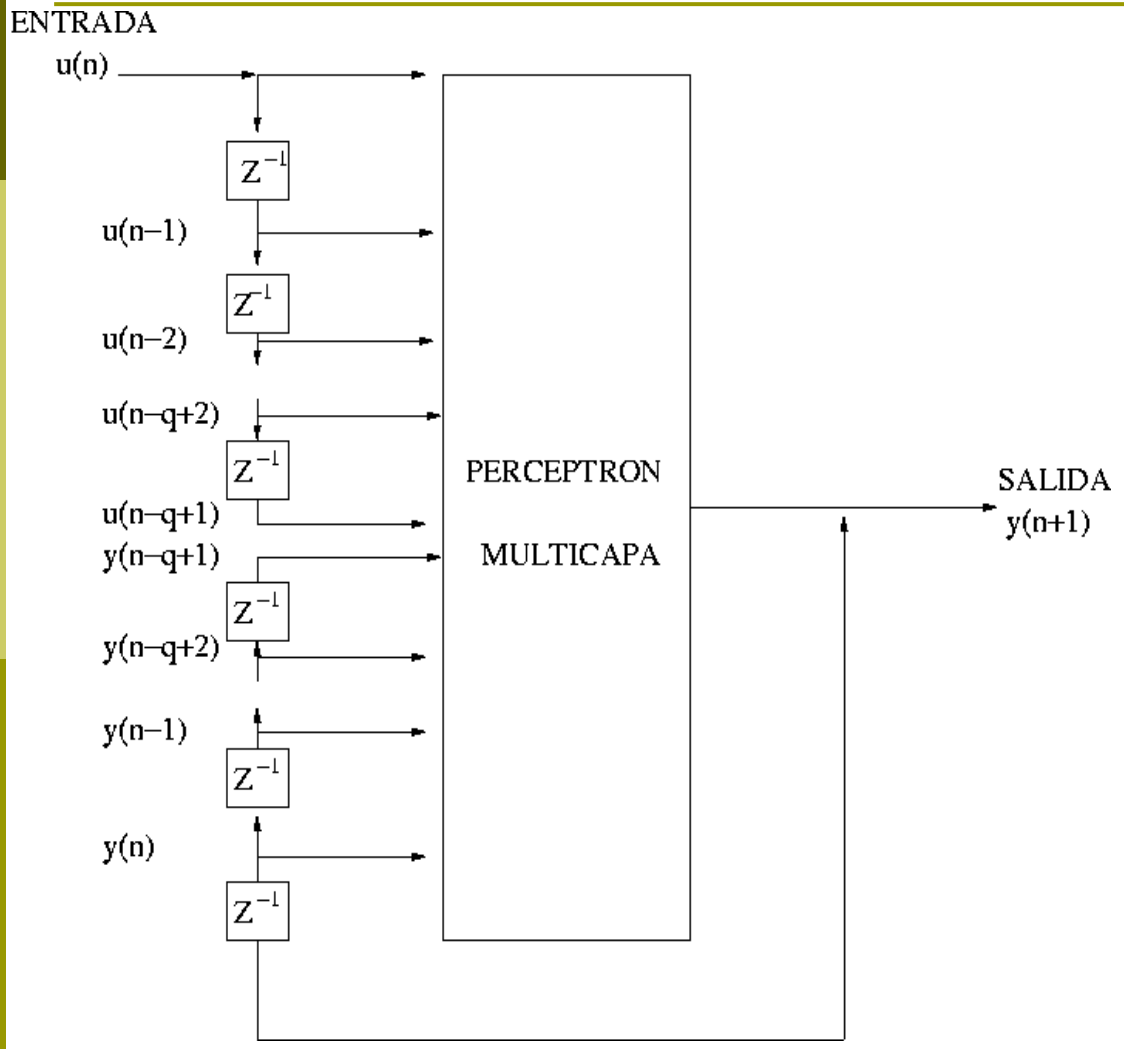
Modelos Generales

- Modelo Recurrente de Entrada-Salida
- Modelo de Espacio-Estados
- Perceptron Multicapa Recurrente

Modelos Concretos

- Redes de Segundo Orden
- Red Neuronal Recurrente Completa
- Red Neuronal Recurrente de Elman
- Red Neuronal Recurrente de Jordan

2.1 Modelo Recurrente de Entrada-Salida



u(n): valor presente de la entrada del modelo

y(n+1): el valor correspondiente de la salida, la salida es dirigida de la entrada por cada unidad de tiempo.

2.1 M. R. Entrada-Salida(2)

- Una única entrada que se aplica a la memoria de linea de retrasos de q unidades

- Una única salida que es retroalimentada a la entrada mediante otra memoria de línea de retrasos también de q unidades

- Los contenidos de estas dos memorias de líneas de retrasos se usan para alimentar la capa de entrada del perceptron multicapa.

2.1 M. R. Entrada-Salida(4)

- Esta es referida en la literatura como modelo autoregresivo no lineal de entradas exógenas (NARX).

- Comportamiento dinámico del modelo NARX

$$y(n+1) = F(y(n),\ldots,y(n-q+1),u(n),\ldots,u(n-q+1))$$

- F : función no lineal de sus argumentos.

2.2 Modelo de Espacio-Estados

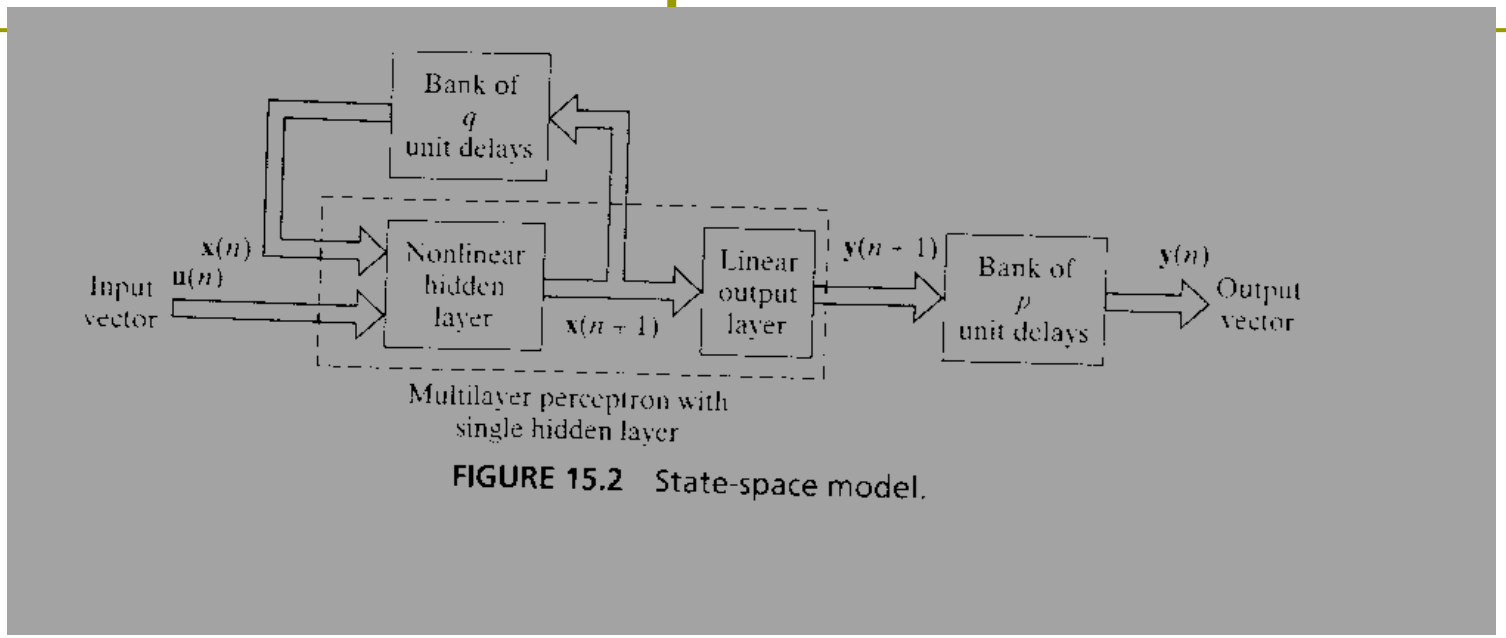


FIGURE 15.2 State-space model.

- Las neuronas ocultas definen el estado de la red

- La salida de la capa oculta es retroalimentada la capa de entrada mediante un banco de unidades de retraso.

- La capa de entrada consta de una concatenación de nodos retroalimentados y nodos fuente.

2.2 M. R. Espacio-Estados(2)

- Comportamiento Dinámico

$$x(n+1) = f(x(n), u(n))$$

$$y(n) = Cx(n)$$

  - f(.,.):función no lineal que caracteriza la capa oculta
  - C: matriz pesos sinápticos, caracterizan capa de salida.
  - La capa oculta es no lineal, la capa de salida es lineal.

2.3. Perceptron Multicapa Recurrente

- Puskorius et al. 1996
- Tiene una o más capas ocultas
- Cada capa de cálculo de un RMLP tiene retroalimentación a ella misma

2.3.Perceptron Multicapa Recurrente(2)

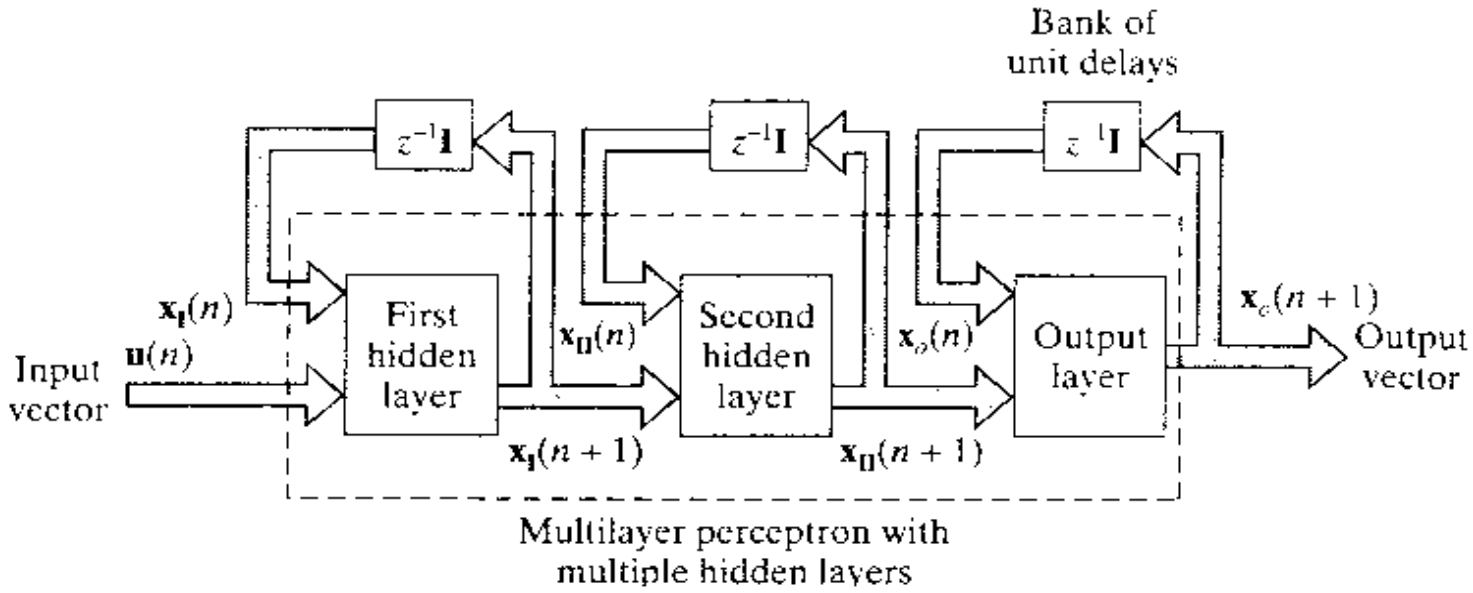


FIGURE 15.4 Recurrent multilayer perceptron.

- xI(n):salida de la primera capa oculta
- xII(n):salida de la segunda capa oculta
- xo(n):salida de la capa salida

2.3. Perceptron Multicapa Recurrente(3)

- Comportamiento Dinámico:

$$x_I(n+1) = \varphi_{II}(x_I(n), u(n))$$

$$x_{II}(n+1) = \varphi_{II}(x_{II}(n), x_I(n+1))$$

.

.

.

$$x_o(n+1) = \varphi_0(x_o(n), x_K(n+1))$$

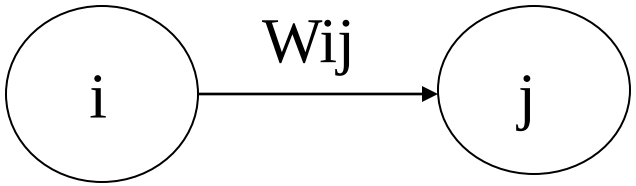
- Engloba al modelo espacio-estados
- La salida del RMLP o cualquier capa oculta no está restringida a tener una forma particular en la función activación

2.4. Redes Concretas
2.4.1 Redes de Segundo-Orden

- Concepto de Orden

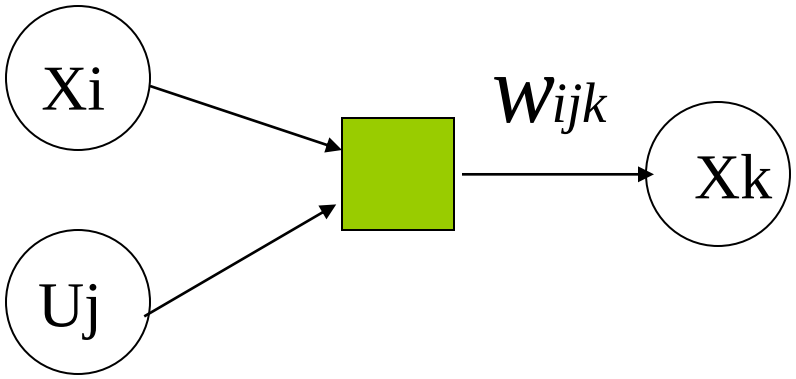
Primer Orden:

$$v_k = \sum_j w_{a,jk} . x_j + \sum_j w_{b,ik} . u_i$$



Segundo Orden:

$$v_k = \sum_i \sum_j w_{ijk} . x_i . u_j$$



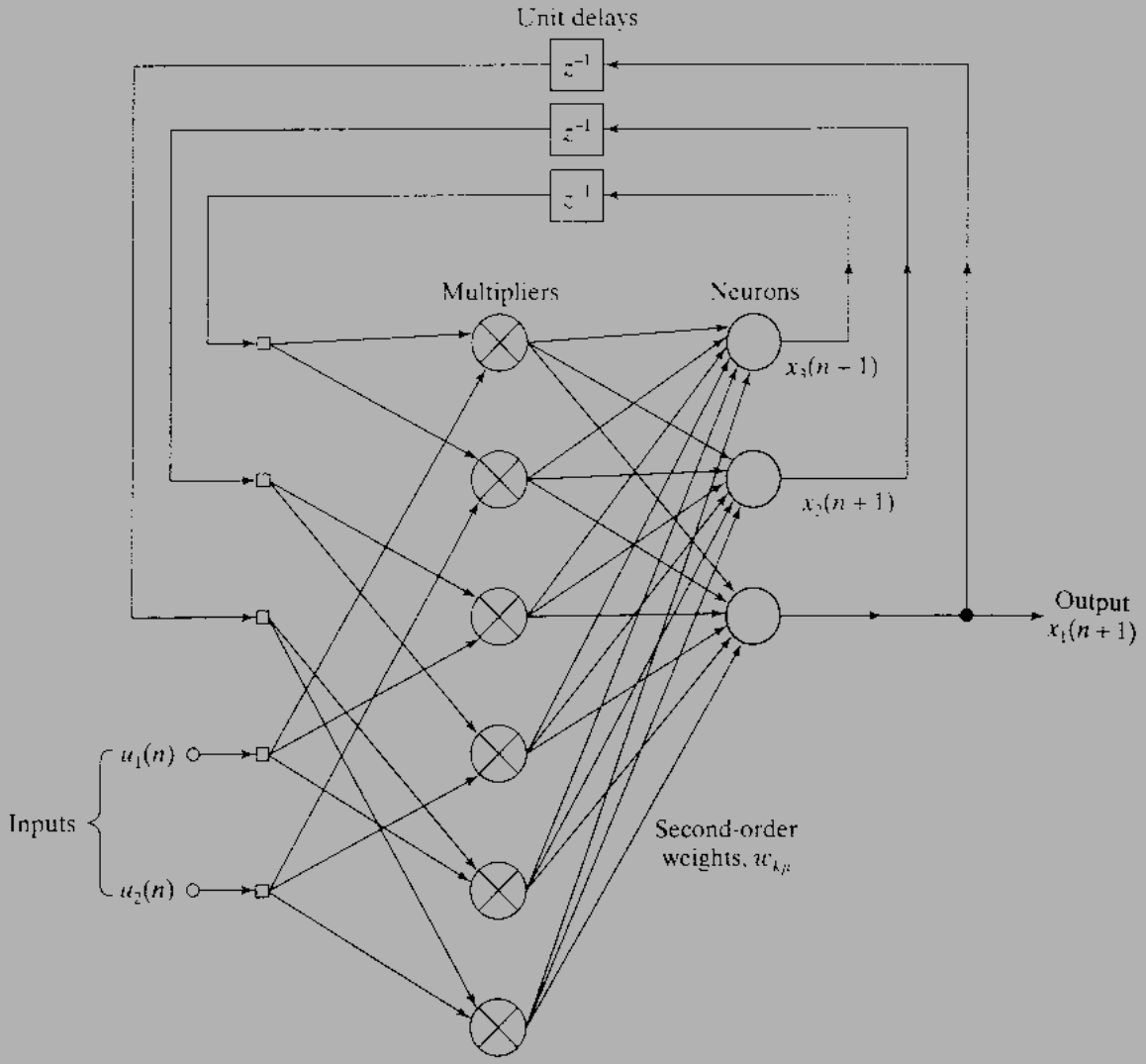


FIGURE 15.5 Second-order recurrent network; bias connections to the neurons are omitted to simplify the presentation. The network has 2 inputs and 3 state neurons, hence the need for 3 x 2 = 6 multipliers.

2.4.2 Redes de Segundo-Orden(3)

- Comportamiento Dinámico:

$$v_k(n) = b_k + \sum_i \sum_j w_{ijk} . x_i(n) . u_j(n)$$

$$x_k(n+1) = \varphi(v_k(n)) = \frac{1}{1+\exp(-v_k(n))}$$

2.4.2. Redes de Segundo-Orden(3)

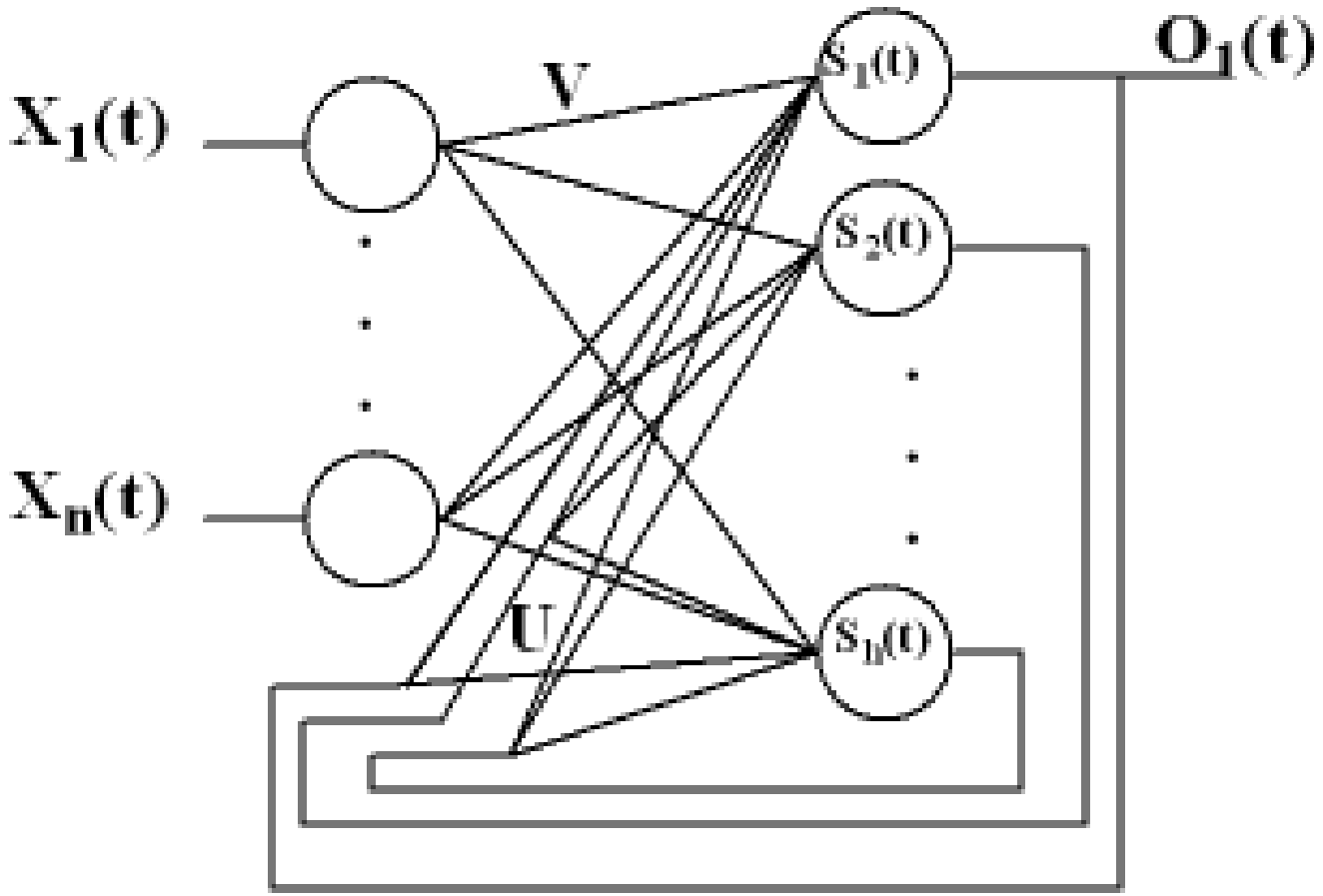
- xi(n)*uj(n) representa el par (estado,entrada)-> siguiente estado

$$\partial(x_i, u_j) = x_k$$

- Las RNRSO son utilizadas para representar y aprender DFA

- un DFA es un dispositivo de procesamiento de información con un número finito de estados.

2.4.3 Red Neuronal Recurrente Completa



2.4.3 Red Neuronal Recurrente Completa

- Capa de neuronas de entrada
- Capa de neuronas ocultas/de salida.
- Cada neurona de la capa oculta/de salida puede tener una función de activación distinta y todas se realimentan a sí mismas y al resto de neuronas de la misma capa

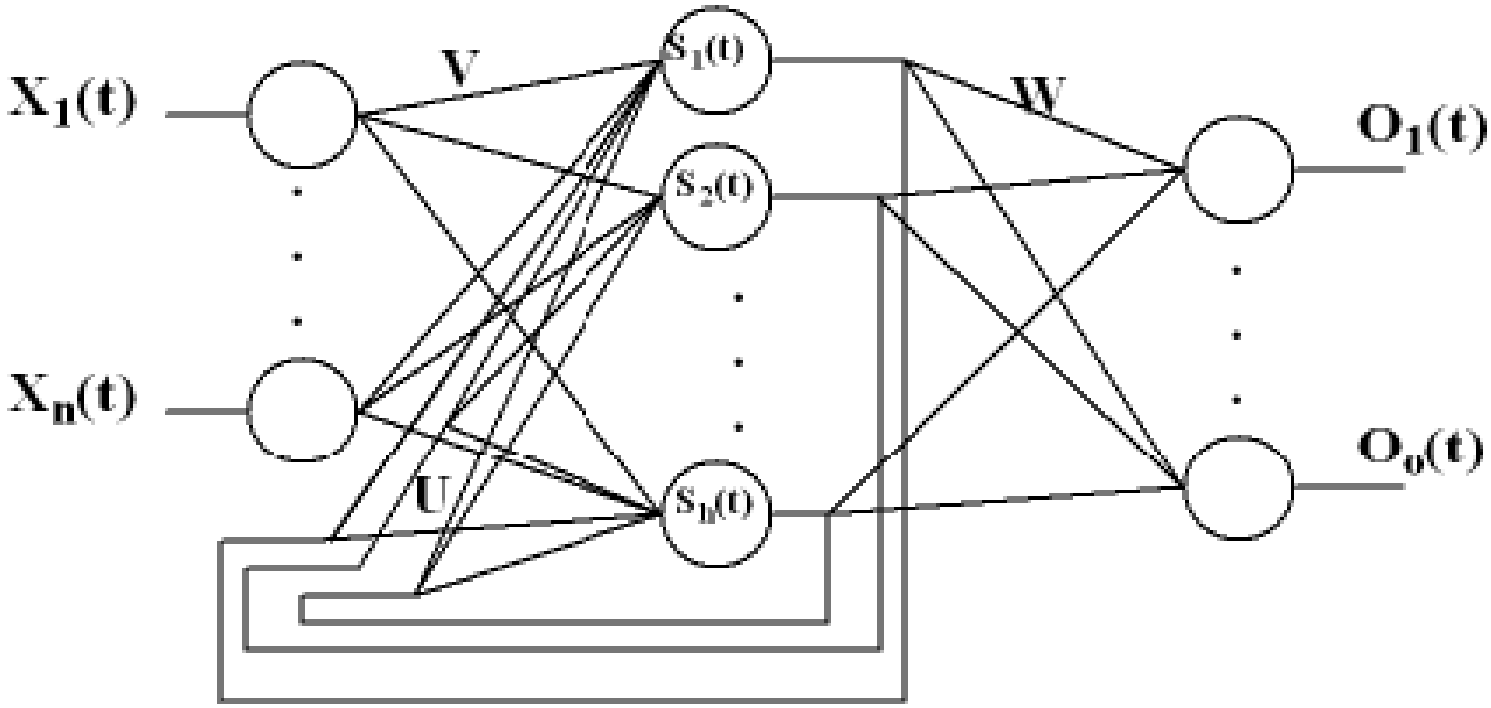
2.4.3 Red Neuronal Recurrente Completa

$$S_i(t+1) = f_i(net_i(t))$$

$$net_i(t) = \sum_j V_{ji}.X_j(t) + \sum_j U_{ji}.S_j(t)$$

$$O_k = S_k(t), 1 <= k <= o$$

2.4.4 Red Neuronal Recurrente de Elman



2.4.4 Red Neuronal Recurrente de Elman

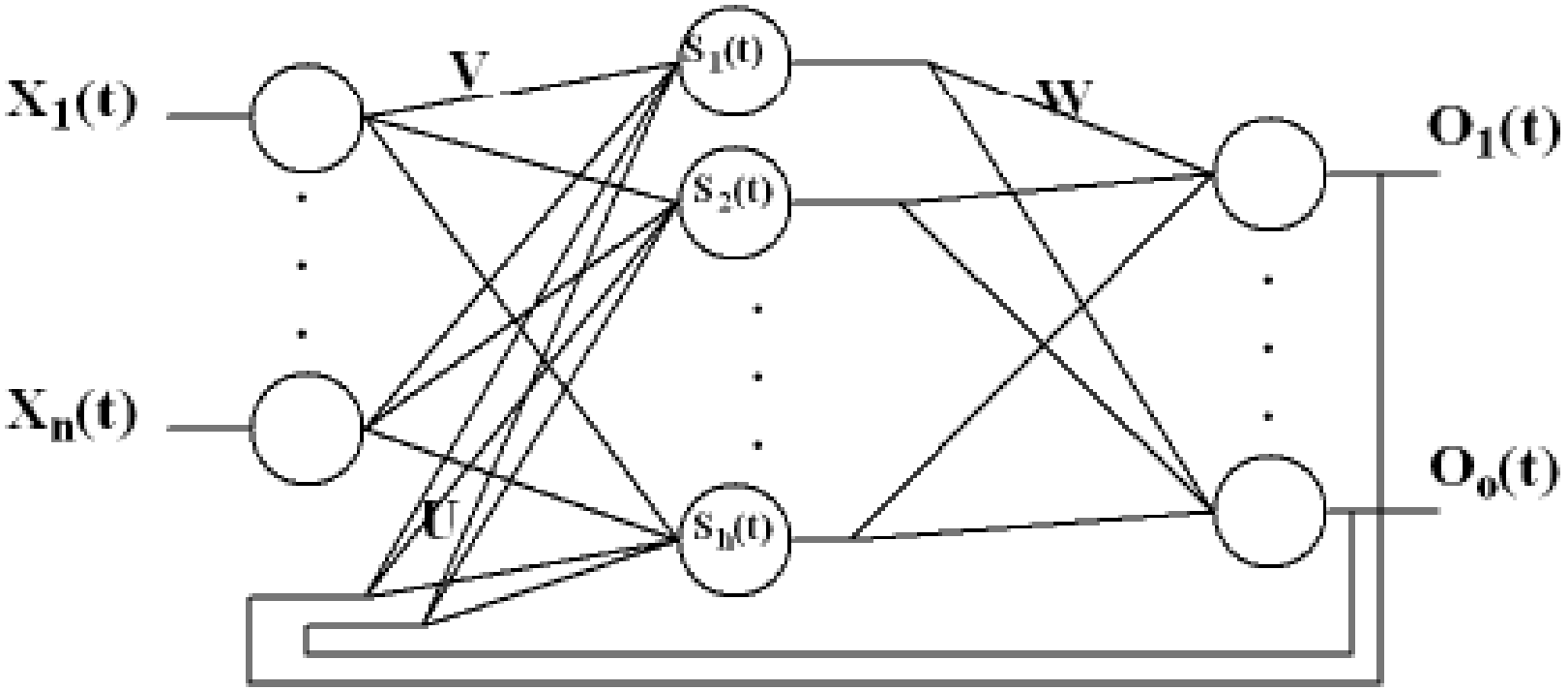
$$S_i(t+1) = f_i(net_i(t))$$

$$net_i(t) = \sum_j V_{ji}.X_j(t) + \sum_j U_{ji}.S_j(t)$$

$$netO_k(t) = \sum_j W_{jk}S_j(t)$$

$$O_k(t) = g(net_{Ok}(t))$$

2.4.5 Red Neuronal Recurrente de Jordan



2.4.5 Red Neuronal Recurrente de Jordan

$$S_i(t+1) = f_i(net_i(t))$$

$$net_i(t) = \sum_j V_{ji}.X_j(t) + \sum_j U_{ji}O_j(t)$$

$$netO_k(t) = \sum_j W_{jk}S_j(t)$$

$$O_k(t) = g(net_{O_k}(t))$$

3. Algoritmos de Aprendizaje

- Dos algoritmos basados en el gradiente:
  - Algoritmo de Retropropagación a Través del Tiempo (BPTT)
  - Algoritmo de Aprendizaje Recurrente en Tiempo Real (RTRL)

- Características comunes:
  - Basados en el método del gradiente descendente
  - Más complicados de implementar
  - Lentos para converger

Heurísticas

- Heurísticas para la mejora del entrenamiento usando el método del gradiente descendente (Giles, 1996):

  - Orden lexicográfico de las muestras de entrenamiento secuencias cortas de símbolos presentados a la red al principio.

Heurísticas(2)

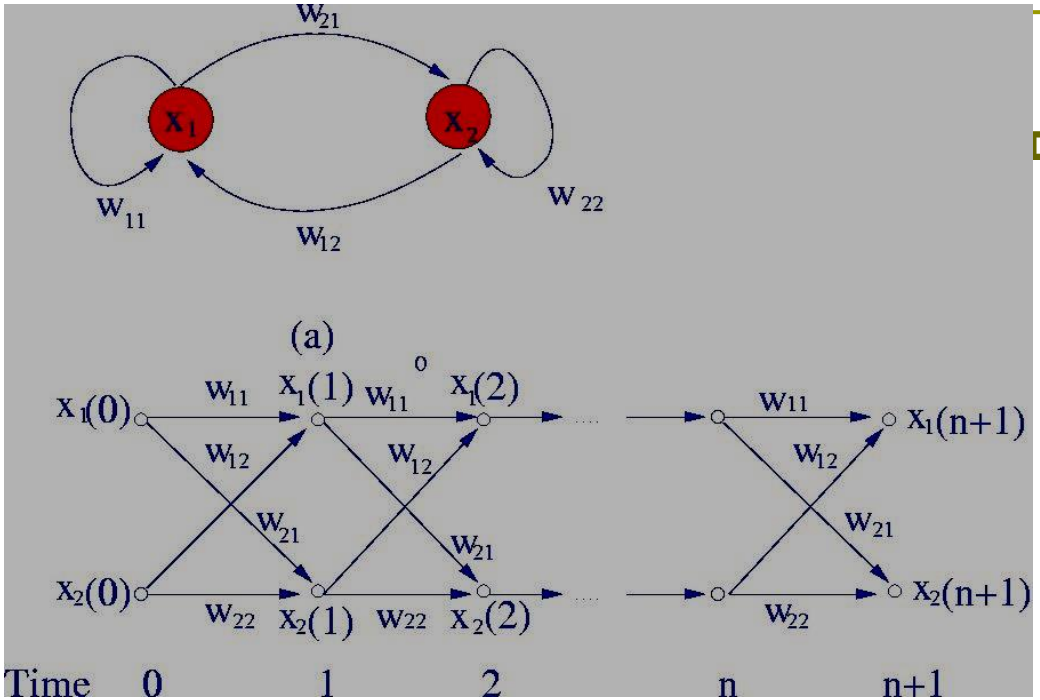
- El entrenamiento debería comenzar con muestras pequeñas e ir incrementando el tamaño conforme el entrenamiento continua

- Actualización de los pesos si el valor absoluto del error sobre la muestra procesada es mayor que un valor dado

3.1 Retropropagación a Través del Tiempo (BPTT)

- Extensión del algoritmo estándar de retropropagación
- Se puede derivar desplegando la operación temporal de la red en una red feedforward de varias capas, la topología crece una capa por cada paso de tiempo.

3.1 Retropropagación a Través del Tiempo (BPTT)(3)



-R.N.R de dos neuronas (Fig a)

- Fig. b representa la red FF de varias capas N*, una nueva capa se ha añadido en cada paso de tiempo

- Dependiendo de como se despliegue conduce a dos implementaciones diferentes del BPTT:
  - entrenamiento por épocas
  - el entrenamiento continuo (tiempo-real).

3.1 Retropropagación a Través del Tiempo (BPTT)(2)

- *La red desplegada N* se relaciona con la red original N como sigue:*

1. *Para cada paso de tiempo, la red N* tiene una capa conteniendo K neuronas, donde K es el número de neuronas contenidas en la red N.*

2. *En cada capa de la red N* hay una copia de cada neurona en la red N.*

3. *Para capa paso de tiempo , la conexión sináptica a partir de la neurona i en la capa l para la neurona j en la capa l+1 de la red N* es una copia de la conexión sináptica a partir de la neurona i a la neurona j de la red*

3.1.1 BPTT por Épocas

- Cada patrón del conjunto de entrenamiento va a representar una época.

- Sea t0 el tiempo de comienzo de una época y t1 su tiempo final. Dada estas épocas:

$$\varepsilon_{total}(t_0, t_1) = \frac{1}{2} \sum_{n=n0}^{n1} \sum_{j \in A} e_j^2(t)$$

j indice neuronas de salida,

$e_j^2(t)$ señal de error de la salida

3.1.1 BPTT por Épocas(2)

1. Desarrollar un paso hacia delante de los datos para el intervalo (t0,t1).

2. Guardaremos:
   - el registro completo de los datos de entrada
   - estado de la red
   - respuestas deseadas sobre el intervalo

3.1.1 BPTT por Épocas(3)

2. Realizar un paso hacia atrás con los datos anteriores, calculando los valores de los gradientes locales:

$$\partial_j(t) = \frac{\partial \mathcal{E}_{total}(t_0, t_1)}{\partial v_j(t)}$$ para todo j, n0<n<=n1

$$\partial_j(t) = \begin{cases} \varphi'(v_j(t)).e_j(t) & t = t1 \\ \varphi'(v_j(t))\left[e_j(t) + \sum_{k \in A} w_{jk}\partial_k(t+1) \right] & t_0 < t < t_1 \end{cases}$$

- Esta expresión se repite, comenzando desde el tiempo t1 y trabajando hacia atrás, paso a paso, hasta llegar a t0.
- El número de pasos desarrollados es igual al número de pasos de tiempo contenidos en la época.

3.1.1 BPTT por Épocas(3)

3. Ajustar el peso wji

$$\Delta w_{ji} = -\eta \frac{\partial_{total}(t_0, t_1)}{\partial w_{ji}} = \eta \sum_{t=t_0+1}^{t_1} \partial_j(t) x_i(t-1)$$

Donde η es la razón de aprendizaje y xi(t-1) es la entrada aplicada a la sinápsis i-ésima de la neurona j en el tiempo t-1

3.1.2. BPTT Truncado

- algoritmo de retropropagación a través del tiempo truncada (BPTT(h)) (Williams and Peng, 1990).

- Para usar BPTT en tiempo real usamos los valores instantáneos de la suma del cuadrado de los errores :

$$\varepsilon(t) = \frac{1}{2}\sum_{j \in A} e_j^2(t)$$

- Profundidad de truncamiento (h): grabaremos la historia relevante de los datos de entrada y el estado de la red para un numero fijo de pasos de tiempo

3.1.2. BPTT Truncado(2)

- Cualquier información más vieja que h pasos de tiempo en el pasado se considera irrelevante, y puede por tanto ser ignorada.
- gradiente local para la neurona j se define por:

$$\partial_j(l) = -\frac{\partial \varepsilon(l)}{\partial v_j(l)}$$ para todo $j \in A$ y $t-h < l < t$

Esto nos lleva a la fórmula:

$$\partial_j(l) = \begin{cases} \varphi'(v_j(l)).e_j(l) & l = t \\ \varphi'(v_j(l))\left[\sum_{k \in A} w_{kj}(l)\partial_k(l+1)\right] & t-h < l < t \end{cases}$$

3.1.2. BPTT Truncado(3)

- Una vez el cálculo de la retropropagación se ha desarrollado hacia atrás en el tiempo t-h+1, el siguiente ajuste se aplica a los pesos sinápticos de la neurona j:

$$\nabla w_{ji}(t) = \eta \sum_{l=t-h+1}^{t} \partial_j(l) x_i(l-1)$$

- η debe de ser suficientemente pequeño para asegurar que valores de pesos no cambian significativamente

3.2 Aprendizaje Recurrente en Tiempo-Real (RTRL).

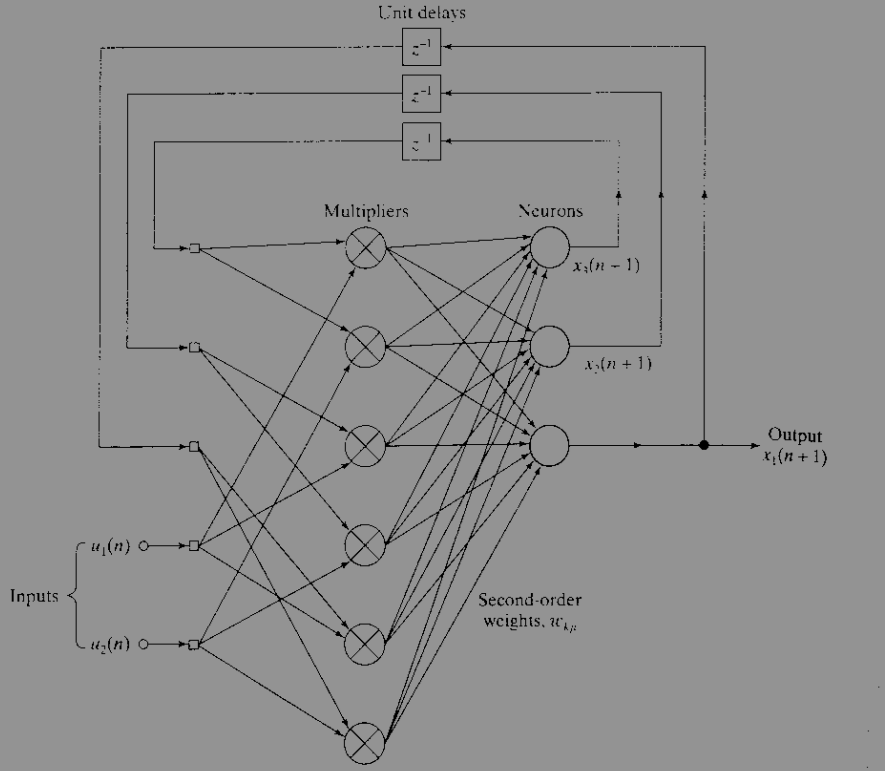


FIGURE 15.5 Second-order recurrent network; bias connections to the neurons are omitted to simplify the presentation. The network has 2 inputs and 3 state neurons, hence the need for 3 x 2 = 6 multipliers.

- Los ajustes se hacen a los pesos en tiempo real mientras la red procesa su señal (Williams y Zipser, 1989).

3.2 Aprendizaje Recurrente en Tiempo-Real (RTRL)(2)

$$v_k(t) = b_k + \sum_i \sum_j w_{kij}.x_i(t).u_j(t) \qquad x_k(t+1) = \frac{1}{1+\exp(-v_k(t))}$$

$$\Delta w_{lmn} = -\eta \frac{\partial E}{\partial w_{lmn}} = \eta(d(t+1) - x_0(t+1)) \frac{\partial x_0(t+1))}{\partial w_{lmn}}$$

$$\frac{\partial x_i(t+1)}{\partial w_{lmn}} = \varphi'(v_i).\partial_{il}.x_m(t).u_n(t) + \sum_{j,k} w_{ijk}.u_k(t).\frac{\partial x_j(t)}{\partial w_{lmn}}$$

Inicialmente (t=0), se considera que las derivadas parciales valen 0

3.3 Otros algoritmos de aprendizaje

- En la actualidad se han desarrollado otros algoritmos adaptados que se muestran más estables y eficientes en la búsqueda:

  - Algoritmos Genéticos o Evolutivos
  - Búsqueda Tabú
  - Enfriamiento Simulado
  - Etc,

4. Aplicaciones

- Identificación de Sistemas
- Reconocimiento de Voz
- Inferencia Gramatical
- Reconocimiento de ADN
- Predicción de Series Temporales

Identificacion de Sistemas

- Modelización de un proceso o una planta de parámetros desconocidos:
  - Planificación experimental
  - Selección de una estructura del modelo
  - Estimación de parámetros
  - Validación del modelo.

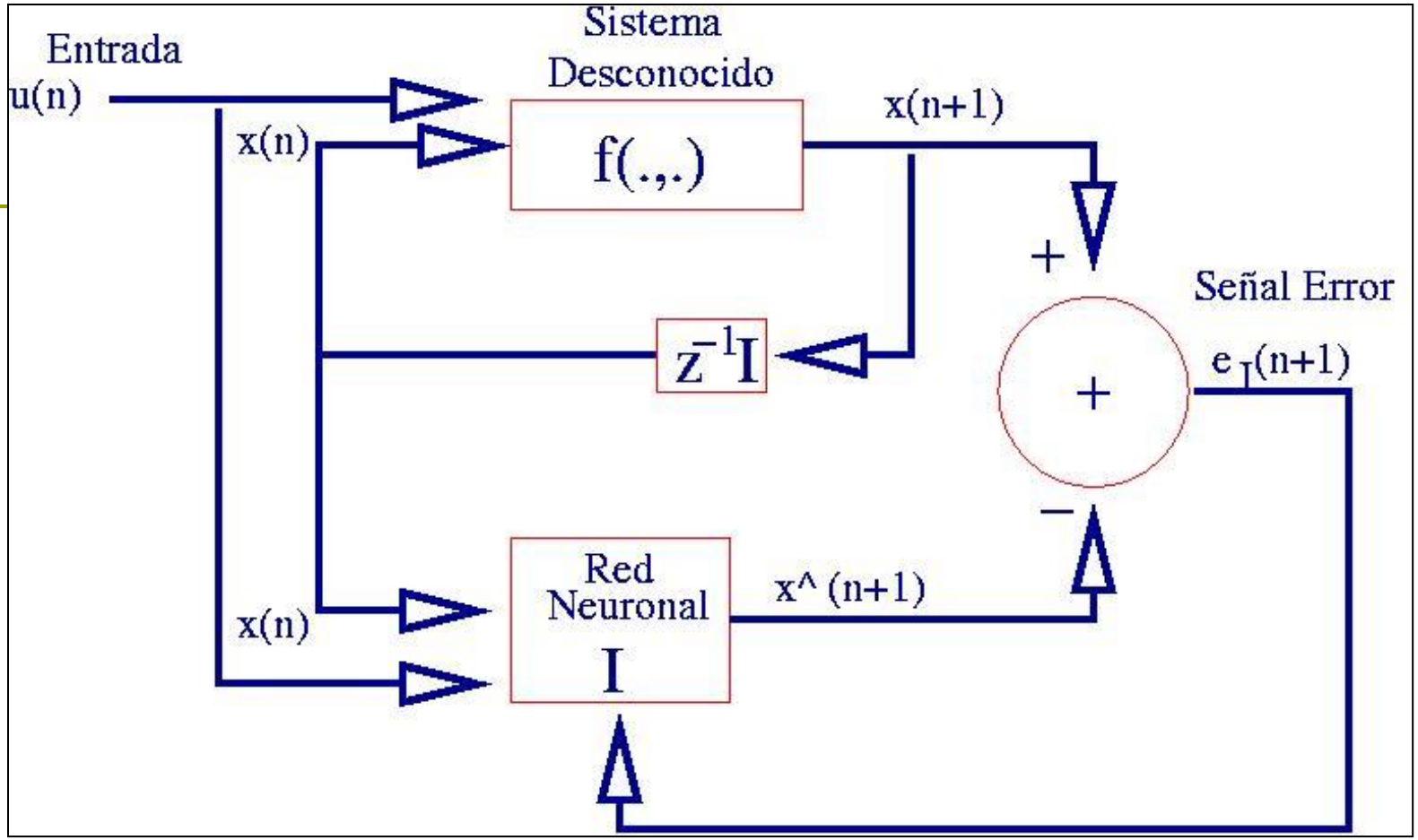
Identificación de Sistemas usando Modelo de Espacio-Estados

- Descripción de la planta:

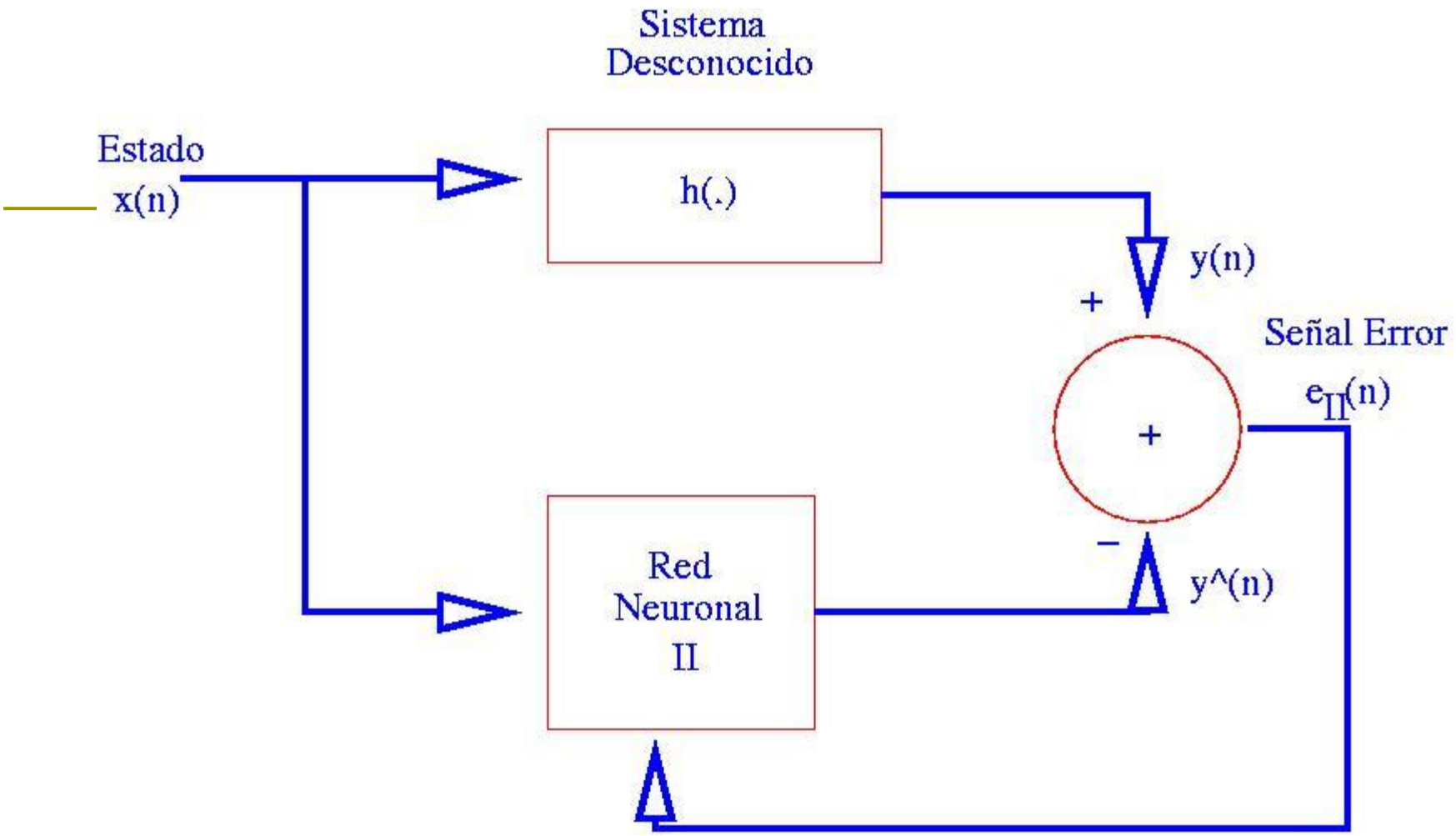
$$x(n+1) = F(x(n), u(n))$$
$$y(n) = H(x(n))$$

donde F(´.´) y H(.) son funciones no lineales, ambas se suponen que son desconocidas;

- Usamos dos redes neuronales para identificar el sistema y la otra para aprender la salida



$$e_I(n+1) = x(n+1) - \hat{x}(n+1)$$



$$e_{II}(n) = y(n) - \hat{y}(n)$$

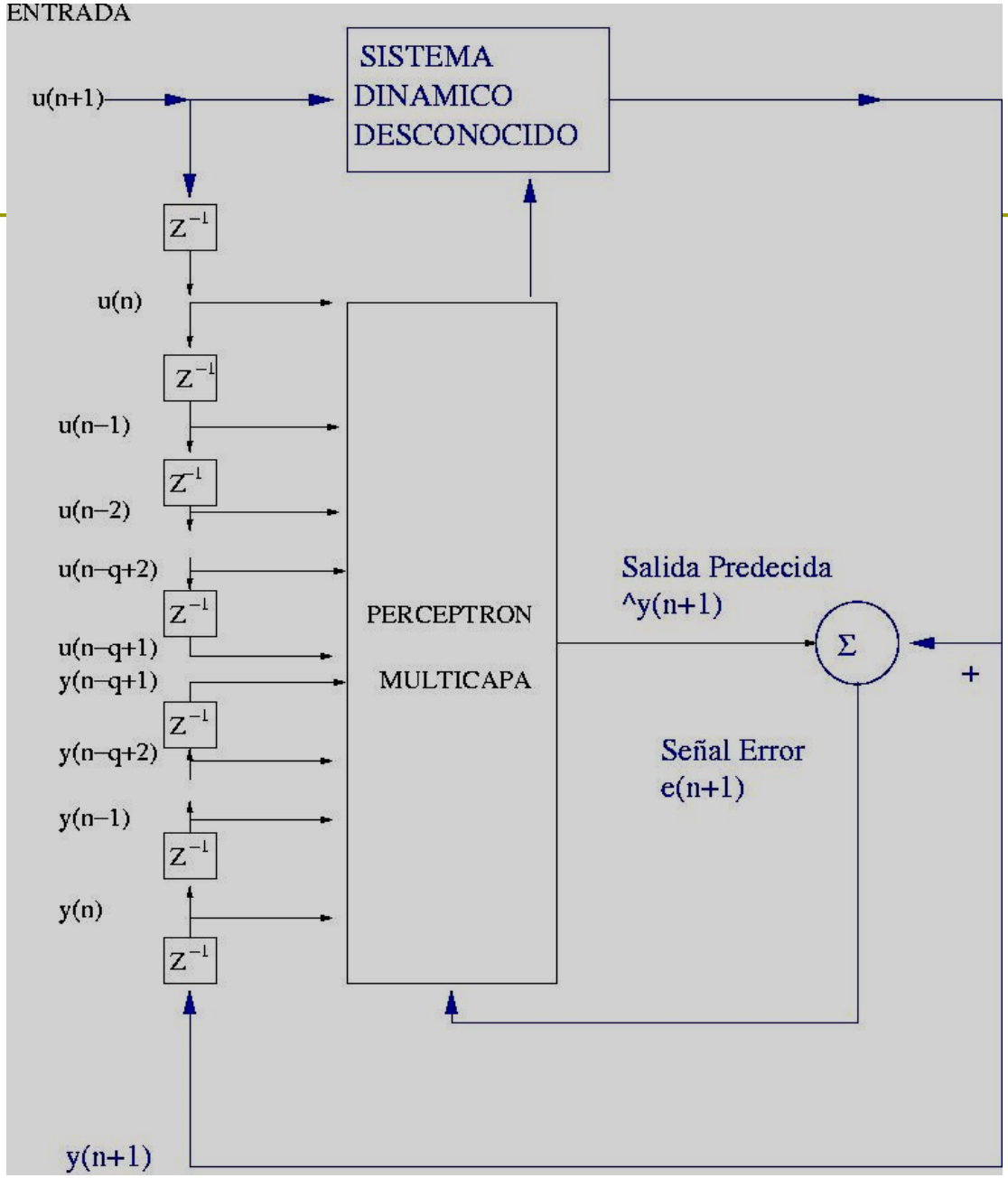
Identificación de Sistemas usando el Modelo de Entrada-Salida

- Supongamos que la planta desconocida es únicamente accesible a través de su salida.
- Sea y(n) denota la salida del sistema debida a la entrada u(n) que varía en el tiempo-discreto n, si utilizamos NARX, el modelo de Identificación es:

$$\hat{y}(n+1) = \varphi(y(n),\ldots,y(n-q+1),u(n),\ldots,u(n-q+1))$$

- Error:

$$e(n+1) = y(n+1) - \hat{y}(n+1)$$


ENTRADA
SISTEMA
DINAMICO
DESCONOCIDO
u(n+1)
Z⁻¹
u(n)
u(n−1)
u(n−2)
u(n−q+2)
u(n−q+1)
y(n−q+1)
y(n−q+2)
y(n−1)
y(n)
PERCEPTRON
MULTICAPA
Salida Predecida
^y(n+1)
Σ
+
Señal Error
e(n+1)
y(n+1)

5. Bibliografía

- "Neural Networks. A comprehensive Foundation", Simon Haykin, Prentice Hall , Second Edition.
- Omlin, Giles, Thornber, "Equivalence in knowledge representation:automata, Recurrent Neural Networks Design and Applications, The CRC Press International Series on Computational Intelligence, 2000,pp. 99-131
- Blanco, Delgado, Pegalajar, "Identification of fuzzy dynamic systems using max-min recurrent neural networks", Fuzzy Sets and Systems, 2000

Reconocimiento de Voz mediante Redes Neuronales Recurrentes Dinámicas

M.Carmen Pegalajar
Dpto. Ciencias de Comput. E I.A.
E-mail:mcarmen@decsai.ugr.es

Aplicaciones

- Objetivo: Concepción y realización de sistemas automáticos que son capaces de interpretar la señal vocal humana en términos de categorías lingüísticas de un universo dado.

- Aplicaciones de sistemas orales:
  - Clásicas:manejo y clasificación de paquetes piezas, control de máquinas-herramienta
  - Reemplazar por completo al humano en tareas rutinarias: teléfono automático de respuestas o información, servicios documentales a partir de una base de datos, etc.
  - Sistemas de ayuda a discapacitados: sistemas para aprender a hablar, prótesis sintéticas de palabras, sillas de ruedas controladas por el habla.
  - Sistemas de comunicación oral con computadores.

Problema y Ejemplos

- Problema: Reconocer la pronunciación de los 10 dígitos: {cero, uno, dos, tres, cuatro, cinco, seis, siete, ocho, nueve}

- Conjunto de ejemplos: diez locutores diferentes, femeninos y masculinos, pronuncian 10 veces cada numero obteniendo un conjunto de ejemplos de 1000 muestras

Ejemplos

- Los datos han sido adquiridos a una frecuencia de 8533Hz.
- Se llevó a cabo una cuantización vectorial y un proceso de etiquetado (16 símbolos): {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
- Ejemplos:

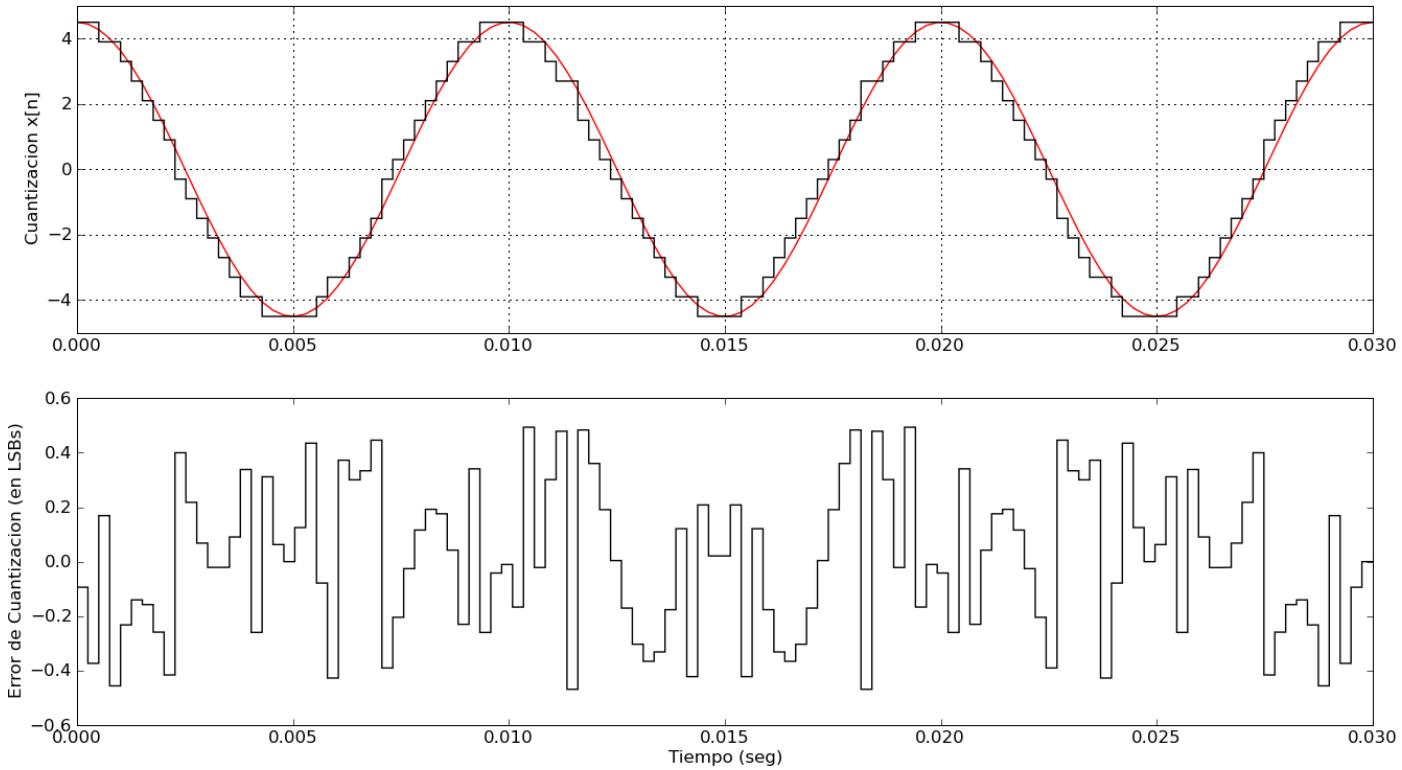
2222256188888 81D888CC994444 CERO

494BBBBBBCC222266166CCCCC99944 CUATRO

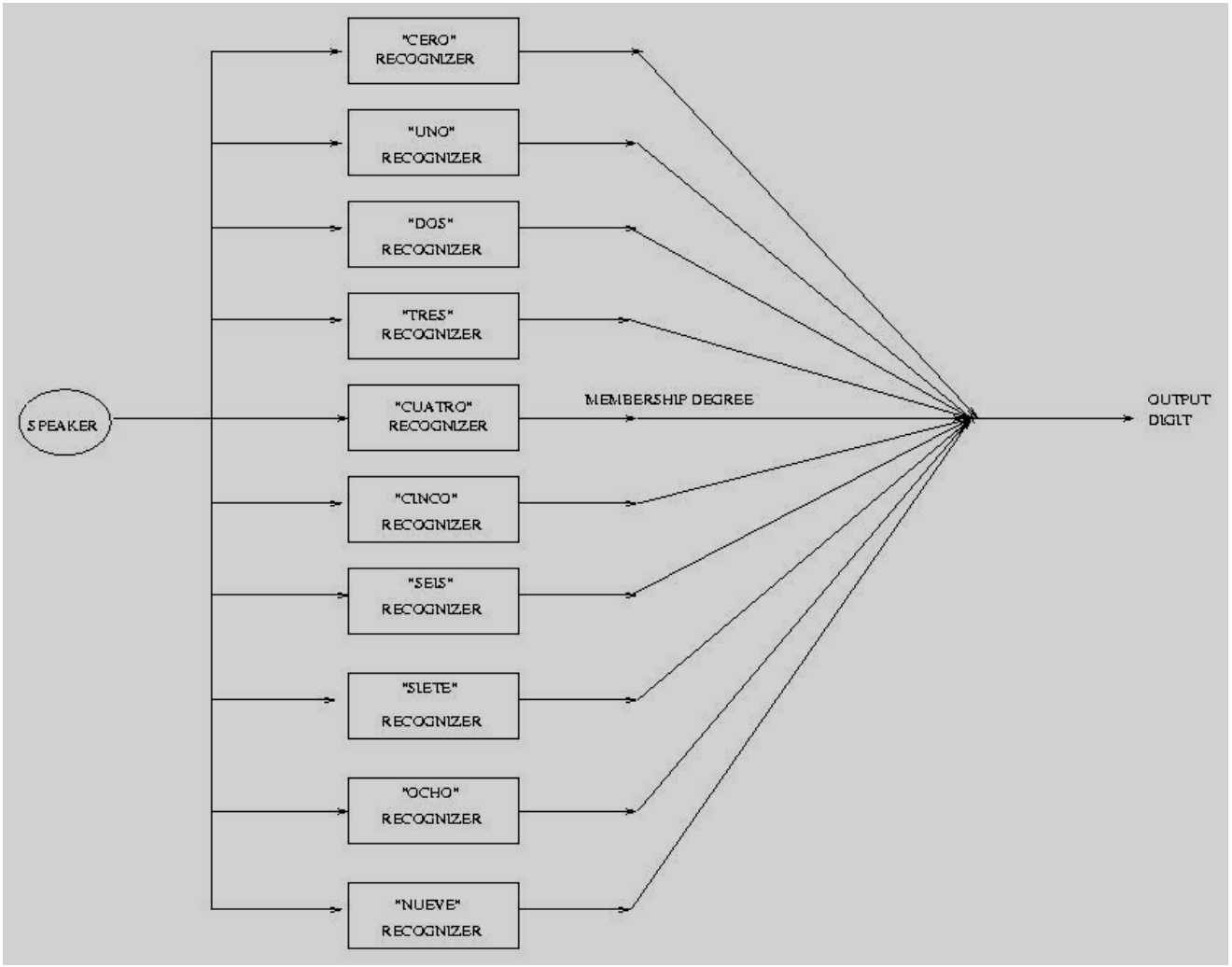
Secuencias con longitud diferente, FFNN no son apropiadas

Ejemplos

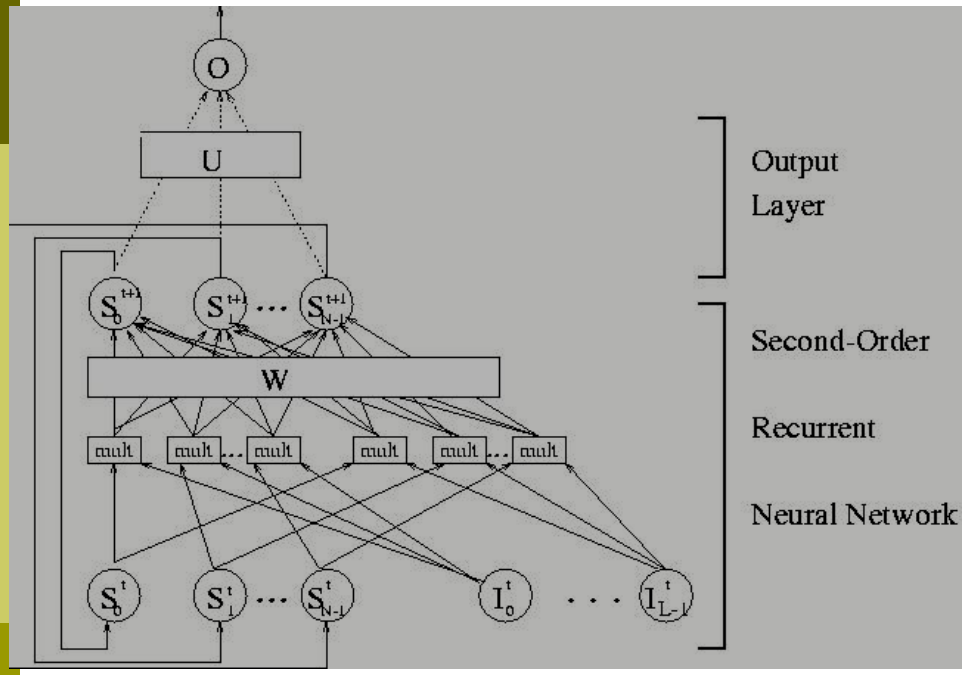
- Cuantización Vectorial:



Sistema Reconocedor Neuronal



Modelo Neuronal: SORNN



- N neuronas recurrentes
- L neuronas de entrada (L=16).
- N*N*L pesos
- Una neurona de salida
- N pesos de salida

$$S_i^{t+1} = g(\theta_i^t)$$

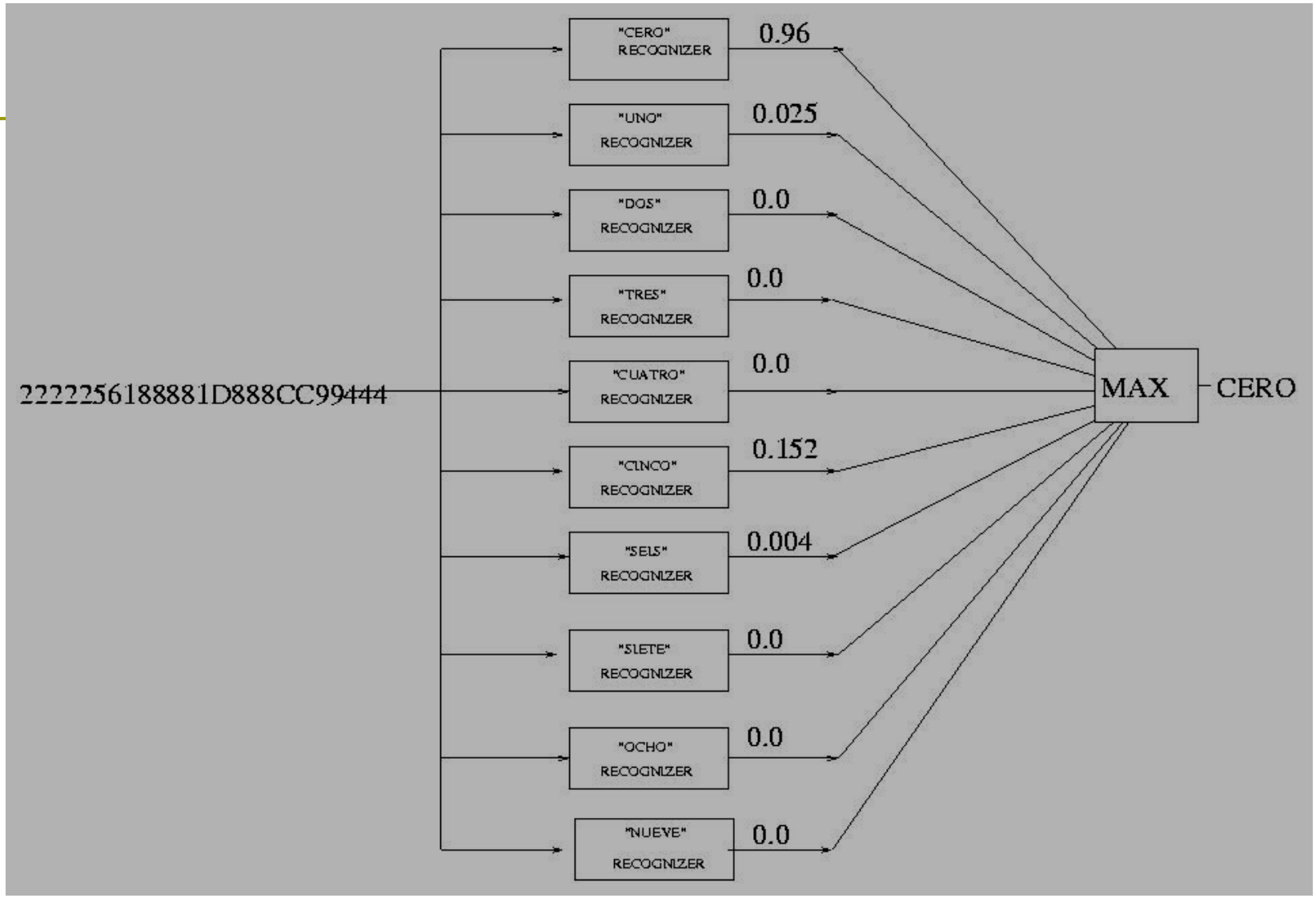
$$\theta_i^t = \sum_{j=0}^{N-1} \sum_{k=0}^{L-1} I_k^t . S_j^t . w_{ijk}$$

$$g(x) = \frac{1}{1+e^{(-x)}}$$

$$O = \sum_{i=0}^{N-1} u_i . S_i^m$$

Topologías Neuronales obtenidas

RECONOCEDO	N	α	ε	μ	CICLOS	%	ERROR
CERO	2	0.1	0.0	0.1	2544	100%	0.00354
UNO	2	0.1	0.0	0.1	2188	100%	0.00409
DOS	2	0.2	0.9	0.1	3100	95%	0.01793
TRES	3	0.1	0.0	0.1	1907	93%	0.012
CUAT	4	0.1	0.0	0.1	2476	91%	0.010209
CINC	3	0.1	0.0	0.1	2029	92%	0.023
SEIS	3	0.1	0.0	0.1	3100	99.5	0.0025
SIET	3	0.1	0.9	0.1	1657	95%	0.006044
OCHO	3	0.1	0.0	0.1	2126	94%	0.05
NUEVE	3	0.1	0.0	0.1	3143	91.05	0.01042


2222256188881D888CC99444
"CERO" RECOGNIZER
"UNO" RECOGNIZER
"DOS" RECOGNIZER
"TRES" RECOGNIZER
"CUATRO" RECOGNIZER
"CINCO" RECOGNIZER
"SEIS" RECOGNIZER
"SIETE" RECOGNIZER
"OCHO" RECOGNIZER
"NUEVE" RECOGNIZER
0.96
0.025
0.0
0.0
0.0
0.152
0.004
0.0
0.0
0.0
MAX
CERO

Referencias

- Inferencia Gramatical mediante Redes Neuronales, Tesis Doctoral, M.Carmen Pegalajar
- Speech Recognition using Fuzzy Second-Order Recurrent Neural Networks, IWANN 2001, Blanco A., Delgado M., PegalajarM.C., Requena I.
- A Real-Coded Genetic Algorithm for Training Recurrent Neural Networks. Neural Network 14 (2001) 93-105. Blanco, Delgado and Pegalajar
- Casacuberta F., Vidal E. :Reconocimiento automático del habla". Ed. Marcombo, 1987.

Inferencia Gramatical mediante Redes Neuronales Recurrentes Dinámicas

M.Carmen Pegalajar
Dpto.Ciencias de la Computación
mcarmen@decsai.ugr.es

IG mediante RNRD

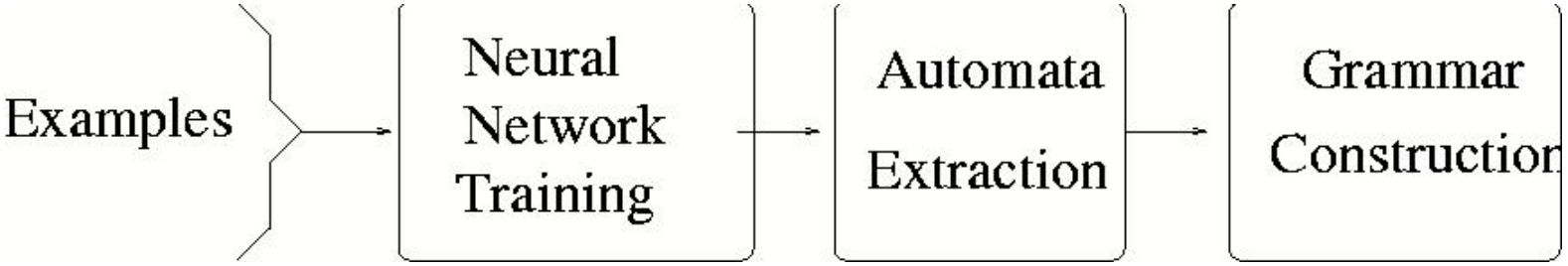
1. Introducción
2. Inferencia Gramatical mediante RN
3. Arquitecturas para realizar IG
4. Algoritmo de Extracción del AF.
5. Ejemplo
6. Bibliografía

1. Inferencia Gramatical

- PROBLEMAS:
  - Reconocimiento de Voz
  - Reconocimiento de texturas y objetos en imágenes.
  - Reconocimiento de estructuras en proteínas
  - Diseño de lenguaje de programación.
  - Búsqueda de información en Textos e Internet

- SOLUCION: La inferencia gramatical es el problema de inducir las reglas gramaticales que caracterizan una gramática, a partir de un conjunto de ejemplos.

2. Inferencia Gramatical (crisp/difusa) mediante R.N.



- Limitación de las redes feedforward: ejemplos de longitud fija.
- En la actualidad, principalmente se utilizan modelos recurrentes.

ab	+
a	-
b	-
abab	+
aba	-
ababab	+
b	-
bb	-
abababab	+
ababababab	+

¿A que lenguaje pertenecen los ejemplos positivos?

¿A que lenguaje pertenecen los ejemplos positivos?

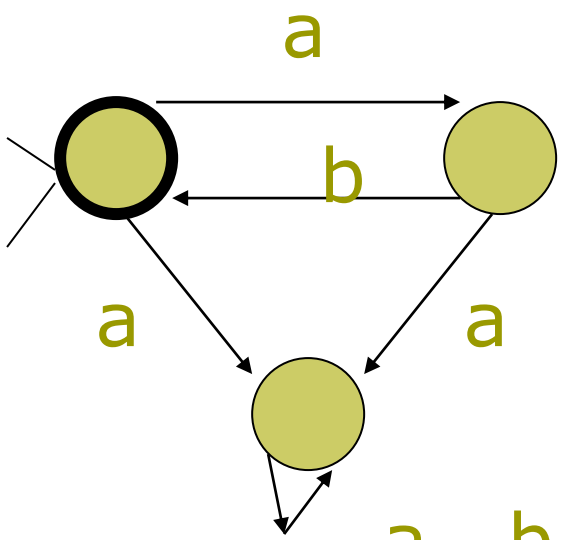
ab	+
a	-
b	-
abab	+
aba	-
ababab	+
b	-
bb	-
abababab	+
ababababab	+

(ab)*

$S \rightarrow a\ B$

$B \rightarrow bS$

$S \rightarrow \varepsilon$



a , b

Descripción de Patrones Usando Gramáticas

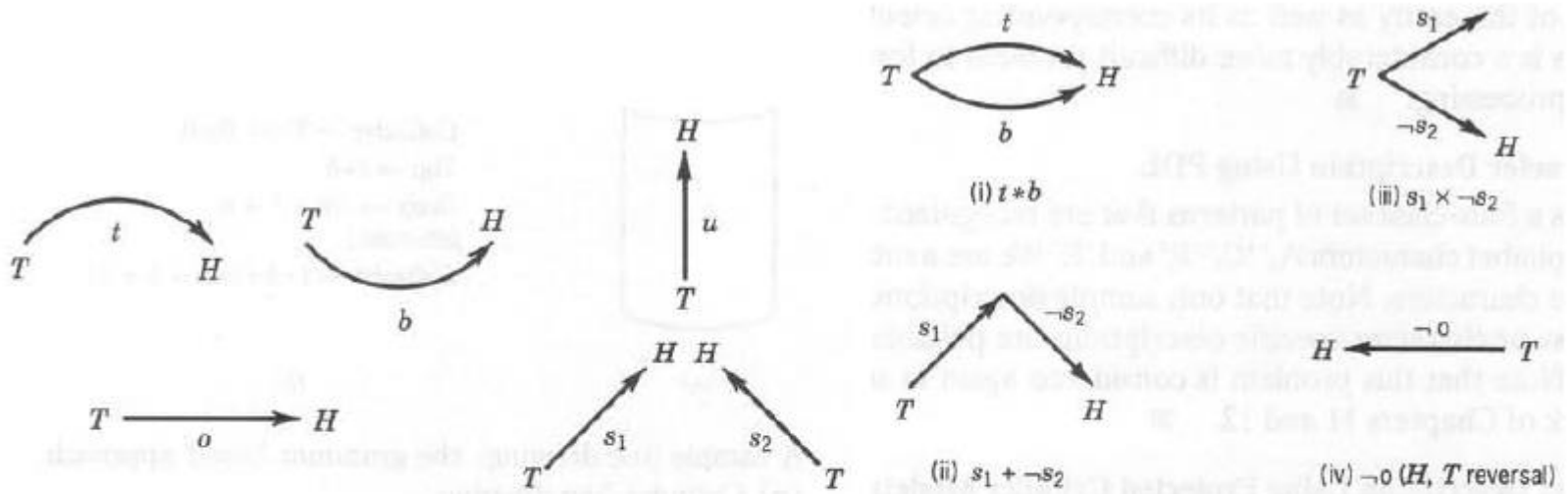


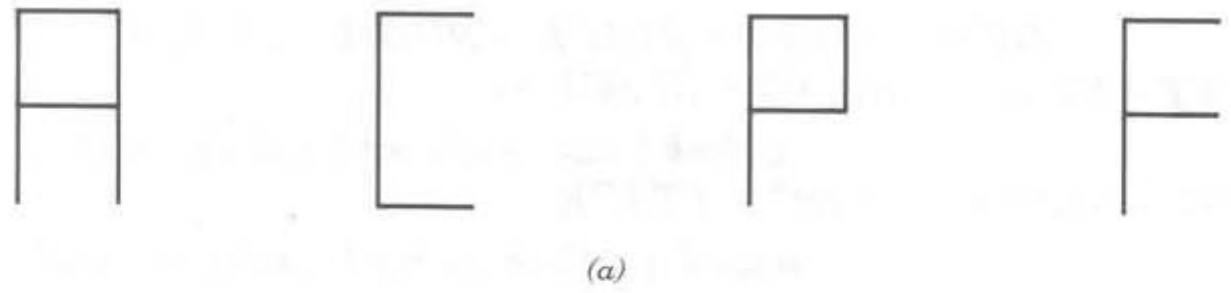
Figure 6: A 2-D line drawing picture description grammar with the set of terminal symbols $\{t, b, u, o, s, *, \neg, +\}$ where $+$ represents head to tail concatenation, $*$ represents head-head and tail-tail attachment, and $\neg$ represents head and tail reversal. H represents heads of lines and T represents the tails. (Schalkoff, Pattern Recognition: Statistical, Structural and Neural Approaches, 1992)

Descripción de Patrones Usando Gramáticas

$$Cylinder \rightarrow Top * Body$$
$$Top \rightarrow t * b$$
$$Body \rightarrow \neg u + b + u$$
(alternate:)
$$Cylinder \rightarrow t * b * (\neg u + b + u)$$

Figure 7: Representation of a cylinder using the line drawing picture description grammar.

Descripción de Patrones Usando Gramáticas



(a)

(b)

$A = u + ((u + 0 + \neg u)^{*}0) + \neg u$

$C = \neg 0 + u + u + 0$

$P = u + ((u + 0 + \neg u)^{*}0)$

$F = u + (0 \times u) + 0$

(c)

Figure 8: Representation of four characters using the line drawing picture description grammar. (a) Pattern data. (b) Primitive representation and interconnection. (c) Corresponding descriptions.

Descripción de Patrones Usando Gramáticas

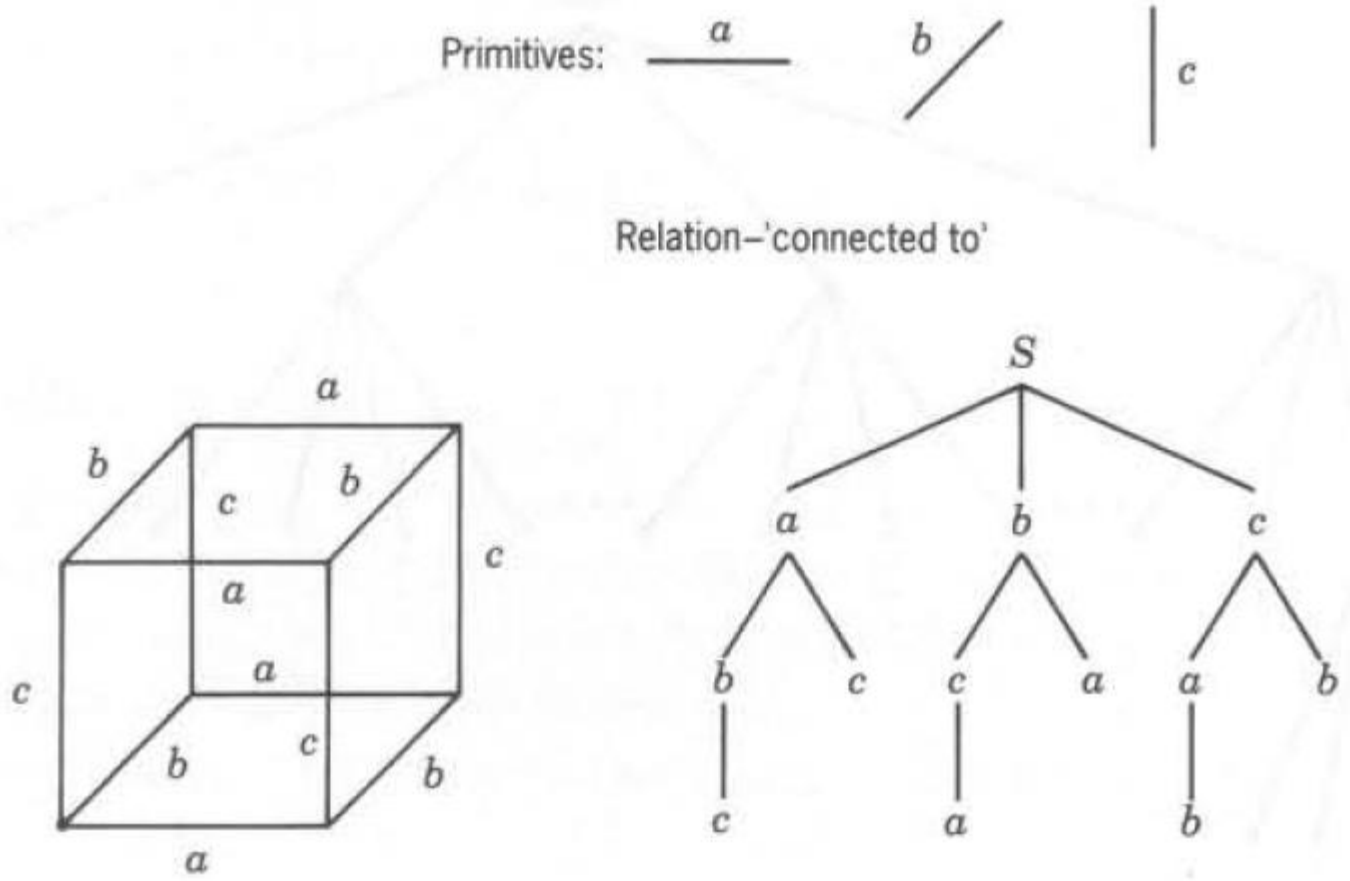


Figure 11: Tree grammar-based representation of a cube.

2.1 Conceptos Asociados

- Una gramática G es una cuadrupla, $G=(N,T,P,S)$, donde
  - N es un conjunto de símbolos no terminales
  - T es un conjunto de símbolos terminales
  - P es un conjunto de producciones de la forma
    $A \rightarrow a$, $A \rightarrow aB$, $A,B \in N$, $a \in T$
  - S es el símbolo de comienzo

2.1 Conceptos Asociados(2)

- Un Autómata Finito (AF) es un sextupla, M=(T,Z,Q,∂,w,q0):
  - T : alfabeto de entrada-salida
  - Z : alfabeto de salida
  - Q: un conjunto de estados
  - $\partial : T \times Q \rightarrow Q$ es una aplicación de transición de estados
  - W: Q$\rightarrow$ Z es una aplicación de salida
  - q0 $\in$ Q es el estado inicial

Ejemplo

$S \rightarrow a\ B$

$B \rightarrow bS$

$S \rightarrow \varepsilon$

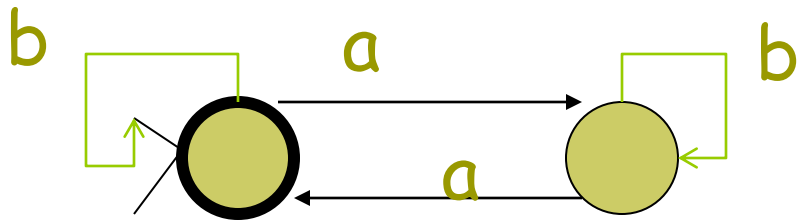
Ejemplo

$S \rightarrow a\ A$

$S \rightarrow bS$

$A \rightarrow b\ A$

$S \rightarrow \varepsilon$



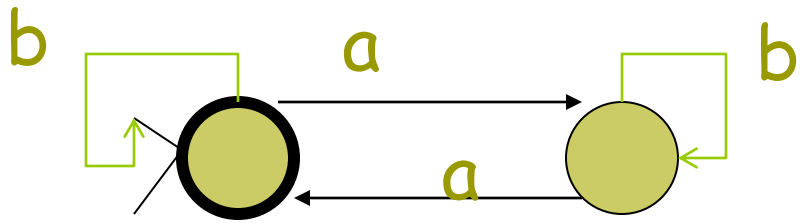
Ejemplo

$S \rightarrow a\ A$

$S \rightarrow bS$

$A \rightarrow b\ A$

$S \rightarrow \varepsilon$



Reconoce cualquier cadena con un número par de a's

2.1 Conceptos Asociados(3)

- Una gramática Difusa G es una cuádrupla, G=(N,T,P,S), donde
  - N es un conjunto de símbolos no terminales
  - T es un conjunto de símbolos terminales
  - P es un conjunto de producciones de la forma

$$A \xrightarrow{\theta} a,\ A \xrightarrow{\theta} aB,\ A,B \in N,\ a \in T,\ \theta \in (0,1]$$

  - S es el símbolo de comienzo

2.1 Conceptos Asociados(4)

- En el caso difuso una secuencia de símbolos pertenece a un lenguaje con un grado de pertenencia.
- Ejemplo: Sea G=(N,T,S,P), donde:
  - N={S,A,B}
  - T={a,b}
  - S es el símbolo de comienzo
  - P es el conjunto de producciones

$$P = \{S \xrightarrow{0.3} aS, S \xrightarrow{0.5} aA, S \xrightarrow{0.7} aB, S \xrightarrow{0.3} bS,$$

$$S \xrightarrow{0.2} bA, A \xrightarrow{0.5} b, B \xrightarrow{0.4} b\}$$

2.1 Conceptos Asociados(5)

- El grado de pertenencia $\mu(x)$ de una secuencia x perteneciente a un lenguaje regular difuso es el máximo de cualquier derivación de x, donde el valor de una derivación específica de x es igual al peso mínimo de las producciones usadas:

$$\mu(x) = \mu(S \overset{*}{\Rightarrow} x) = \max_{S \overset{*}{\Rightarrow} x}(\min(\mu(S \to \alpha_1), \mu(\alpha_1 \to \alpha_2), \ldots, \mu(\alpha_m \to x))$$

Conceptos Asociados (6)

- Sea G=(N,T,S,P), donde:
  - N={S,A,B}
  - T={a,b}
  - S es el símbolo de comienzo
  - P es el conjunto de producciones

$$P = \{S \overset{0.3}{\rightarrow} aS, S \overset{0.5}{\rightarrow} aA, S \overset{0.7}{\rightarrow} aB, S \overset{0.3}{\rightarrow} bS, S \overset{0.2}{\rightarrow} bA, A \overset{0.5}{\rightarrow} b, B \overset{0.4}{\rightarrow} b\}$$

El grado de pertenencia de ab es:

$$\mu(ab) = \mu(S \overset{*}{\Rightarrow} ab) = \max_{S \overset{*}{\Rightarrow} ab}(\min(S \overset{0.5}{\rightarrow} aA, A \overset{0.5}{\rightarrow} b),$$

$$\min(S \overset{0.7}{\rightarrow} aB, B \overset{0.4}{\rightarrow} b)) = \max_{S \overset{*}{\Rightarrow} ab}(0.5,0.4) = 0.5$$

Conceptos Asociados (7)

- Un Autómata de estados difuso (FFA), es un sextupla, M=(T,Z,Q, ∂,w,q0):
  - T : alfabeto de entrada-salida
  - Z : alfabeto de salida
  - Q: un conjunto de estados
  - $\partial : T \times Q \times (0,1] \rightarrow Q$ es una aplicación de transición de estados
  - W: Q $\rightarrow$ Z es una aplicación de salida
  - q0 $\in$ Q es el estado inicial difuso

Como en el caso crisp existe una equivalencia entre gramática-autómata

Conceptos Asociados(8)

- Consideremos la gramática:
- Sea G=(N,T,S,P), donde:
  - N={S,A,B}, T={a,b}, S es el símbolo de comienzo

$$P = \{S \xrightarrow{0.3} aS, S \xrightarrow{0.5} aA, S \xrightarrow{0.7} aB, S \xrightarrow{0.3} bS, S \xrightarrow{0.2} bA, A \xrightarrow{0.5} b, B \xrightarrow{0.4} b\}$$

Sea M=(T,Z,Q,∂,w,q0) tal que L(G)=L(M):

T={a,b}, Z={Final, No Final}, Q={q0,q1,q2,q3},

$$\partial(q0,a,0.5) = q1, \partial(q0,b,0.2) = q1, \partial(q0,a,0.3) = q0,$$
$$\partial(q0,b,0.3) = q0, \partial(q0,a,0.7) = q2, \partial(q1,b,0.5) = q3,$$
$$\partial(q2,b,0.4) = q3$$

Conceptos Asociados(9)

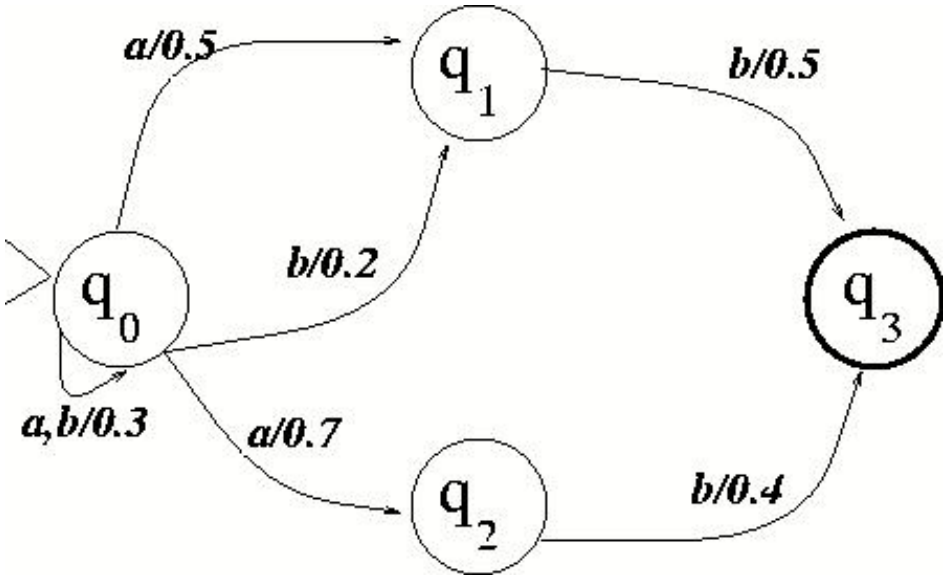
- Aplicaciones salida:

  w(q0)="No final", w(q1)="No final",

  w(q2)="No final", w(q3)="final"

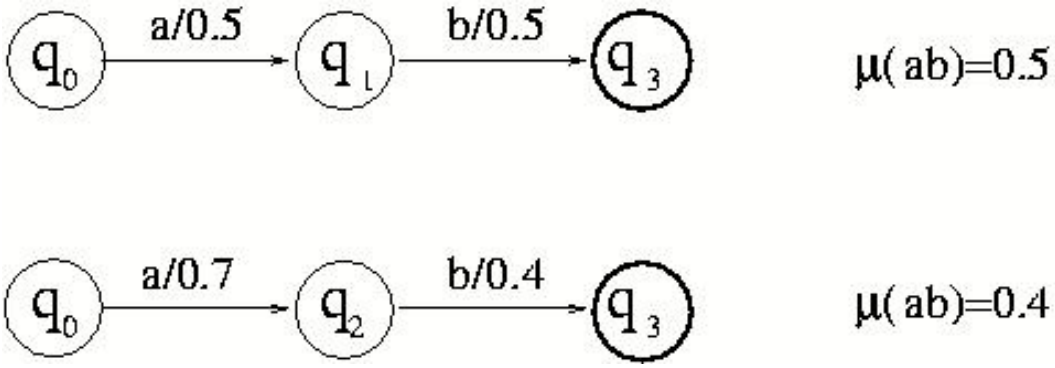
Los estados finales son representados por un círculo más grueso

$$P = \{S \xrightarrow{0.3} aS, S \xrightarrow{0.5} aA, S \xrightarrow{0.7} aB, S \xrightarrow{0.3} bS, S \xrightarrow{0.2} bA, A \xrightarrow{0.5} b, B \xrightarrow{0.4} b\}$$



Conceptos Asociados(10)

- El grado de pertenencia de una secuencia x calculado por FFA es el máximo grado de todos los caminos que aceptan la secuencia.
  - El grado de pertencia de "ab" es 0.5

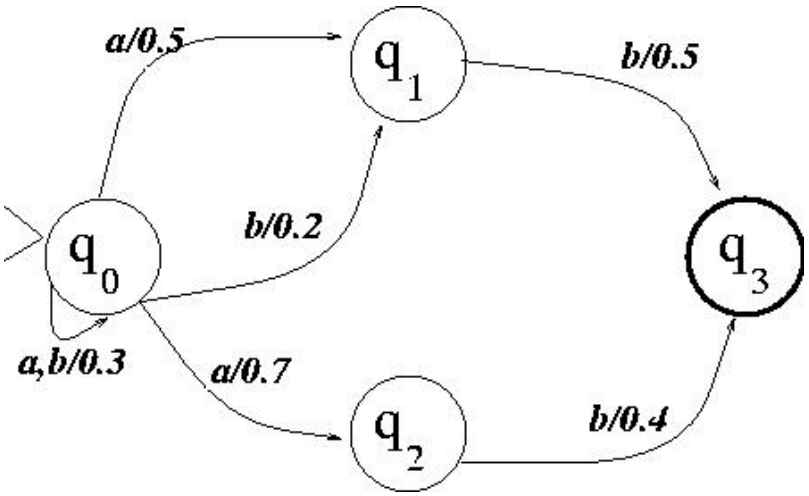


Conceptos Asociados(11)

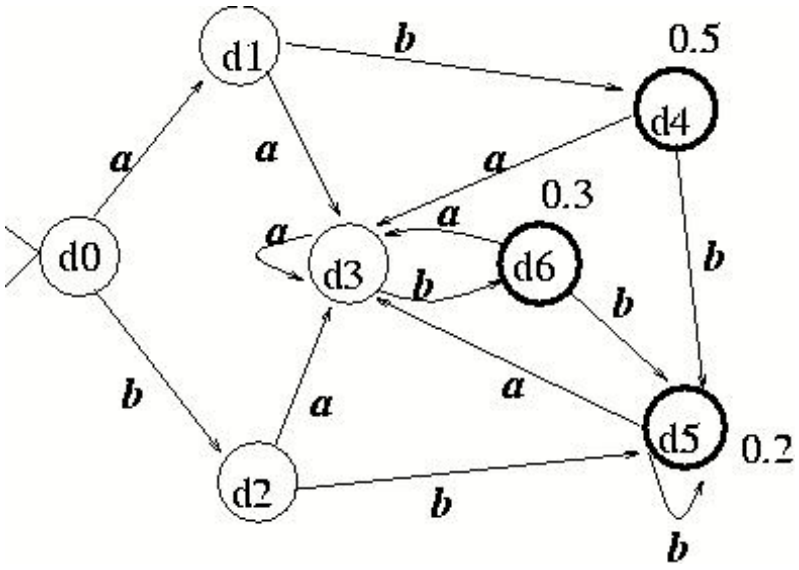
- TEOREMA(Thomason and Marinos, 1974).Dado un FFA, M, existe un DFA A con alfabeto de salida $Z \subseteq \{\theta: \theta$ es grado de pertenencia$\} \cup \{0\}$ que calcula las funciones de pertenencia : $\mu: T^* \rightarrow [0,1]$ del lenguaje L(M) en la salida de sus estados (aplicación de salida w)

Ejemplo

FFA



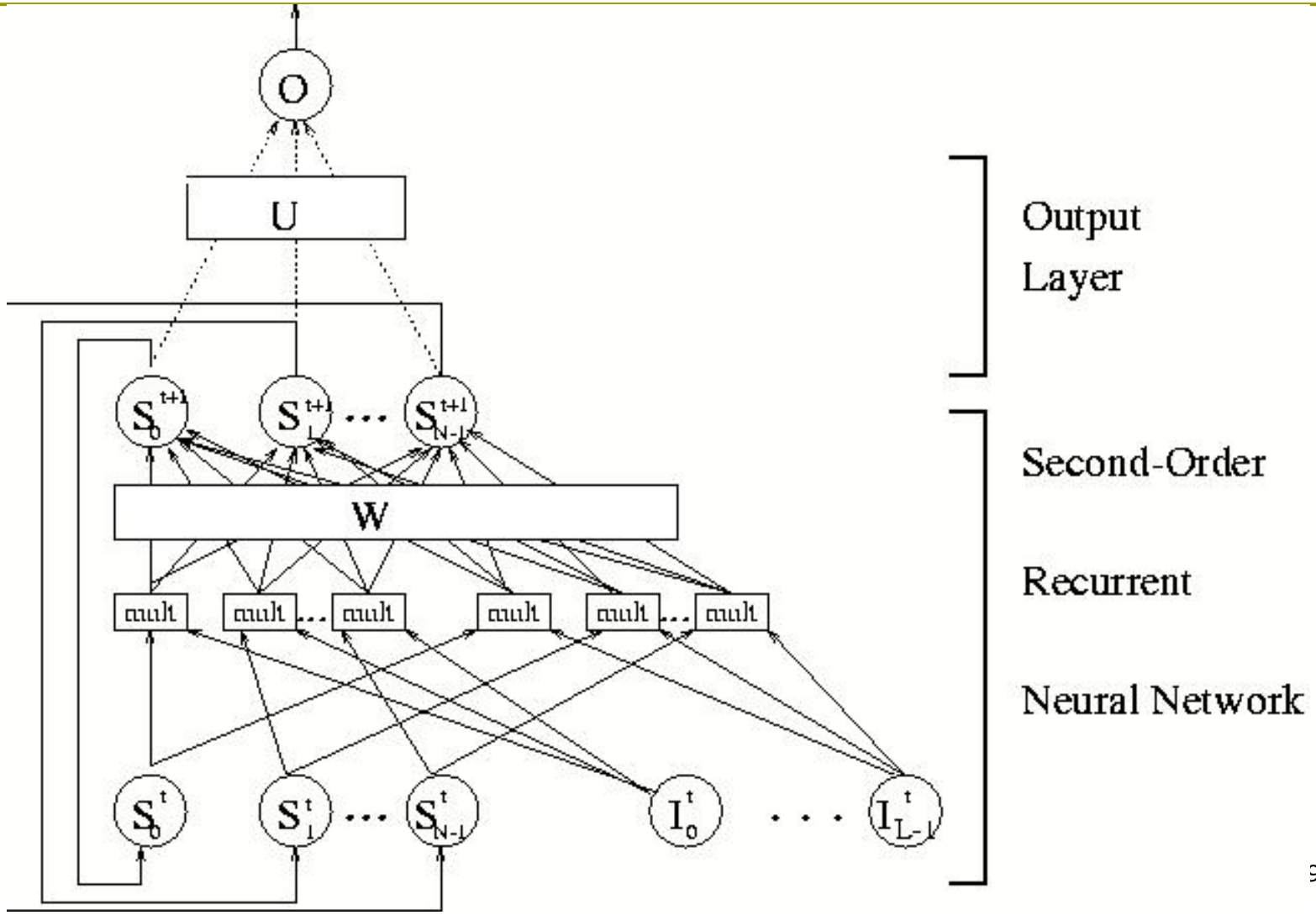
DFA



Aplicaciones FFA

- Detectar y cuantificar lesiones en las arterias a partir de arteriogramas.
- Diseño de interfaces inteligentes.
- Monitorización clínica
- Análisis léxico
- Reconocimiento sintáctico del envejecimiento del esqueleto a partir de Rayos-X.

3 Arquitecturas para realizar IG Difusa.
3.1 Modelo de única salida



4.1 Dinámica de la Red

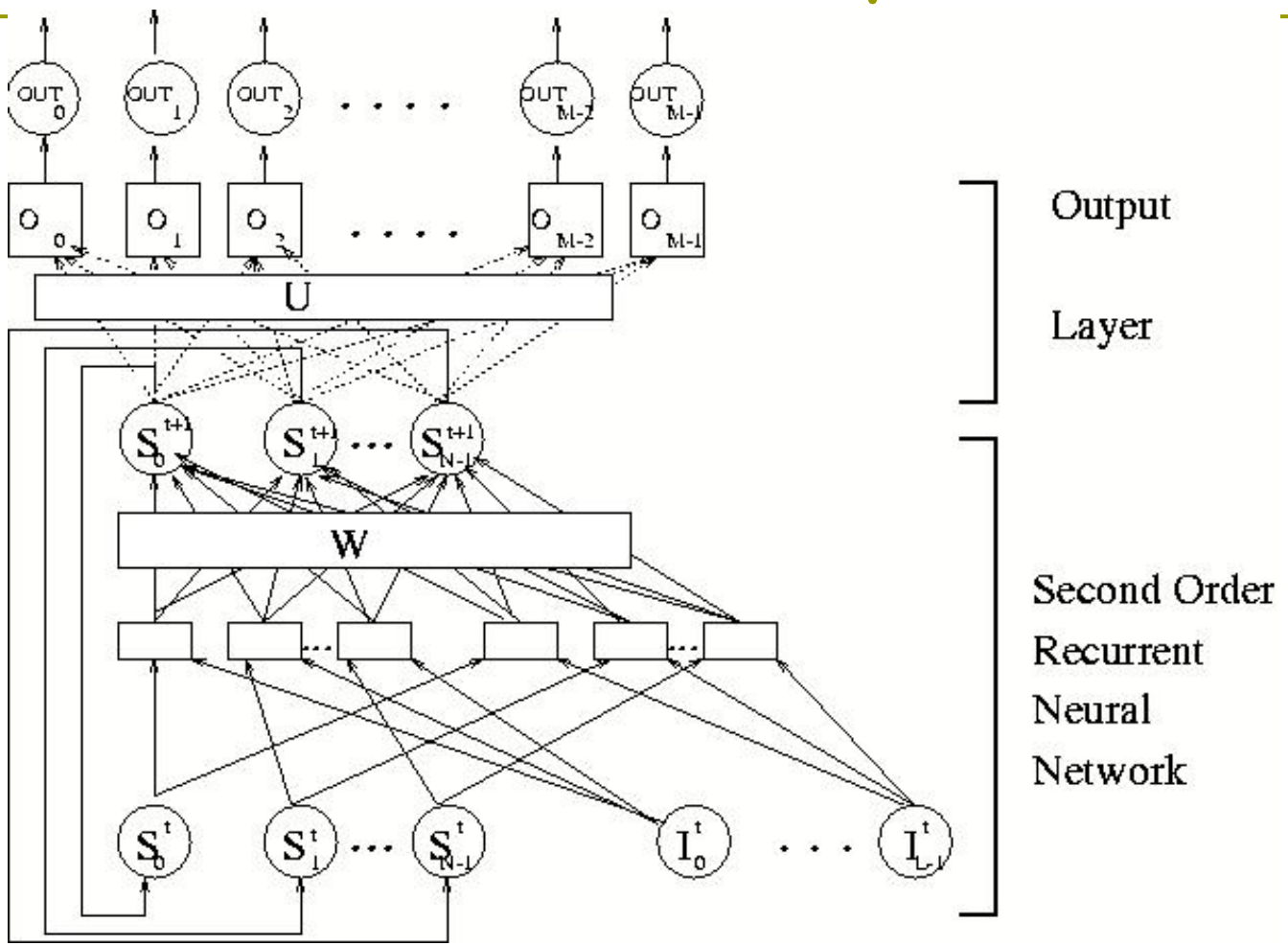
$$S_i^{t+1} = g(\theta_i^t)$$

$$\theta_i^t = \sum_{j=0}^{N-1} \sum_{k=0}^{L-1} I_k^t S_j^t w_{ijk}$$

$$g(x) = \frac{1}{1+e^{(-x)}}$$

$$O = \sum_{i=0}^{N-1} u_i S_i^m$$

3.2 Modelo de Salida Múltiple



3.3 Dinámica de la Red

$$S_i^{t+1} = g(\theta_i^t)$$

$$\theta_i^t = \sum_{j=0}^{N-1} \sum_{k=0}^{L-1} I_k^t S_j^t w_{ijk}$$

$$g(x) = \frac{1}{1+e^{(-x)}}$$

$$O_p = g(\sigma_p)$$

$$\sigma_p = \sum_{i=0}^{N-1} u_{pi} S_i^p$$

$$OUT_P = \begin{cases} 1 & O_p = \max\{O_0, O_1, ..., O_{M-1}\} \\ 0 & \text{en otro caso} \end{cases}$$

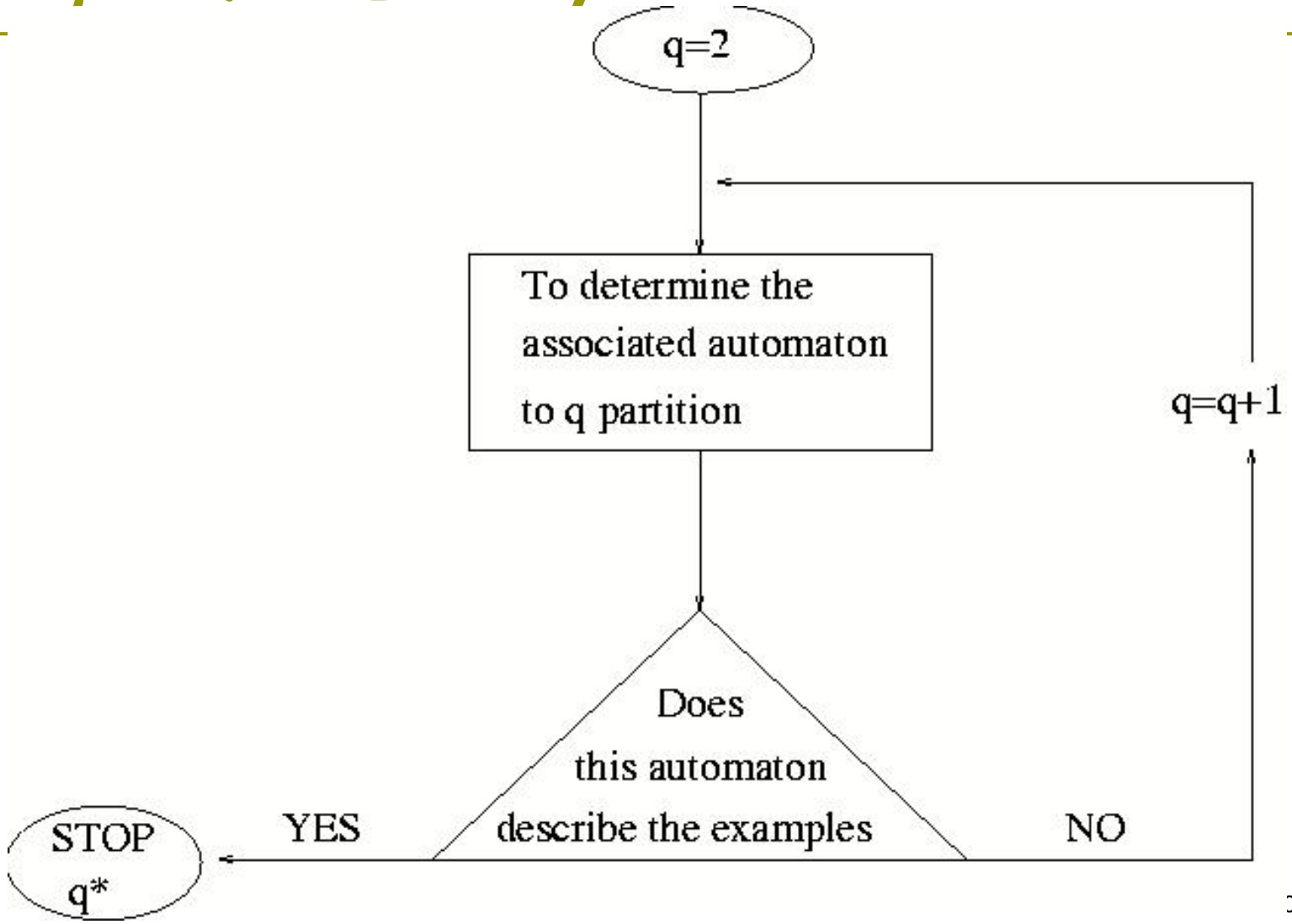
$\mu(x) = i * r$, donde r es la precisión deseada

4. Extracción del FFA

4.1 Particionar el espacio de salida

- Construir una secuencia de particiones del espacio en la que cada una va a representar a un estado del autómata.
- Si la red tiene N neuronas ocultas , cualquier valor de estado será un pto de $[0,1]^N$
- Dividimos $[0,1]^N$ en q subintervalos, obteniendo exactamente q^N hipercubos.
- La idea consiste en encontrar algún óptimo q* asociado a los estados del autómata.

Diagrama Ensayo-error



Construcción del q-autómata

- El algoritmo se basa en construir un árbol de búsqueda con la estrategia de primero en anchura:
  - Los nodos representan los estados del q-autómata
  - Los enlaces son las transiciones entre los estados

Construcción del q-autómata(2)

1. La raíz representa el estado inicial. Es etiquetada por el hipercubo que contiene el pto $S_0^0 = 1,\ S_i^0 = 0\ \ \forall i \neq 0$ y el grado pertenencia devuelto por la salida.
2. Comenzando en el nodo actual Bi, para cada símbolo s ∈{a0,aL-1}:
   a) Introducirlo en la red
   b) Obtener los valores de S_i , definen un pto de un hipercubo Bj
   c) La salida será el grado de pertenencia de Bj.

En estas circunstancias puede ocurrir:

Construcción del q-autómata(3)

- Bj y su grado de pertenencia se ha visitado antes: el árbol es podado en este nodo.

$$\partial(B_i, s) = B_j$$

- Si Bj=Bi , se crea un ciclo en el árbol $\partial(B_i, s) = B_i$ y el árbol se poda en este nodo.

- Si Bj junto con el grado de pertenencia no ha sido visitado antes, creamos un nuevo estado Bj y la correspondiente transición

$$\partial(B_i, s) = B_j$$

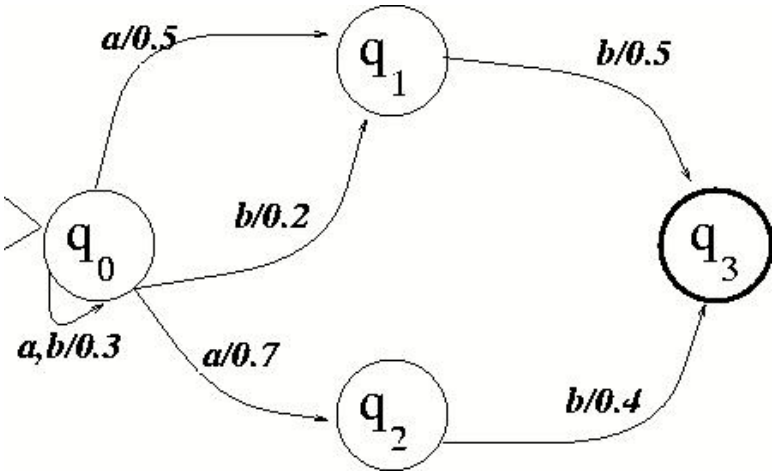
Construcción del q-autómata(4)

3. Repetir el paso 2 hasta que el conjunto de símbolos sea procesado.
4. Si nuevos nodos se han creado cuyos hipercubos no han sido visitados antes volver al paso 2. En otro caso el procedimiento finaliza.

llegado este pto tenemos que chequear el conjunto de ejemplos, si los reconoce todos hemos terminado, en caso contrario q=q+1 y volvemos a empezar.

EJEMPLO

1. Consideremos el autómata:



2. Construcción ejemplos:
   - 200 secuencias aleatorias {a,b}
   - Procesar las secuencias con el autómata para asociarles un grado de pertenencia

3) Entrenamiento red neuronal:

- 2 neuronas de entrada, L={a,b}
- 2 neuronas recurrentes ocultas
- 11 neuronas en la capa de salida (precisión decimal)
- Razón de aprendizaje 0.1
- Tolerancia del error 0.000000125

Extracción autómata : q*=7

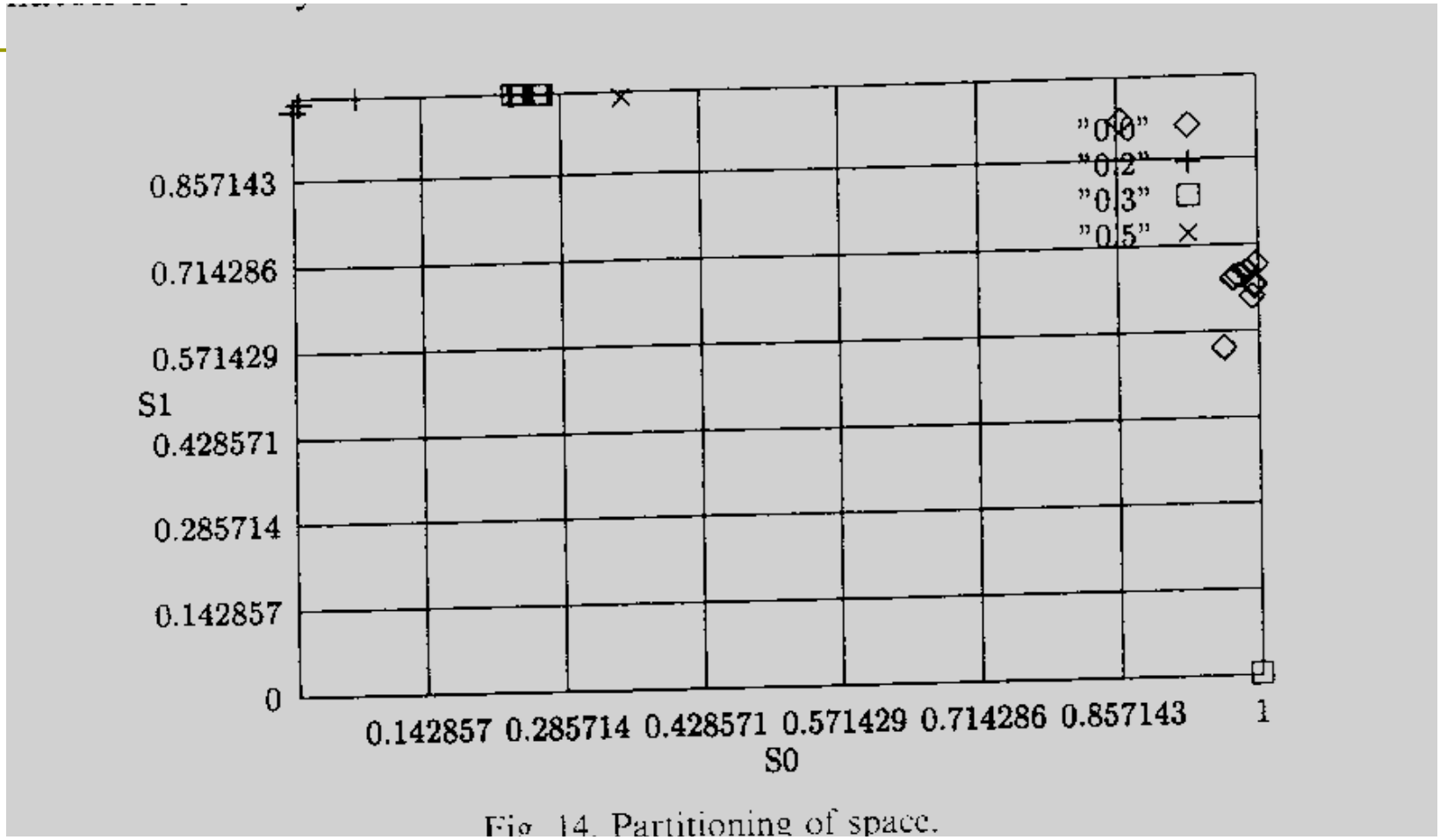
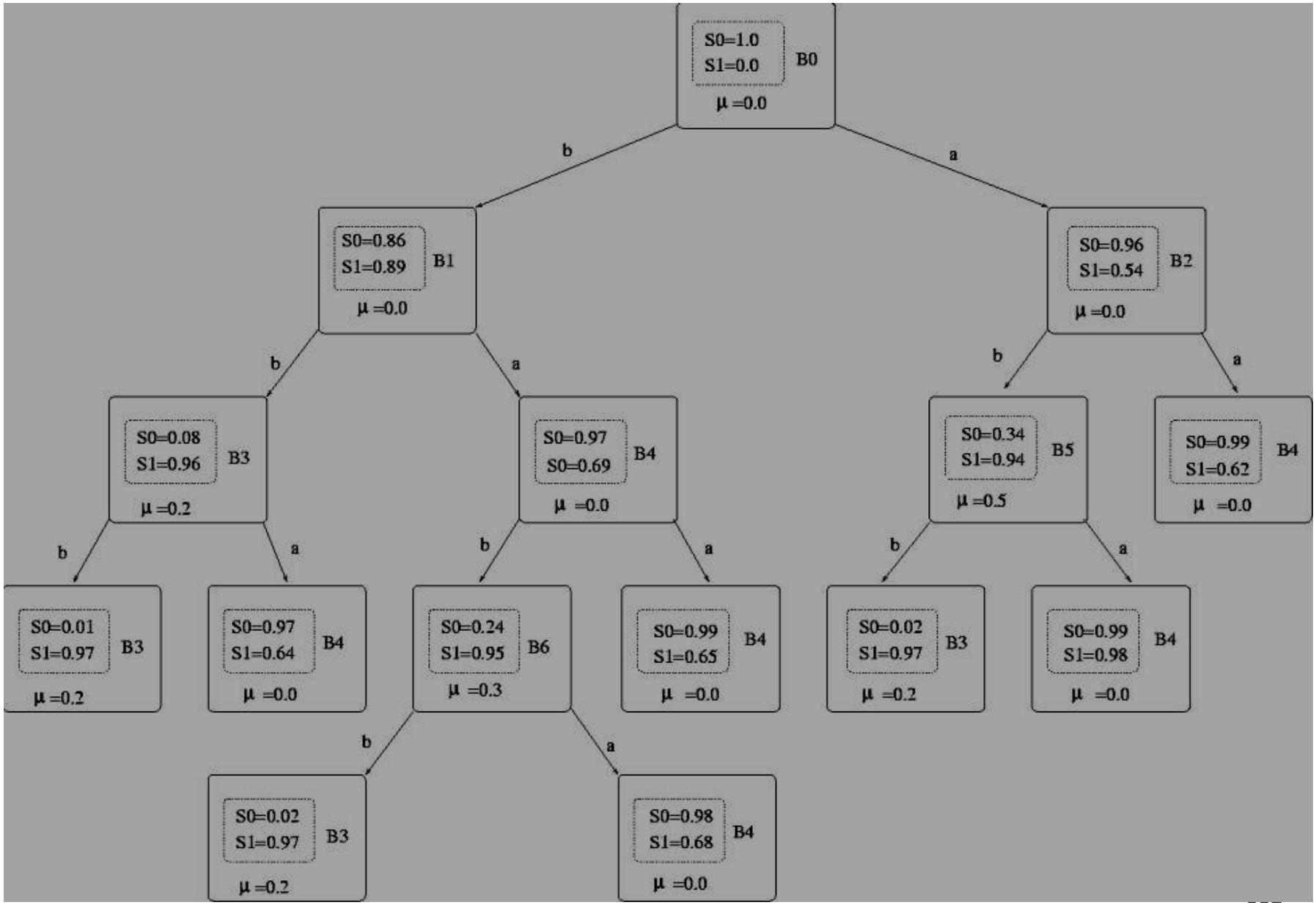
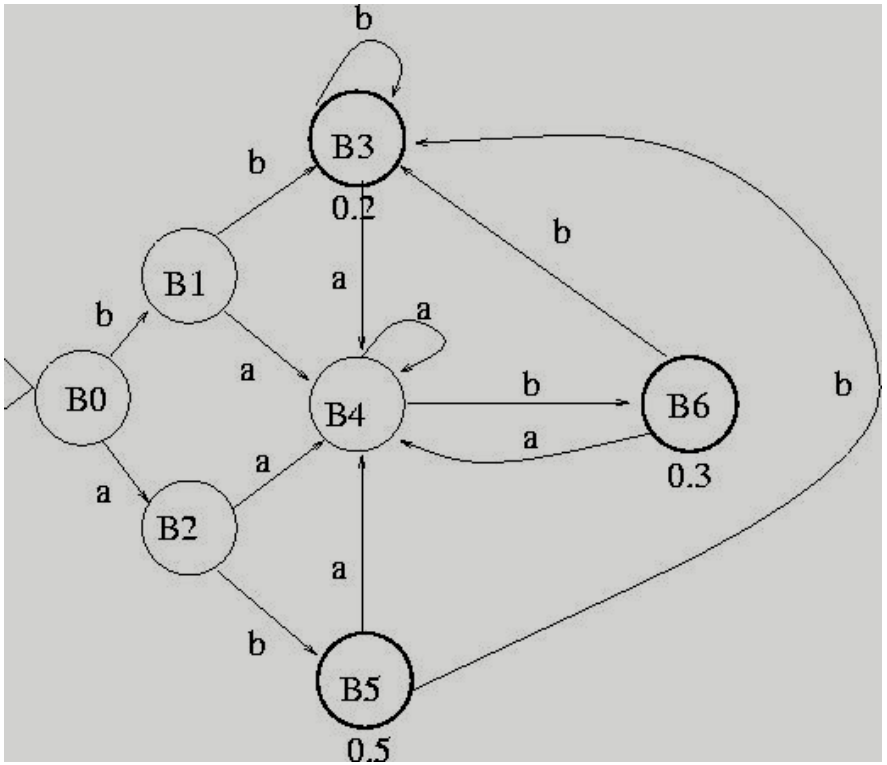


Fig. 14. Partitioning of space.


S0=1.0
S1=0.0
B0
μ =0.0
b
a
S0=0.86
S1=0.89
B1
μ =0.0
S0=0.96
S1=0.54
B2
μ =0.0
b
a
b
a
S0=0.08
S1=0.96
B3
μ =0.2
S0=0.97
S0=0.69
B4
μ =0.0
S0=0.34
S1=0.94
B5
μ =0.5
S0=0.99
S1=0.62
B4
μ =0.0
b
a
b
a
b
a
S0=0.01
S1=0.97
B3
μ =0.2
S0=0.97
S1=0.64
B4
μ =0.0
S0=0.24
S1=0.95
B6
μ =0.3
S0=0.99
S1=0.65
B4
μ =0.0
S0=0.02
S1=0.97
B3
μ =0.2
S0=0.99
S1=0.98
B4
μ =0.0
b
a
S0=0.02
S1=0.97
B3
μ =0.2
S0=0.98
S1=0.68
B4
μ =0.0

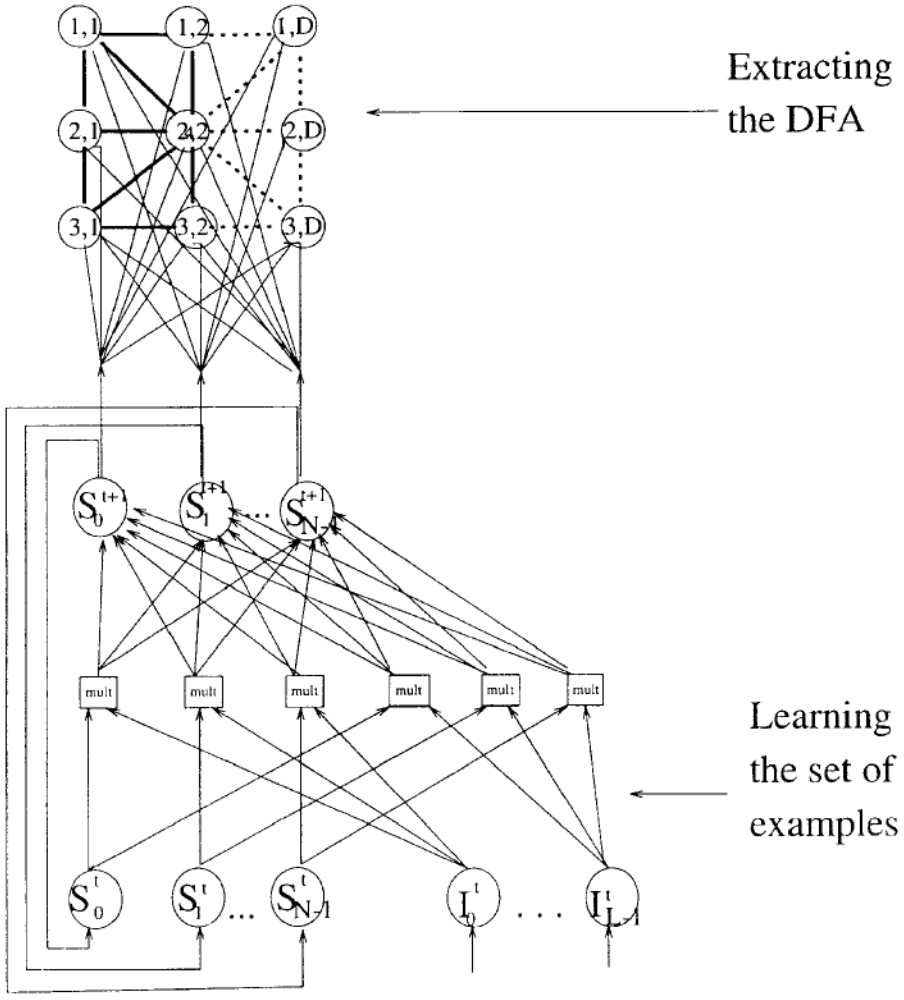
- Autómata extraido:



Extracción del AFD utilizando un SOM

1. Entrenar un SOM
2. Etiquetar los nodos del SOM con final/no final
3. Extraer el autómata

Extracción del AFD utilizando un SOM



Bibliografía

- Blanco, Delgado and Pegalajar. Extracting Rules from a (fuzzy/crisp) recurrent neural networks using a self-organizing map, International Journal of Intelligent Systems 15 (7) (2000) 595-621.
- Blanco, Delgado and Pegalajar, A real-coded genetic algorithm for training recurrent neural networks, Neural Networks, 14,2001,93-105.
- Giles, Omlin,Thornber, Equivalence in knowledge representation: automata, recurrent neural networks, and dynamical fuzzy systems in: Proceedings of the IEEE, 1999.

Bibliografía(2)

- Friedrich, S., Klaus P.A., "Clinical Monitoring with fuzzy automata. Fuzzy Sets and Systems, 61 (1994) 37-42.
- Lalande, Jaulent, "A fuzzy automaton to detect and quantify fuzzy artery lesions from arteriogrmas, in:Proceedings of the Sixh International Conference IPMU'96 , vol. 3, 1996, pp. 1481-1487.

Clasificación de ADN mediante Redes Neuronales Recurrentes

M.Carmen Pegalajar
Dpto. Ciencias de Comput. E I.A.
E-mail:mcarmen@decsai.ugr.es

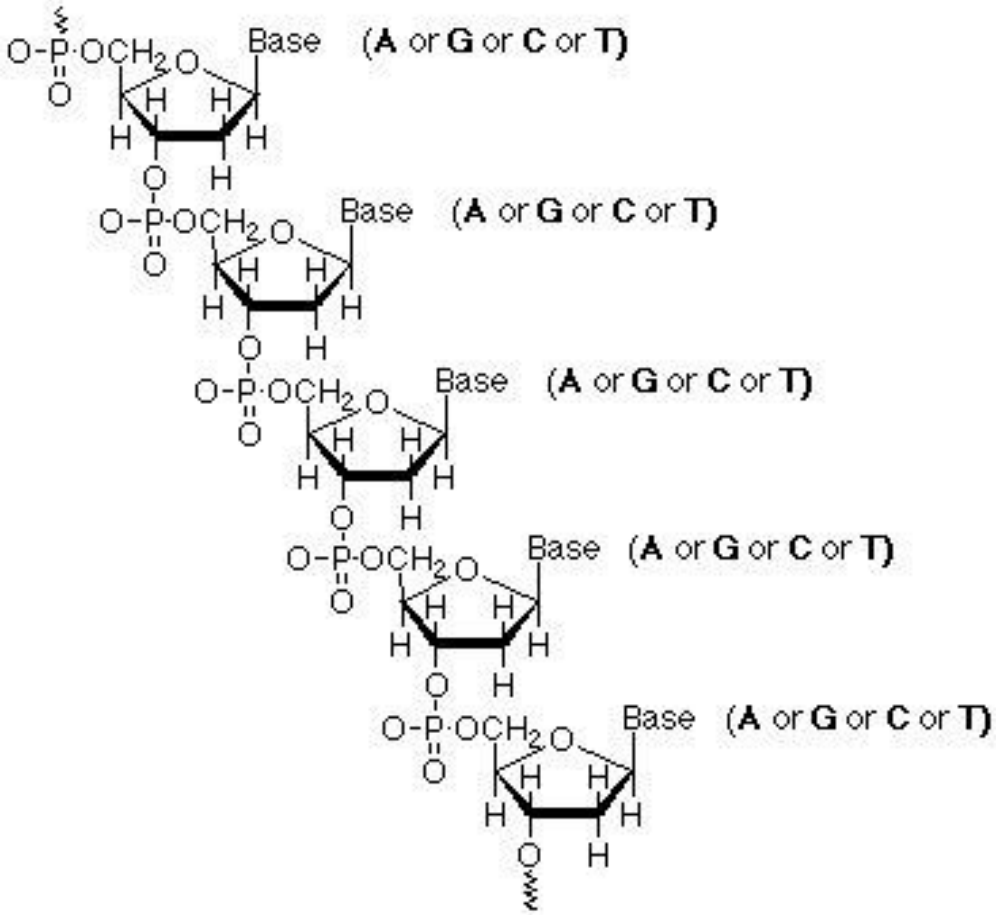
Clasificación de ADN mediante Redes Neuronales Recurrentes

1. Conceptos relacionados con ADN
2. Clasificación de ADN mediante RNR
3. Bibliografía

1.Conceptos relacionados con ADN

- Acido desoxirribonucleico
- Formado por:
  - azúcar (2- desoxi-D-ribosa)
  - ácido fosfórico
  - 4 bases nitrogenadas:
    - Adenina (A)
    - Guanina (G)
    - Citosina (C)
    - Timina (T)

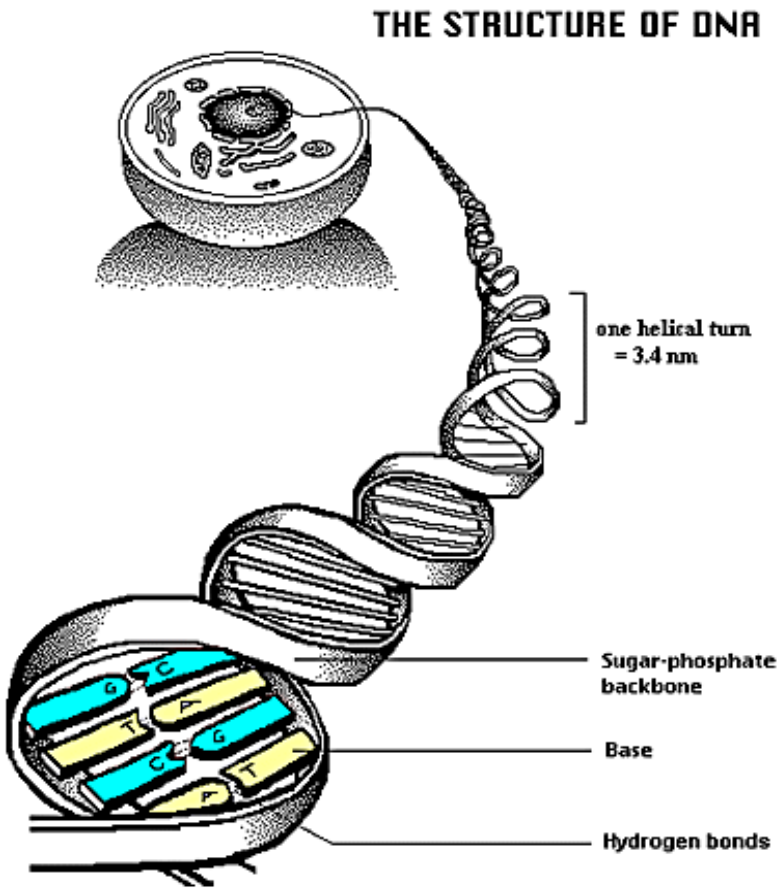
1.1 Estructura del ADN



1.1 Estructura del ADN

- Doble hélice
- Bases complementarias unidas por puentes hidrógenos:
  - A – T
  - C – G

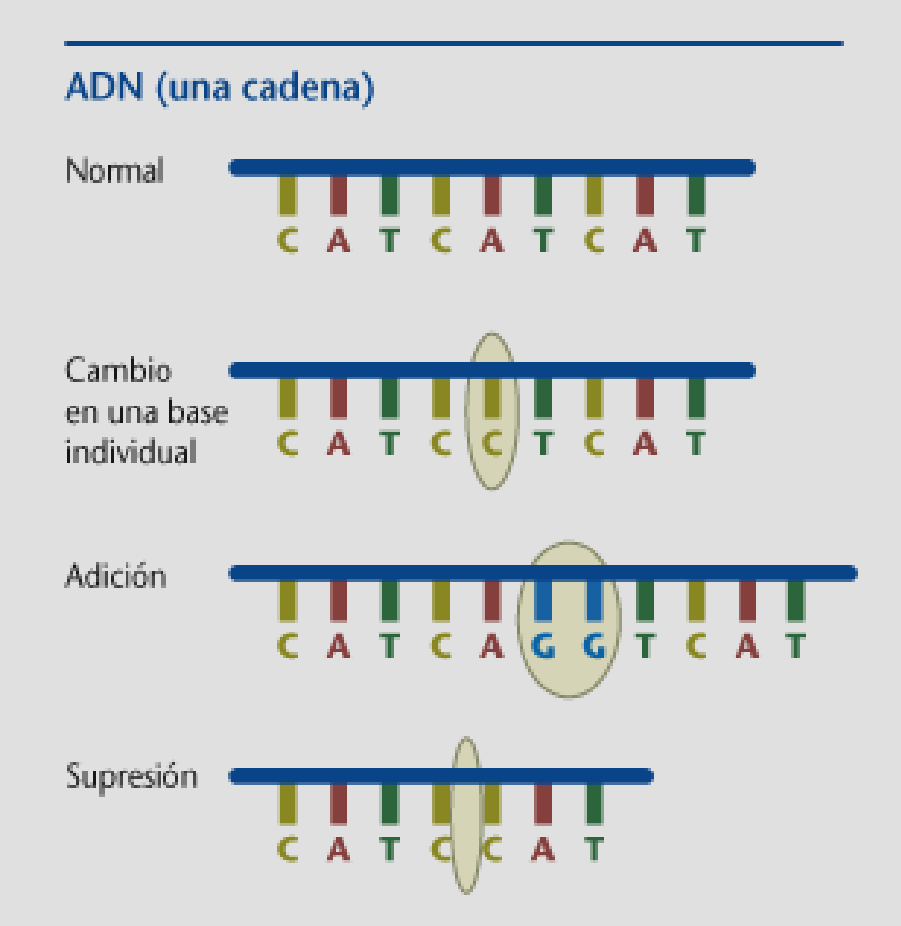
1.1 Estructura del ADN



1.2 ADN (mutaciones)

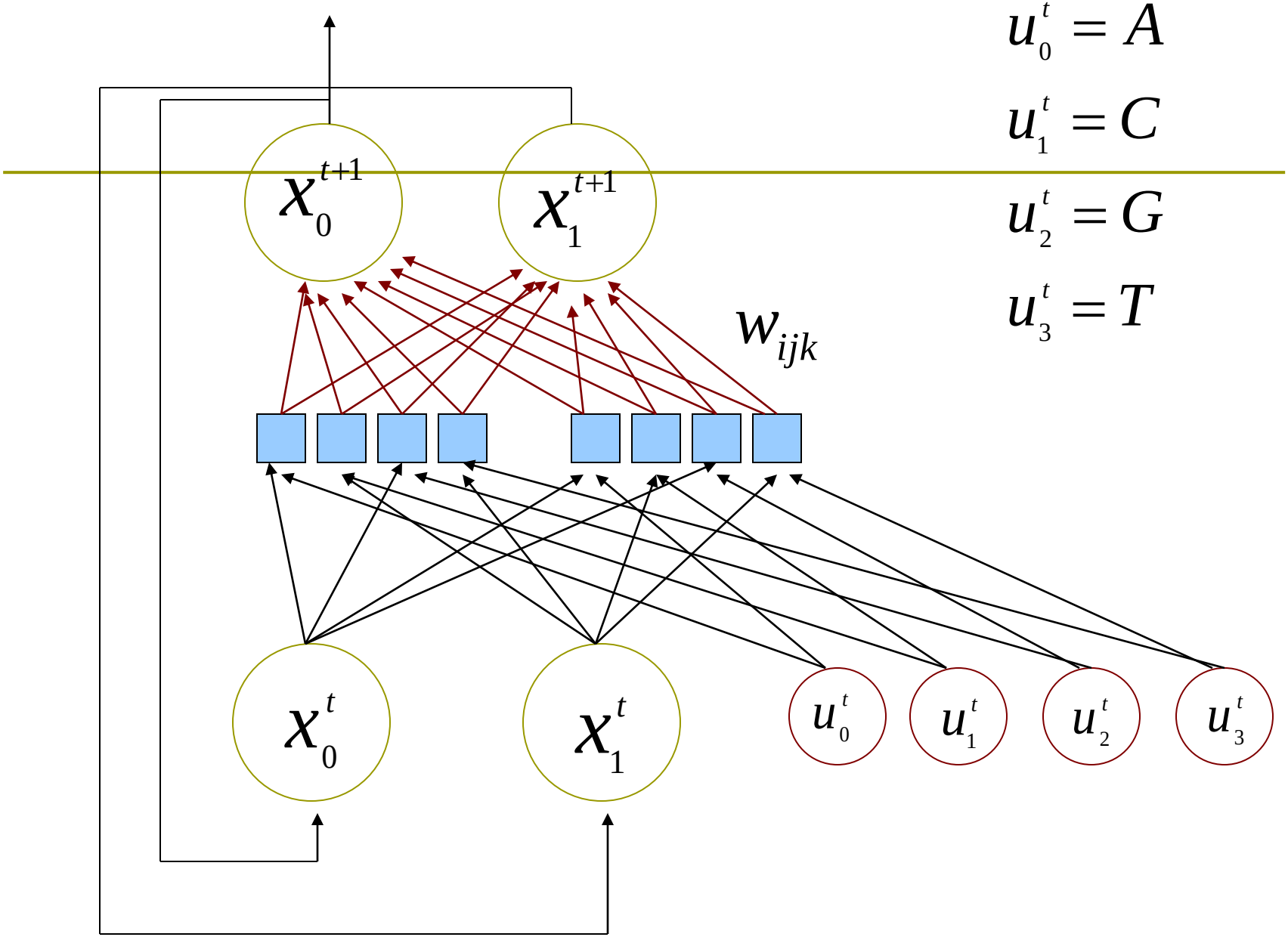
- Aunque la replicación del ADN es muy precisa, no es perfecta.
- El material genético puede sufrir alteración cualitativa o cuantitativa, y redistribución:
  - La forma más sencilla de mutación implica un cambio en una base individual a lo largo de la secuencia de bases ordenada de un gen en particular.
  - Agregar una o más bases.
  - Eliminar una o mas bases.
  - Algunas veces, grandes segmentos de una molécula de ADN se repiten, se eliminan o se translocan accidentalmente.

1.2 ADN (mutaciones)



2. Clasificación de ADN mediante RNRD

- Las secuencias de ADN no tienen longitud fija
- Limitaciones de las Redes Feedfordward:
  - Poco poder de representación de patrones dependientes de tiempo o del contexto
  - Limitadas a ejemplos de longitud fija
- Las Redes Recurrentes Dinámicas no tienen estas limitaciones, contemplaran una base nitrogenada por unidad de tiempo

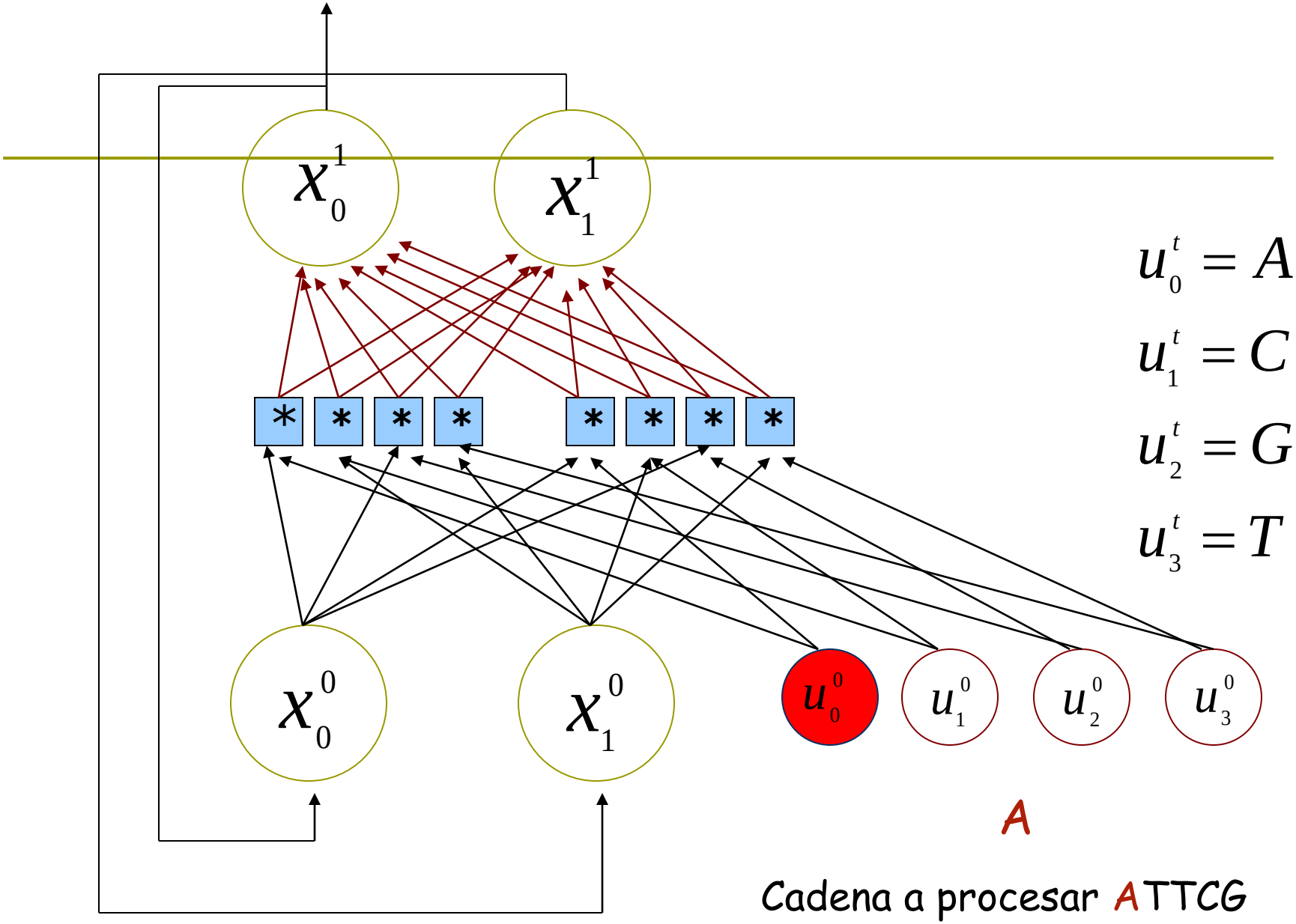

$u_0^t = A$
$u_1^t = C$
$u_2^t = G$
$u_3^t = T$
x_0^{t+1}
x_1^{t+1}
w_{ijk}
x_0^t
x_1^t
u_0^t
u_1^t
u_2^t
u_3^t

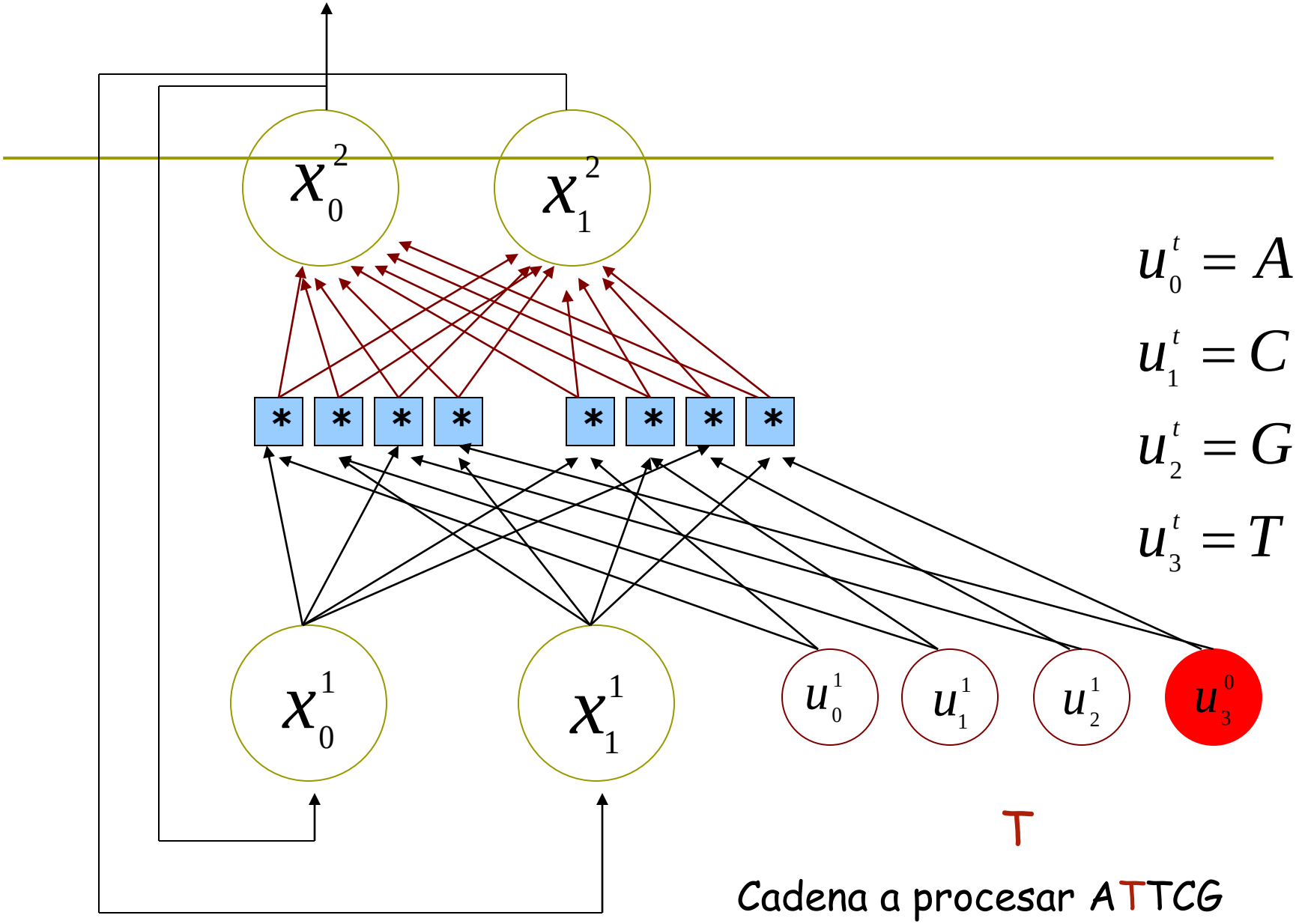
Supongamos que queremos procesar la secuencia ATTCG

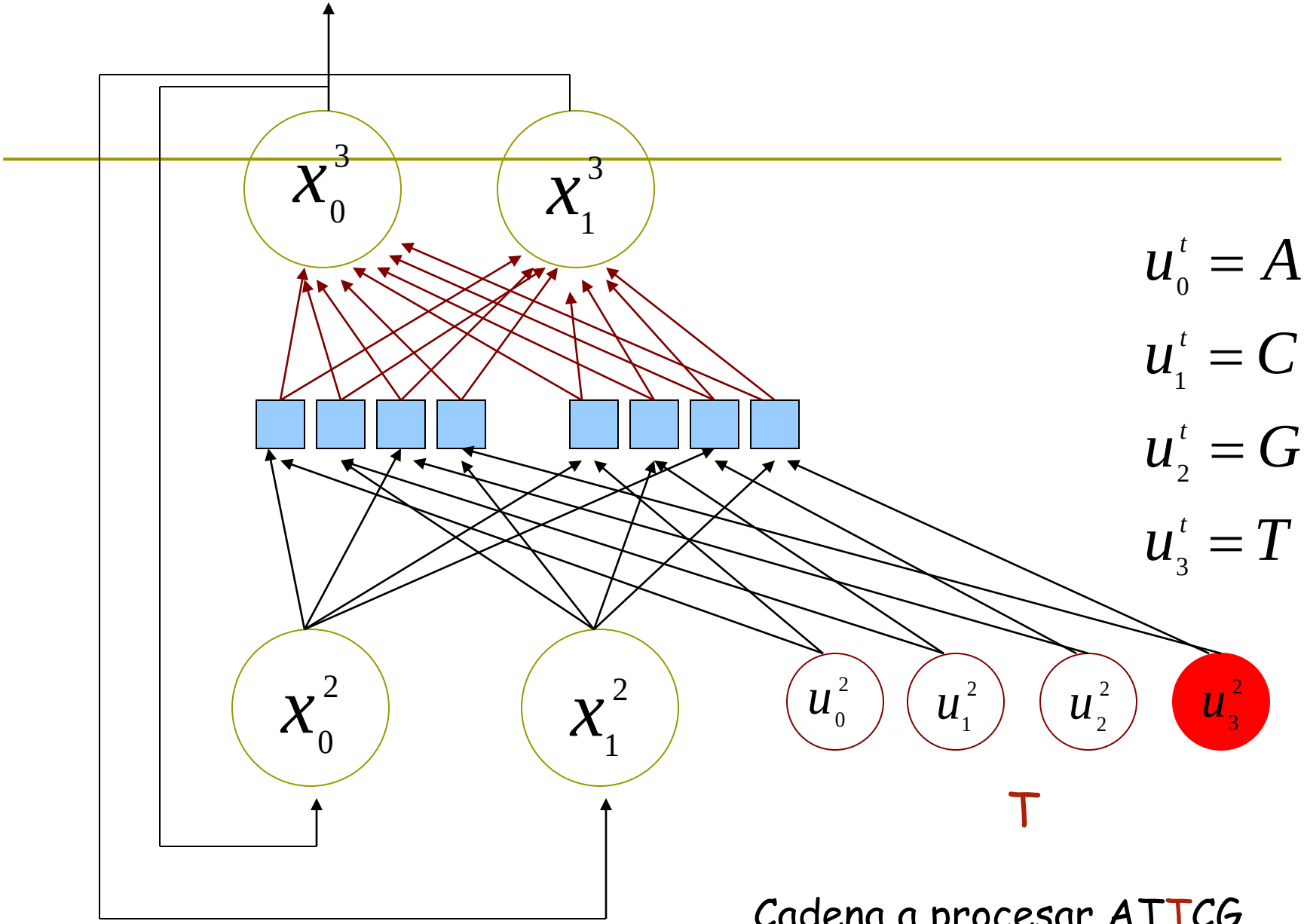
- En t=0 la red recibe el carácter A:
  - Lo codifica en la capa de entrada y
  - Lo procesa por las neuronas recurrentes, estas toman un valor determinado en t=1

- En t=1 la red recibe el carácter T
  -Lo codifica en la capa de entrada y
  -Lo procesa por las neuronas recurrentes que toman un valor en t=2

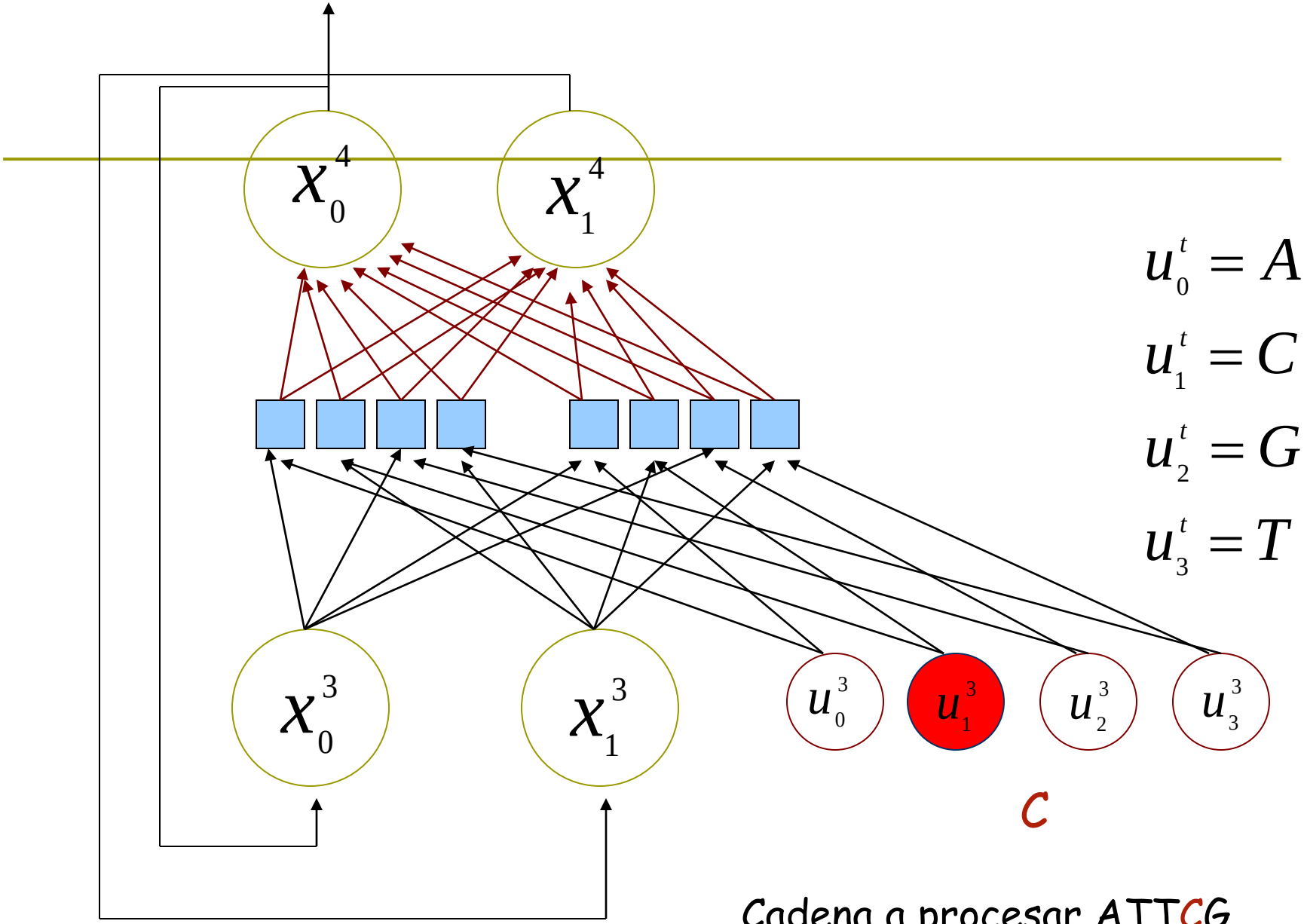
- En t=2 la red recibe el carácter T
  -Lo codifica en la capa de entrada y
  -Lo procesa por las neuronas recurrentes que toman un valor en t=3, y así sucesivamente...

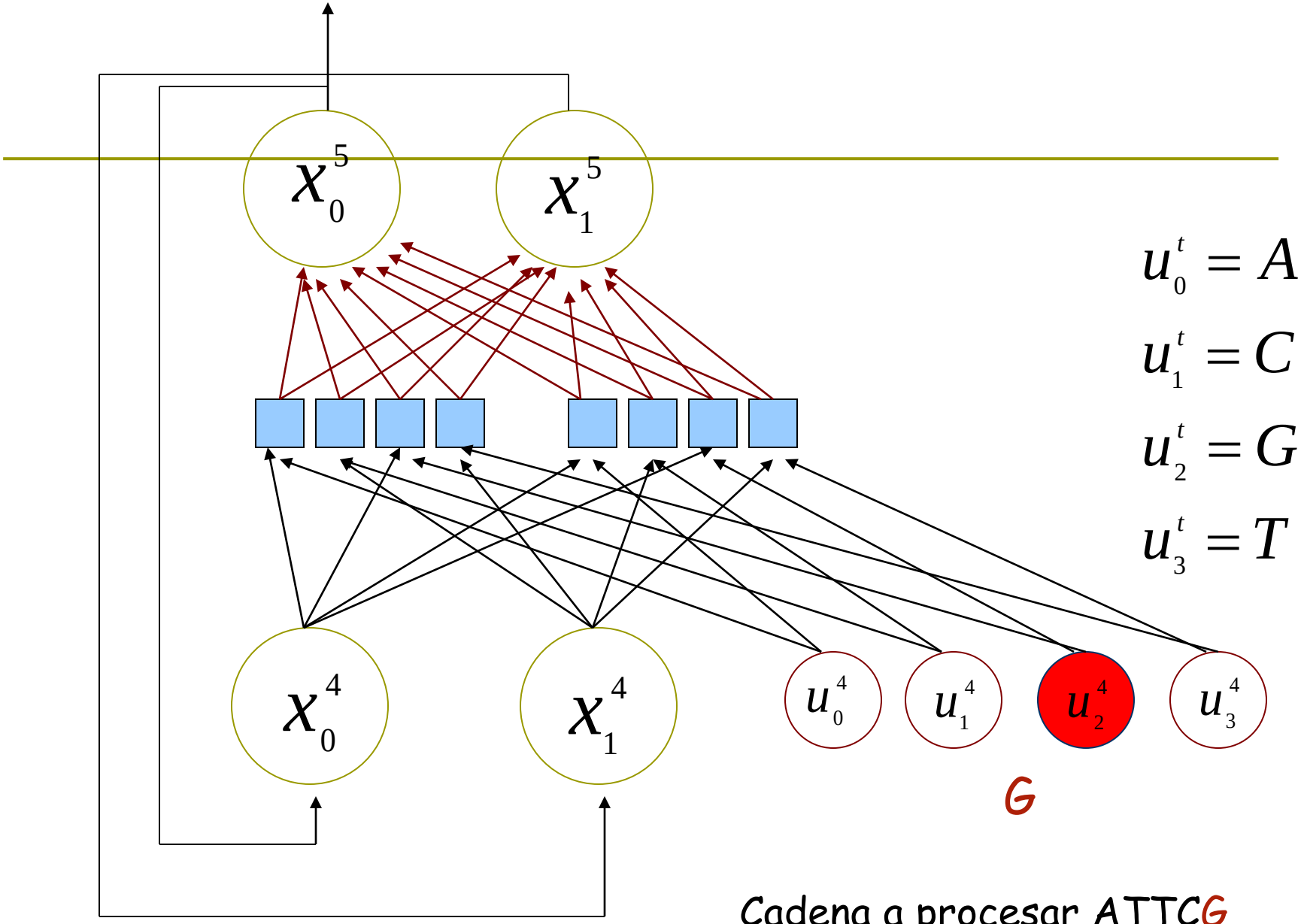

x_0^1
x_1^1
x_0^0
x_1^0
u_0^0
u_1^0
u_2^0
u_3^0
$u_0^t = A$
$u_1^t = C$
$u_2^t = G$
$u_3^t = T$
A
Cadena a procesar ATTCG


x_0^2
x_1^2
x_0^1
x_1^1
u_0^1
u_1^1
u_2^1
u_3^0
$u_0^t = A$
$u_1^t = C$
$u_2^t = G$
$u_3^t = T$
T
Cadena a procesar ATTCG



Cadena a procesar ATTCG


x_0^4
x_1^4
$u_0^t = A$
$u_1^t = C$
$u_2^t = G$
$u_3^t = T$
x_0^3
x_1^3
u_0^3
u_1^3
u_2^3
u_3^3
C
Cadena a procesar ATTCG



Cadena a procesar ATTCG

Clasificación de una cadena de ADN

- Si X_0^f es igual a 1 ó está próximo, la secuencia será clasificada como ADN.
- Si X_0^f es igual a 0 o está próximo, la secuencia será clasificada como no ADN

- Si el error cometido es pequeño se considerará que ha habido un acierto:

$$(d - x_0^f) * (d - x_0^f) < \varepsilon^2$$

f: fin de secuencia (la secuencia de longitud (f-1) ha sido procesada por la red)

d: valor deseado, d=0 ejemplo negativo, d=1 ejemplo positivo.

ε :tolerancia del error

5. Bibliografía

- Montana and Davis, "Training Feedforward neural networks using genetics algorithms, in Proceedings of Eleventh International Joint Conference on Artificial Intelligence, 1989, pp. 762-767.
- "Neural Networks. A comprehensive Foundation", Simon Haykin, Prentice Hall , Second Edition.
- Omlin, Giles, Thornber, "Equivalence in knowledge representation:automata, Recurrent Neural Networks Design and Applications, The CRC Press International Series on Computational Intelligence, 2000,pp. 99-131
- Speech Recognition using Fuzzy Second-Order Recurrent Neural Networks, IWANN 2001, Blanco A., Delgado M., PegalajarM.C., Requena I.
- Pegalajar et al.A Real-Coded Genetic Algorithm for Training Recurrent Neural Networks. Neural Network 14 (2001) 93-105.
- Pegalajar et al.(2000), "A genetic algorithm to obtain the optimal recurrent neural network". International Journal of Approximate Reasoning, 23, 67-83

Redes Neuronales Recurrentes y Series Temporales

M.Carmen Pegalajar
Dpto. Ciencias de Comput. E I.A.
E-mail:mcarmen@decsai.ugr.es

Redes Neuronales Recurrentes y Series Temporales

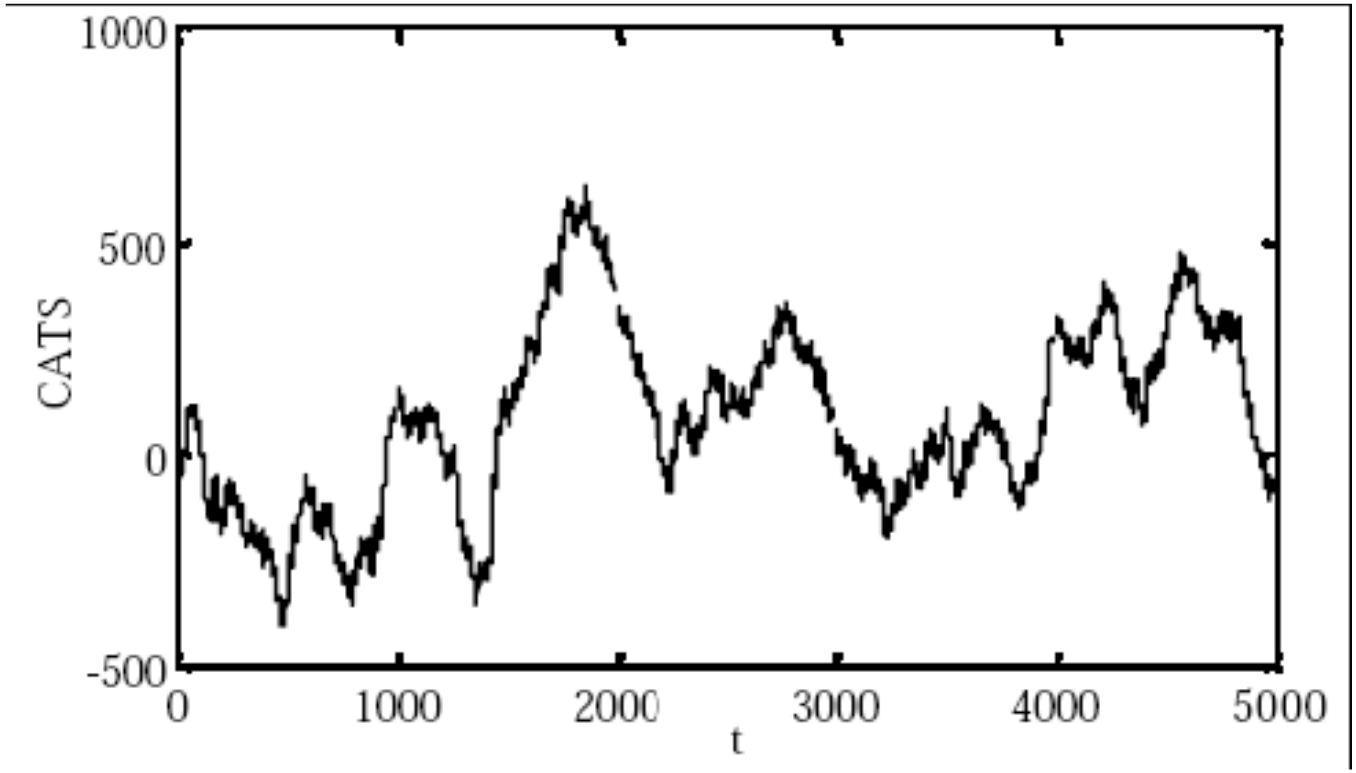
- Una serie temporal es una secuencia de observaciones realizadas sobre un fenómeno concreto e indexadas en el tiempo

- Su evolución temporal depende explícitamente de:

  - la variable tiempo

  - Valores de la misma serie, medidos en instantes anteriores al actual

Ejemplo



1. Modelado del problema

$$Y(t+1) = F(Y(t), Y(t-1), Y(t-2), \ldots)$$

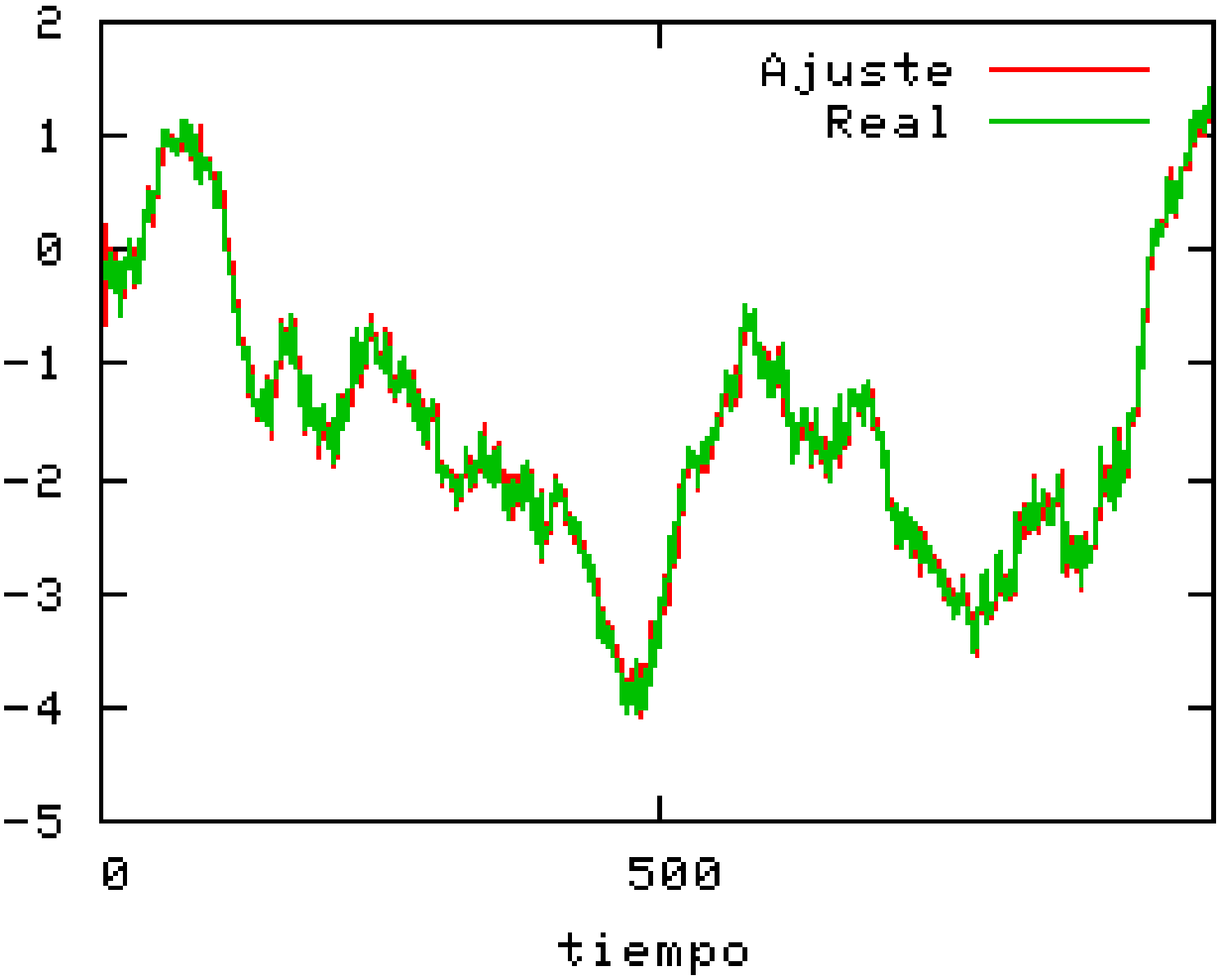
2. RNR aplicadas al aprendizaje y predicción de Series Temporales

- Se han entrenado tres tipos de redes: RNR Completa, RNR Elman y RNR Jordan.

- Los algoritmos de entrenamiento que obtuvieron mejores resultados fueron evolutivos, y evolutivos híbridos.

2. RNR aplicadas al aprendizaje y predicción de Series Temporales

- Problema: Benchmark CATS.
- Red que obtuvo los mejores resultados: RNR Completa entrenada con algoritmos evolutivos híbridos

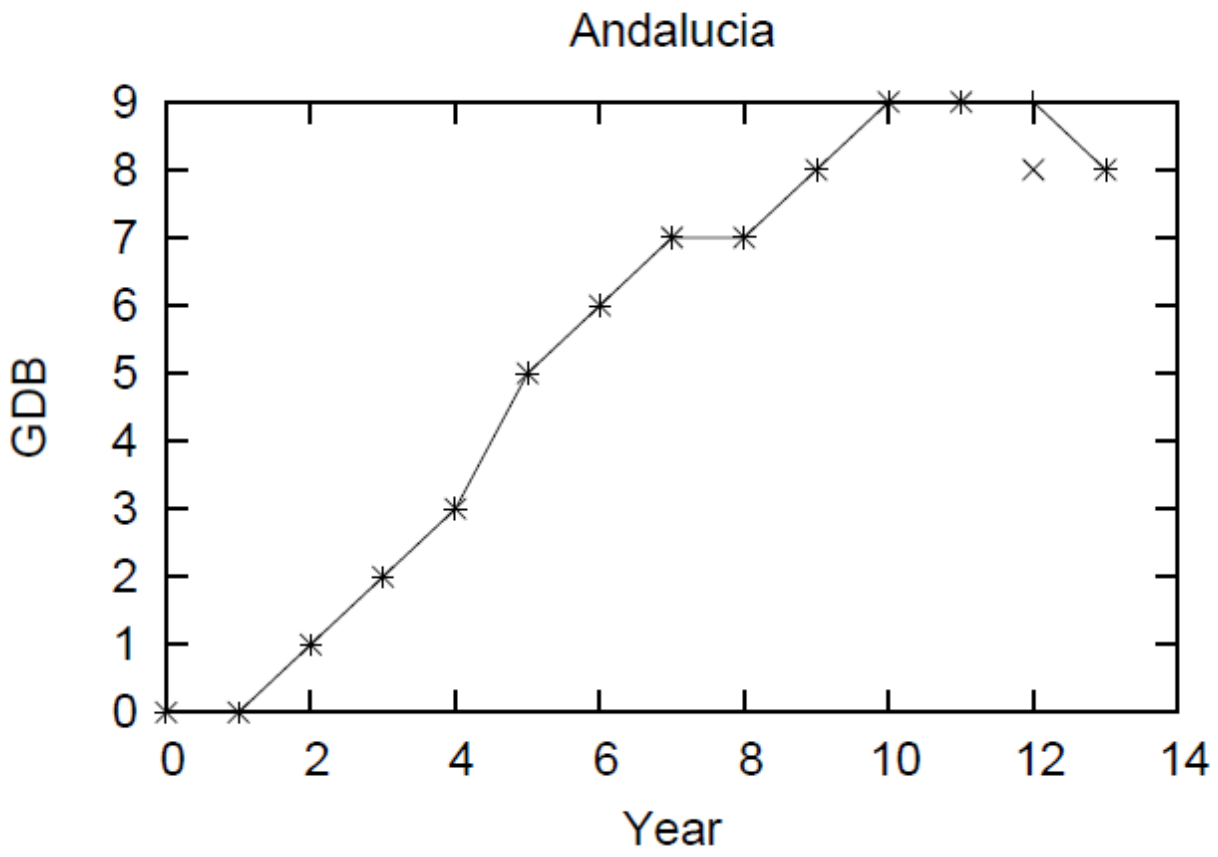

Ajuste
Real
2
1
0
-1
-2
-3
-4
-5
0
500
tiempo

Predicción del Endeudamiento Español por Regiones

- Conjunto de datos disponible pequeño, años 1986-1996, predicción para los años 1997-2000.
- Parámetro de las redes:
  - Número de Neuronas de Entrada :1 (valor de la serie temporal en el tiempo t)
  - Número de Neuronas Ocultas: 7
  - Número de Neuronas Salida: 1 (valor de la serie temporal en el tiempo t+1)
  - Función de Activación Sigmoidal y lineal para las unidades de salida.

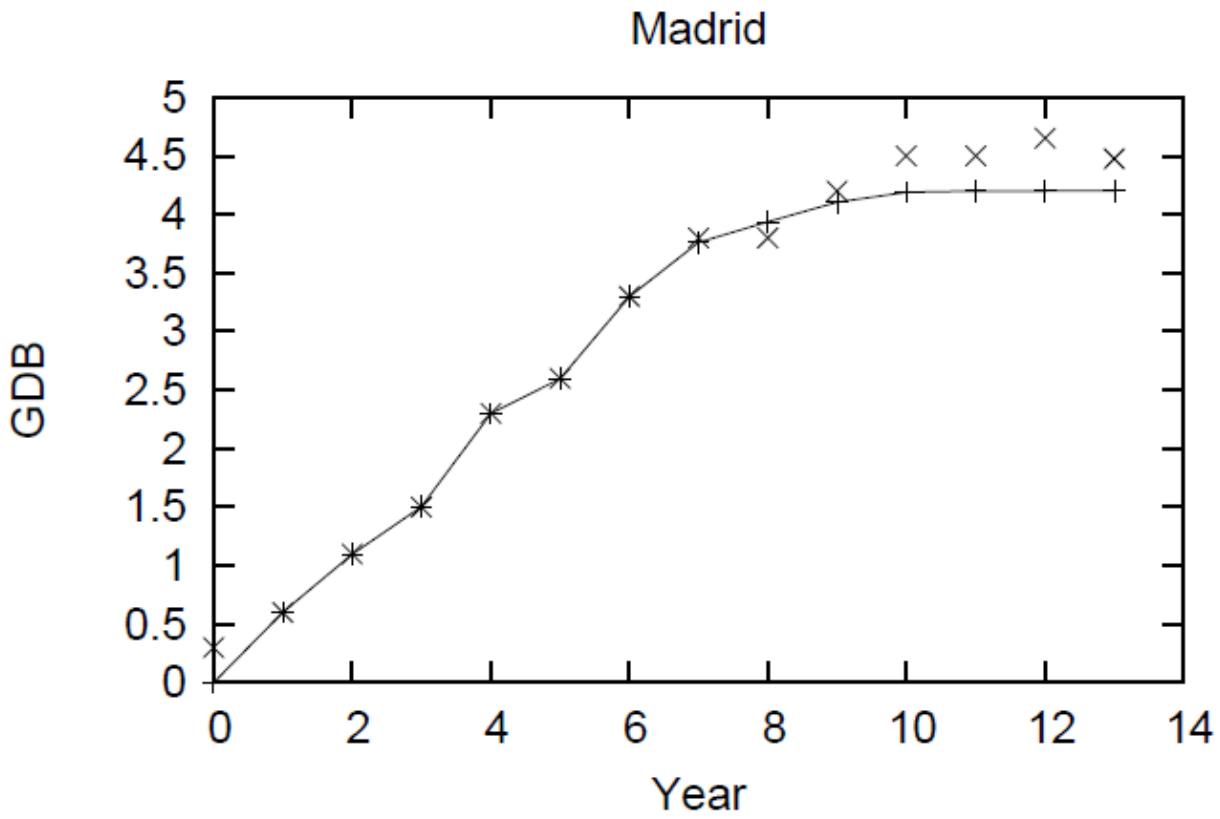
Andalucia

Algoritmo	Error Training	Error Test	Tiempo
BPTT	7.09e-01	1,62	3,17
RTRL	3,5	5,44	3,77
CHC	1.12e-02	2.67e-01	2,17


Andalucia
GDB
Year
0
1
2
3
4
5
6
7
8
9
0
2
4
6
8
10
12
14

Madrid

Algoritmo	Error Training	Error Test	Tiempo
BPTT	2.00e-01	4.18e-01	3,4
RTRL	8.11e-01	1,38	3,7
CHC	3.11e-03	1.74e-01	2,33


Madrid
GDB
Year
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
2
4
6
8
10
12
14

Bibliografía

- Aussem A., Dinamical Recurrent Neural Networks towards prediction and modeling of dynamical systems, Neurocomputing, vol. 28, no. 1-3, pp. 207-232,1999
- Danilo P. Mandic, Chambers J.A., (2001), Recurrent Neural Networks for Prediction, Ed. John Wiley & sons
- Michael Hüsken, Stagge P., (2003), Recurrent Neural Networks for Time Series classification, Neurocomputing, vol. 50, pp. 223-235.
- Pegalajar M.C., Navarro M.A. , Cuéllar M.P., Pérez R., (2003), A FIR Neural
- Network to model the autonomous indebtedness, in Proc. SIGEF'03, vol. 2, León 2003 (Spain), pp. 199-209
- Evolutionary Training for Dynamical Recurrent Neural Networks: An Application in Finantial Time Series Prediction. Mathware and Soft Computing 13 (2006), 89-110.
- Cuéllar M.P., Delgado M., Pegalajar M.C., (2005), An application of non-linear programming to train recurrent neural networks in time series prediction problems, in Proc. ICEIS.05, Miami (USA)
- Zemomi R., Racaceanu D., Zerhonn N., (2003), Recurrent Radial Basis fuction network for Time Series prediction, Engineering appl. Of Artificial Intelligence, vol. 16, no. 5-6, pp. 453-463.