UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

REDES NEURONALES ARTIFICIALES AUTO-ORGANIZATIVAS DINÁMICAS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

ÁLVARO SERRANO MUSALEM

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

MIEMBROS DE LA COMISIÓN:
JORGE VERGARA QUEZADA
PABLO HUIJSE HEISE

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: ÁLVARO SERRANO MUSALEM
FECHA: ENERO 2015
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

REDES NEURONALES ARTIFICIALES AUTO-ORGANIZATIVAS DINÁMICAS

El análisis de series de tiempo es de gran importancia ya que la gran mayoría de los datos que se miden cada día son función del tiempo. Su estudio está motivado tanto por el deseo de entender la naturaleza del sistema que se está midiendo, como también predecir su comportamiento futuro. Una técnica de análisis consiste en realizar una cuantificación espacio temporal, es decir, identificar y clasificar las secuencias de datos que presenten una dinámica espacio temporal similar.

La red neuronal artificial (RNA) Gas Neuronal Creciente (GNG: *Growing Neural Gas*) es un algoritmo diseñado para la cuantificación espacial de datos. Este trabajo de título busca extender la cuantificación espacial del algoritmo GNG a una cuantificación espacio temporal en series de tiempo. La extensión se realiza sobre la unidad básica, la neurona, la cual es extendida a un segmento temporal.

Se proponen dos extensiones de GNG las cuales son: (i) uso de conexiones temporales en lugar de conexiones espaciales, y (ii) utilización de segmentos en lugar de puntos. Los algoritmos propuestos se llaman GSG (*Growing Segment Gas*) y MGSG (*Merge Growing Neural Gas*). Este último introduce en GSG el uso de contextos para la cuantificación espacio temporal. Dada la forma en la que se extendió el algoritmo GNG, los algoritmos propuestos GSG y MGSG, resultan ser una generalización de GNG y MGNG, respectivamente.

El desempeño de GSG y MGSG se evalúa utilizando como métrica, el error de cuantificación temporal (TQE) para distintos retardos y se comparan los resultados con los obtenidos con la redes GNG y MGNG.

Los resultados obtenidos en MGSG muestran una ventaja estadísticamente significativa[1] en 3 de las 4 series de datos evaluadas, mientras que GSG sólo muestra ventajas sobre GNG en 2 de las 4 series de datos evaluadas. Además se muestran ventajas en términos de la visualización que otorgan la redes resultantes de GSG y MGSG. Considerando el buen desempeño de MGSG se propone como trabajo futuro construir algoritmos predictivos basados en MGSG para el estudio de series de tiempo. Por último los algoritmos propuestos son una primera aproximación a la cuantificación espacio temporal utilizando segmentos, por esta razón se exponen posibles mejoras de los algoritmos propuestos para trabajos futuros.

[1] Basado en el TQE utilizando la Prueba t de Student con una tolerancia de 5%.

Tabla de contenido

Notación

RNA: *Red Neuronal Artificial*
SOM: *Self Organizing Map*
NG: *Neural Gas*
GNG: *Growing Neural Gas*
TQE: *Temporal Quantization Error*
BMU: *Best Matching Unit*
MSOM: *Merge Self Organizing Map*
MNG: *Merge Neural Gas*
γ-GNG: *Gamma Growing Neural Gas*
GSG: *Growing Segment Gas*
MGSG: *Merge Growing Neural Gas*

1 Introducción

1.1 Motivación

Una serie de tiempo corresponde a una secuencia de datos ordenados cronológicamente en el tiempo. Las series de tiempo representan una gran cantidad de los datos existentes en el día a día, esto motiva el desarrollo de técnicas que permitan su estudio.

El análisis de series de tiempo tiene como objetivos identificar la naturaleza del sistema estudiado y la predicción de valores futuros. Estos objetivos dependen de la calidad de la representación de los patrones observados en la serie de tiempo. Una solución para realizar esto es construir una buena cuantificación espacio temporal de la serie de tiempo.

Gas Neuronal Creciente (GNG) corresponde a un algoritmo de cuantificación espacial de buen desempeño [1], que además considera relaciones de vecindad. Su buena cuantificación de sistemas estáticos, motiva a extender el algoritmo con el fin de incluir el tiempo y obtener así un algoritmo de cuantificación espacio temporal. Con una extensión desarrollada sobre GNG se espera una buena cuantificación que luego pueda ser utilizada para algoritmos predictivos. Además existe el deseo de desarrollar un algoritmo que entregue visualizaciones más representativas de los sistemas estudiados.

1.2 Objetivos

1.2.1 Objetivo general

El objetivo principal de este trabajo de título es extender la cuantificación espacial del algoritmo de gas neuronal creciente (GNG: *Growing Neural Gas*) a una cuantificación espacio temporal, con el fin de cuantificar eficientemente series de tiempo.

1.2.2 Objetivos específicos

Los objetivos específicos de este trabajo son:

- Analizar las ventajas y desventajas de GNG.
- Analizar la utilidad de la unidad básica de GNG (neurona puntual) en la cuantificación espacio temporal de series de tiempo.
- Desarrollar un algoritmo de cuantificación espacio temporal basado en GNG, extendiendo la unidad básica de GNG.
- Evaluar el desempeño del algoritmo propuesto utilizando métricas de cuantificación espacio temporal utilizadas en la literatura sobre series de tiempo.

1.3 Alcance

El aporte de este trabajo de título consiste en extender la actual cuantificación espacial del algoritmo GNG a una cuantificación espacio temporal. Esta cuantificación espacio

temporal intenta capturar de mejor manera la dinámica de los datos, con esto se espera abrir las puertas a trabajos futuros que implementen predicción basada en los algoritmos propuestos.

1.4 Estructura

La estructura de este informe es como sigue:

- **Capítulo 2 Estado del Arte:** Se introducen las redes neuronales artificiales no supervisadas como el punto de partida de este trabajo y se presentan los algoritmos básicos de esta área. Posteriormente se exponen las debilidades de estos algoritmos en torno al estudio de series de tiempo. Además se presentan las propuestas actuales que buscan solucionar estas limitaciones y se introduce la métrica llamada TQE (*Temporal Quantization Error*), que mide la distorsión en la cuantificación espacio temporal. Esta métrica será utilizada para evaluar el desempeño de los algoritmos propuestos para medir el desempeño de cuantificación espacio temporal.
- **Capítulo 3 Desarrollo de una red neuronal auto-organizativa con la capacidad de cuantificar espacial y temporalmente:** En este capítulo se construye la extensión de GNG para lograr una cuantificación espacio temporal. En la sección 3.3 y en la sección 3.4 se presenta la metodología para obtener los algoritmos propuestos GSG (*Growing Segment Gas*) y MGSG (*Merge Growing Segment Gas*) respectivamente. Finalmente se expone el proceso utilizado para evaluar los algoritmos propuestos.
- **Capítulo 4 Resultados:** Se presentan los resultados de GSG y MGSG en cuatro bases de datos: Mackey-Glass, Lorenz, Laser NH3-FIR y Rössler, y se compara el desempeño de estos algoritmos con los algoritmos GNG y MGNG. El desempeño de cada red se evalúa utilizando la métrica TQE.
- **Capítulo 5 Conclusiones:** Se concluye el trabajo a partir de los resultados obtenidos. Se realiza un resumen de las debilidades y ventajas de los algoritmos propuestos en base a los análisis realizados. Por último se proponen ideas para mejorar estos algoritmos en trabajos futuros.

2 Estado del Arte

En este capítulo se realiza una introducción a las RNA auto-organizativas, con el fin de mostrar el punto de partida de este trabajo, se introducen los métodos SOM, NG y GNG junto con sus limitaciones en el estudio de series de tiempo. Finalmente se introducen los métodos actuales que buscan solucionar tales limitaciones.

La detección de patrones es un proceso de gran complejidad para los sistemas inteligentes y se complejiza aún más cuando se tiene una gran cantidad de datos. Una buena solución a este problema consiste en la cuantificación de datos, en donde se intenta representar en una mínima cantidad de datos toda la dinámica y estructura original de los datos, simplificando el proceso de detección de patrones. Uno de los algoritmos que desarrolla esta tarea son las RNA no supervisadas, las cuales buscan capturar la información de todo el sistema en un subconjunto de vectores de referencia. Las RNA están inspiradas en el funcionamiento de las neuronas biológicas, emulando de éstas la habilidad de aprender y generalizar a partir de datos presentados a la red. La tolerancia frente a muestras ruidosas o incompletas de las RNA es una propiedad atractiva para problemas que involucran análisis de series de tiempo o reconocimiento de patrones.

Estas redes se separan en dos grupos según su naturaleza de aprendizaje: RNA supervisadas y RNA no supervisadas. Las del tipo supervisadas exigen que los datos estén separados en clases, pues su aprendizaje está determinado por el error que corresponde a la diferencia entre la salida de la red y la clase de la muestra. Lo anterior es una limitación pues se debe tener información previa con respecto a la clase de los datos, la cual no siempre se tiene. Por otro lado, las RNA no supervisadas extraen información directamente de los datos, siendo ideales para la detección de patrones o *clusters* en datos sobre los cuales no se tiene información previa.

2.1 SOM (*Self Organizing Maps*)

Dentro de las redes del tipo no supervisadas se encuentra un algoritmo introducido por Teuvo Kohonen llamado Mapa Auto-Organizativo (SOM: *Self Organizing Maps*) [2]. Este tipo de red permite proyectar un espacio de alta dimensión ($\mathbb{R}^d$) en un espacio de menor dimensión ($\mathbb{R}^a$) utilizando un mapeo topológicamente ordenado. Usualmente estas redes son utilizadas para visualizar datos en espacios de alta dimensión, por lo que se escoge un espacio de proyección con dimensión menor o igual a tres.

La arquitectura de la red neuronal SOM se define por N neuronas, cada una asociada a un vector prototipo $w_i \in \mathbb{R}^d$ en el espacio de entrada y a un vector de posición $p_i \in \mathbb{R}^a$ en la grilla de salida. Lo anterior se observa en la Figura 1, las neuronas de la red se encuentran al centro ilustrando que están asociadas a la izquierda con los vectores prototipos en el espacio de entrada (espacio original) y a la derecha con una posición en la grilla de salida (espacio proyectado).

El funcionamiento del algoritmo SOM comienza presentando los datos $x(t) \in \mathbb{R}^d$ a la red, las neuronas se activan según la proximidad de sus vectores prototipos w_i al vector de entrada $x(t)$ presentado en ese instante.

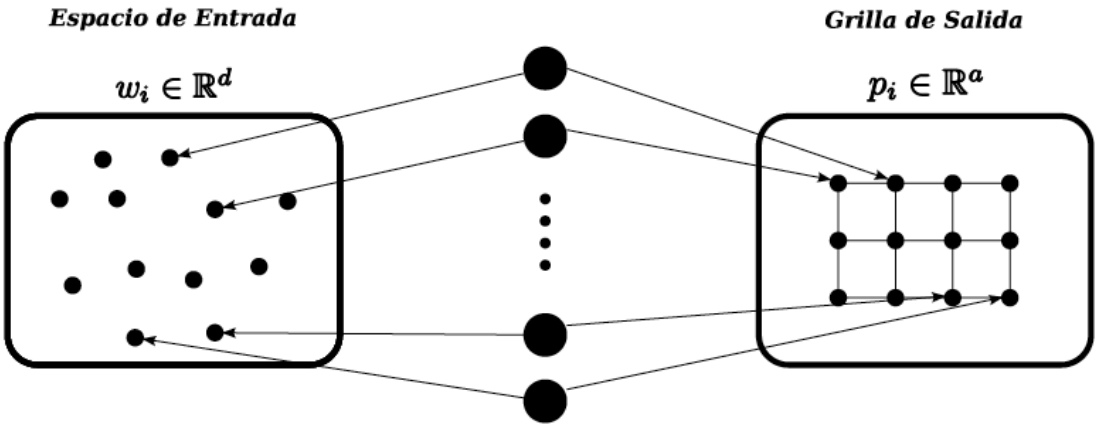


Figura 1: Arquitectura de la red SOM

La BMU (*Best Matching Unit*) se obtiene como la neurona cuyo vector prototipo está más cercano a $x(t)$. La obtención de la BMU se ilustra en Figura 2, en donde la muestra $x(t)$ está marcada con una X.

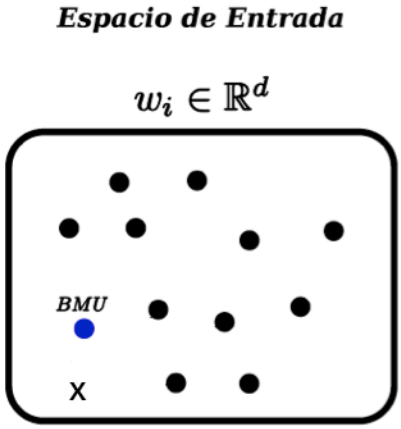


Figura 2: La BMU corresponde a la neurona cuyo vector prototipo se encuentra más cercano a la muestra actual denotada por una X.

Posteriormente se busca la posición de la BMU en la grilla de salida y se identifican las neuronas vecinas. Los vectores prototipos de la BMU y sus neuronas vecinas se ajustan hacia la muestra $x(t)$, incorporando así la información del nuevo dato en la red. El ajuste de la BMU es mayor que el de sus neuronas vecinas pues no es deseable que estas últimas pierdan el aprendizaje obtenido en etapas anteriores. En la Figura 3 se ilustra este proceso, primero se identifican los vecinos de la BMU, denotados por I, II y III en la grilla de salida; finalmente se ajustan sus vectores prototipos en el espacio de entrada.

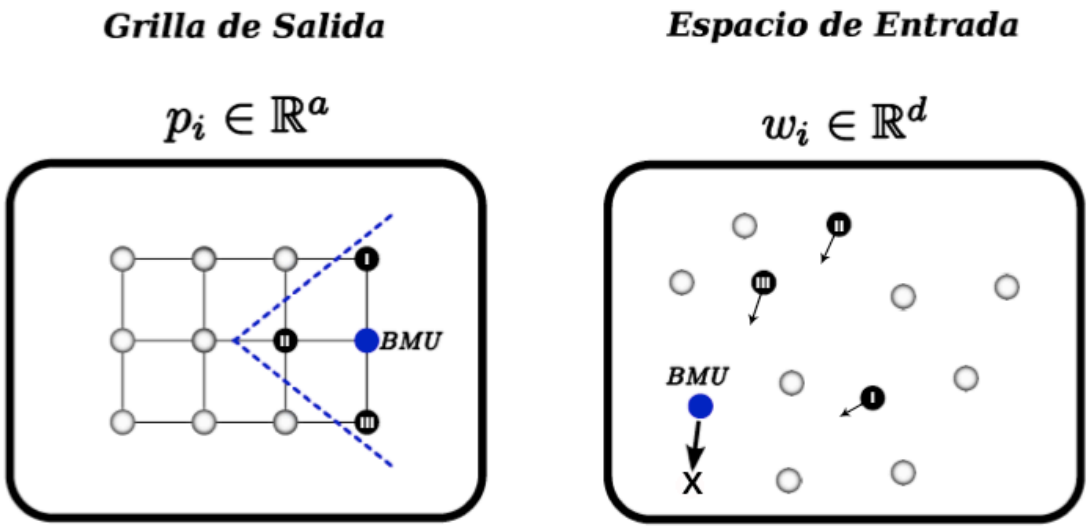


Figura 3: Etapa de aprendizaje de SOM. Una vez ya determinada la BMU, se ubican sus vecinos topológicos en la grilla de salida. Luego los vectores prototipo de la BMU y de sus vecinos se mueven hacia la muestra actual denotada por una X en el espacio de entrada.

Como resultado, se forma un mapeo en donde ciertas entradas activan ciertos sectores en la grilla de salida. El hecho de que dos entradas distintas generen patrones de actividad parecidos sugiere que ambas entradas poseen características similares, información con la cual se pueden encontrar *clusters* u otra información en los datos de entrada.

2.1.1 Algoritmo SOM: Pseudocódigo

El algoritmo SOM comienza con una etapa de inicialización y posteriormente con dos etapas: competencia y ajuste. Hay distintos tipos de inicialización para los vectores prototipos [3], por ejemplo:

- **Inicialización aleatoria:** Los vectores prototipos se fijan de manera aleatoria al comienzo del algoritmo .
- **Inicialización basada en las muestras:** Se inicializan los vectores prototipos de manera ordenada como una combinación lineal de los 2 vectores propios principales de los datos de entrada.

La etapa de competencia consiste en encontrar la BMU, es decir la neurona más cercana a la muestra $x(t)$ presentada en ese instante a la red. En una red con N neuronas la BMU $\hat{\imath}$ se obtiene según

$$\hat{\imath} = \underset{i=1\dots N}{argmin} \|x(t) - w_i(t)\|_2 \ , \tag{2.1}$$

en donde $x(t), w_i(t) \in \mathbb{R}^d$ la entrada a la red y el vector prototipo de la neurona i respectivamente en el tiempo t.
La etapa de ajuste busca acercar los vectores prototipo al vector de entrada. Para esto se define una función de vecindad gaussiana $h_{i,\hat{\imath}}(t)$ entre la neurona i y la BMU $\hat{\imath}$. La función de vecindad viene dada por

$$h_{i,\hat{\imath}}(t) = exp\left(-\frac{d_G(i,\hat{\imath})}{\sigma(t)}\right), \tag{2.2}$$

con $d_G(i,\hat{\imath})$ como la distancia entre la BMU y la neurona i; $\sigma(t)$ corresponde al ancho de la vecindad y decrece en el tiempo según

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{\frac{t}{T}}, \tag{2.3}$$

con σ_i y σ_f son parámetros definidos al inicio del entrenamiento, T corresponde al número máximo de épocas de entrenamiento. Si el valor inicial $\sigma_i > \sigma_f$, el tamaño de la vecindad ira decayendo en el tiempo a su valor final σ_f.

A partir de lo anterior, la ecuación que determina el aprendizaje de los vectores prototipos viene dada por

$$w_i(t+1) = w_i(t) + h_{i,\hat{\imath}}(t) \cdot \epsilon(t) \cdot [x(t) - w_i(t)] \quad \forall i \in 1, \dots, N, \tag{2.4}$$

en donde la tasa de aprendizaje $\epsilon(t)$ se define decreciente en el tiempo, con el fin de asegurar la convergencia del algoritmo, según

$$\epsilon(t) = \epsilon_i \left(\frac{\epsilon_f}{\epsilon_i}\right)^{\frac{t}{T}}, \tag{2.5}$$

con ϵ_i y ϵ_f parámetros definidos al inicio del entrenamiento. El utilizar $\sigma(t)$ y $\epsilon(t)$ decrecientes en el tiempo permite realizar un ajuste grueso y rápido al inicio, dando paso a un ajuste más fino a medida que avanza el algoritmo.

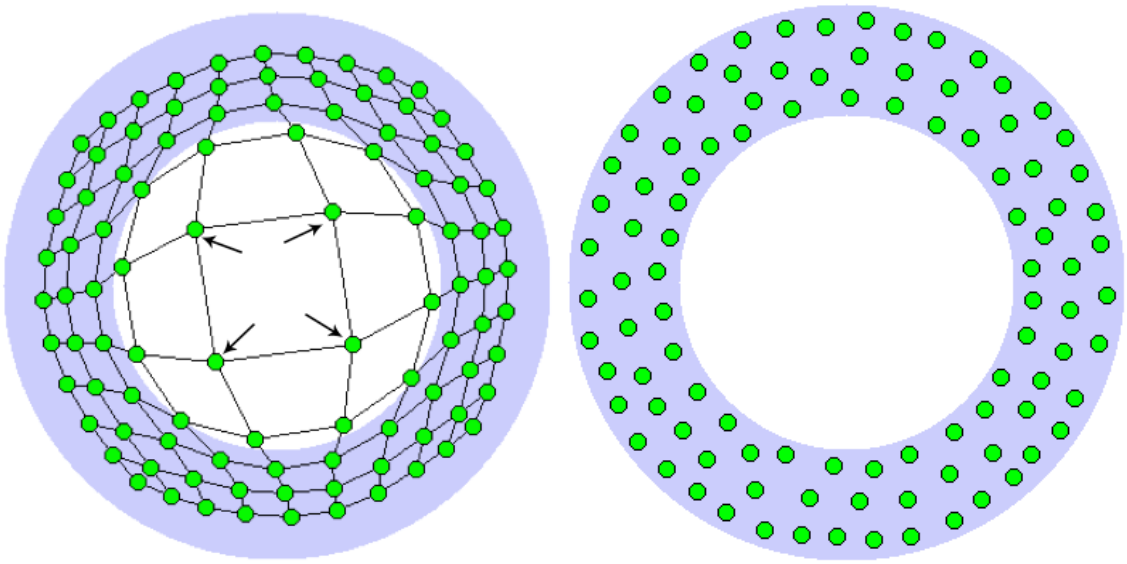
Los pasos del algoritmo SOM son los siguientes:

1. Inicializar aleatoriamente los vectores prototipos $w_i, i = 1, \dots, N$ donde N corresponde al número de neuronas de la red SOM.
2. Presentar un vector de entrada $x(t)$ en el tiempo t.
3. Obtener BMU $\hat{\imath}$ haciendo uso de la ecuación (2.1).
4. Actualizar los vectores $w_i(t)$ utilizando la ecuación (2.4).
5. $t \to t + 1$.
6. Volver a 2 si $t < T$, donde T es el número máximo de épocas de entrenamiento.

2.2 Gas Neuronal

Gas Neuronal (NG) [4] es una RNA no supervisada similar a SOM, pero que no posee una topología restrictiva (grilla) en las neuronas de salida. La Figura 4 es utilizada para ilustrar esta diferencia al visualizar como las redes se ajustan a los datos cuya distribución tiene forma de anillo. La red SOM a la izquierda posiciona neuronas de manera incorrec-

ta, al interior de anillo, donde no hay datos. Por otro lado, NG al no tener una grilla restrictiva, distribuye uniformemente sus neuronas sobre el anillo.



(a) (b)

Figura 4: Comparación topológica entre SOM y NG frente a una misma distribución de datos de entrada con forma de anillo. a) Corresponde a SOM, la naturaleza de su topología genera neuronas mal ubicadas (indicadas con las flechas). b) NG no tiene restricciones topológicas y las neuronas se distribuyen de manera correcta. Figura obtenida de [5].

Otra diferencia de NG con respecto a SOM consiste en que NG busca reducir una función de costos global, la cual determina las reglas de ajuste de las neuronas de la red.

La mayor debilidad de SOM es que no busca reducir una función de costos global, sino que busca reducir el error local entre la muestra actual y la BMU (es un criterio heurístico). En cambio, NG tiene una función de costos definida, la cual se busca optimizar y desde la cual se deriva la regla de ajuste. La función de costos de NG se define como

$$E(w,\sigma) = \frac{1}{2C(\sigma)} \sum_{i=1}^{N} \int \|x - w_i\|^2 \, h_\sigma\big(k_i(x,w)\big) \, p(x) \, \partial x. \tag{2.6}$$

La función $k_i(x,w) \in \{0,\dots,N-1\}$ entrega la posición en el ranking de cercanía a la muestra $x \in \mathbb{R}^d$ en el que está la neurona i, con respecto a las otras neuronas de la red. La función de vecindad h_σ entrega un valor dependiendo de la cercanía de la neurona i al dato x, siendo máximo para la neurona más cercana a x y decreciendo a medida que se consideran neuronas más alejadas. La función de vecindad h_σ se define por

$$h_\sigma\big(k_i(x,w)\big) = e^{-\frac{k_i(x,w)}{\sigma}}. \tag{2.7}$$

$C(\sigma)$ tiene un fin normalizador y viene dado por $C(\sigma) = \sum_{i=1}^{N} h_\sigma(k_i) = \sum_{k=0}^{N-1} h_\sigma(k)$. El parámetro $\sigma \in \mathbb{R}$ determina el ancho de la vecindad considerada y $p(x)$ corresponde a la

distribución de probabilidad de la entrada, la cual es desconocida y estacionaria. De la ecuación (2.6) se obtiene el valor esperado del error de cuantificación de la red. Al minimizar la función de costos (2.6) se obtiene la regla de ajuste para NG

$$\Delta w_i = \epsilon(t) h_\sigma\big(k_i(x, w)\big)(x - w_i) \quad \forall i \in 1, \dots, N, \tag{2.8}$$

con $\epsilon(t)$ la tasa de aprendizaje decreciente en el tiempo definida tal como en la ecuación (2.5) del algoritmo SOM. De esta manera el ajuste de las neuronas más cercanas a x es mayor que el de aquellas neuronas más alejadas de x.

El algoritmo NG es:

1. Inicializar aleatoriamente los vectores prototipo $w_i,\ i = 1, \dots, N$, donde N es el número total de neuronas en la red NG.
2. Presentar un vector de entrada $x(t)$.
3. Obtener ranking de vecindad $(w_{i_0},\ w_{i_1}, \dots, w_{i_{N-1}})$, en donde w_{i_0} es la neurona más cercana a $x(t)$, w_{i_1} es la segunda más cercana y así sucesivamente.
4. Calcular $k_i(x, w) \in \{0, \dots, N-1\}$ como la posición de w_i en el ranking de vecindad, siendo 0 para la neurona más cercana y $N-1$ para la más lejana de $x(t)$.
5. Actualizar todas las neuronas según

$$\Delta w_i = \epsilon(t) h_\sigma(k_i(x, w))(x(t) - w_i) \quad \forall i = 1 \dots N. \tag{2.9}$$

6. $t \to t + 1$
7. Volver a 2 si $t < T$, donde T es el número máximo de épocas de entrenamiento.

2.3 Gas Neuronal Creciente

Gas Neuronal Creciente (GNG) [6] fue propuesto por Bernd Fritzke y corresponde a una extensión de NG, ya que implementa conexiones entre neuronas, así como también reglas de inserción y remoción de neuronas. Esto permite que el número de neuronas en la red crezca y decrezca según sea necesario al mantener una medida local del error acumulado en cada neurona de la red. Además, inserta una nueva neurona cada λ pasos, en el punto medio entre la neurona con el mayor error acumulado y su vecina con el mayor error acumulado. Las conexiones se generan entre las 2 neuronas más cercanas al dato actual, poseen un contador de edad de las conexiones el cual permite eliminar conexiones que no hayan sido activadas. Además al final de cada iteración si una neurona no posee ninguna conexión entonces se elimina.

A modo de ejemplo la Figura 5, muestra la evolución del algoritmo GNG sobre una base de datos cuya distribución tiene forma de anillo. En (a) se muestra la inicialización de 2 neuronas desconectadas en el espacio de entrada. Luego (b) corresponde a la red GNG después de 2600 pasos, es decir 2600 datos presentados a la red, en donde el número de neuronas y conexiones creció a 6. En (c) y (d) se muestra la red con 19200 y 35600 pasos, respectivamente (19200 y 35600 datos presentados, respectivamente).

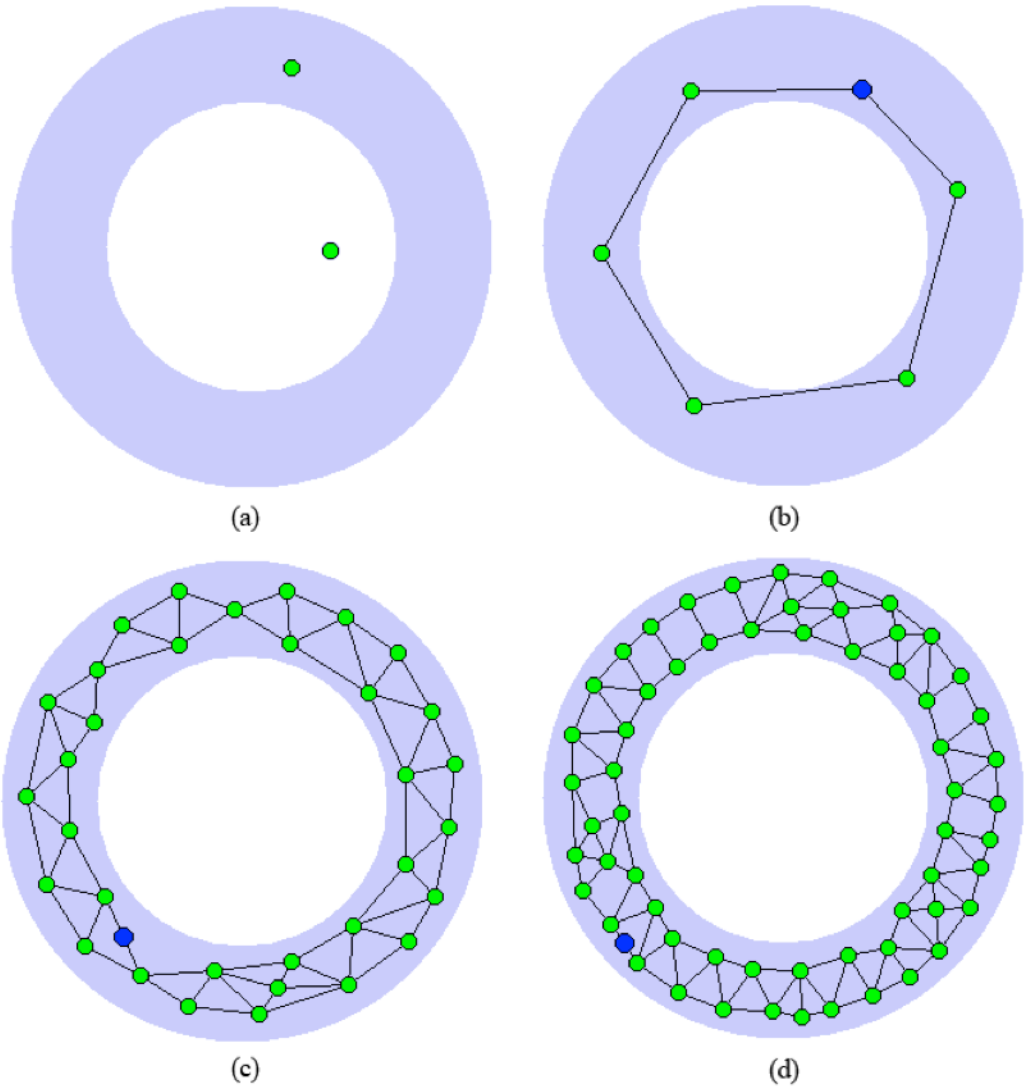


Figura 5: Evolución de GNG para un conjunto de datos de entrada con distribución con forma de anillo, con cada iteración se presenta un dato nuevo a la serie. a) Se inician con 2 neuronas aleatorias, b) luego de 2600 iteraciones c) 19200 iteraciones y d) 35600 iteraciones. Figura obtenida de [5].

El algoritmo GNG propuesto por Bernd Fritzke considera:

- Un conjunto A de índices de neuronas. Cada neurona $i \in A$ tiene asociada un vector de referencia $w_i \in \mathbb{R}^d$.
- Un conjunto C de conexiones entre pares de neuronas, cada conexión tiene asociada un entero que denota la edad de la conexión.
- Cada neurona $i \in A$ posee una variable local, $error(i) \in \mathbb{R}$, que mide el error acumulado por la neurona hasta el momento.

Tomando en cuenta lo anterior el algoritmo GNG es el siguiente:

1. Comenzar con 2 neuronas desconectadas, $A = \{1, 2\}$, con vectores w_1 y w_2 inicializados aleatoriamente en el espacio de entrada $\mathbb{R}^d$. Además inicializar el error acumulado de ambas neuronas en cero, $error(1) = error(2) = 0$.
2. Presentar un vector de entrada, $x(t)$, a la red GNG.
3. Encontrar la BMU de $x(t)$, I_t y la segunda neurona más cercana J_t mediante

$$I_t = \underset{i \in A}{argmin}\, d_i(t), \qquad J_t = \underset{i \in A \setminus I_t}{argmin}\, d_i(t), \tag{2.10}$$

   utilizando $d_i(t) = \|x(t) - w_i\|$ y A el conjunto de índices de las neuronas de la red GNG.
4. Incrementar en uno la edad de todas las conexiones que salen de I_t.
5. Actualizar el error acumulado de I_t según

$$\Delta error(I_t) = \left\|x(t) - w_{I_t}\right\|^2. \tag{2.11}$$

6. Mover la BMU y sus vecinos hacia $x(t)$ utilizando

$$\Delta w_{I_t} = \epsilon_w\left(x(t) - w_{I_t}\right), \qquad \Delta w_j = \epsilon_n\left(x(t) - w_j\right) \quad \forall j \in N_{I_t}, \tag{2.12}$$

   donde N_{I_t} corresponde al conjunto de los vecinos de I_t. El parámetro $\epsilon_w \in \mathbb{R}$ es la tasa de aprendizaje para la BMU y $\epsilon_n \in \mathbb{R}$ la tasa de aprendizaje para los vecinos de I_t.
7. Si no existe una conexión entre I_t y J_t crearla. Fijar la edad de la conexión entre ambas neuronas en cero.
8. Remover las conexiones cuya edad sea mayor a la edad máxima a_{max}. Eliminar las neuronas sin conexiones.
9. Si la iteración actual, t, es un entero múltiplo del parámetro λ, entonces insertar una nueva neurona a la red. La inserción de una nueva neurona es según:
   a. Buscar en la red GNG la neurona u con el mayor error acumulado.
   b. Entre los vecinos de u, encontrar aquella neurona v con el mayor error acumulado.
   c. Insertar una neurona r entre u y v según

$$w_r = 0.5w_u + 0.5w_v. \tag{2.13}$$

   d. Crear conexiones entre u y r, entre r y v; remover la conexión entre las neuronas u y v.

e. Reducir el error acumulado de las neuronas u y v multiplicado sus errores acumulados por una constante α, fijar el error acumulado de la neurona r como el nuevo valor del error acumulado de u.

10. Reducir el error acumulado de todas las neuronas multiplicando sus errores acumulados por una constante β.
11. Incrementar el número de la iteración ($t = t + 1$) y volver al punto 2 si no se ha alcanzado algún criterio de término (e.g., número máximo de neuronas).

Como se expuso, el algoritmo GNG propone insertar neuronas cerca de aquellas que poseen el mayor error acumulado. Sin embargo, en [7] proponen la maximización de la entropía como criterio de inserción de neuronas. Esto se hace balanceando el número de victorias entre las neuronas, de esta manera se evita el caso extremo de neuronas que no representan nada. Esta última variante se implementa haciendo unos pocos cambios al algoritmo GNG:

- Para todas las neuronas $i \in A$ se sustituye la variable local $error(i)$ por un contador de victorias $wcount_i$.
- En el paso 1 del algoritmo GNG los contadores de victoria se inicializan en cero para ambas neuronas.
- En el paso 5 se incrementa en una unidad el contador de victorias de la BMU mediante

$$wcount_{I_t} = wcount_{I_t} + 1. \tag{2.14}$$

- En el paso 9 los pasos para insertar una nueva neurona se cambian como sigue:
  - Buscar en la red GNG la neurona u con la mayor cantidad de victorias.
  - Entre los vecinos de u, encontrar aquella neurona v con la mayor cantidad de victorias.
  - Insertar una neurona r entre u y v según

$$w_r = 0.5w_u + 0.5w_v. \tag{2.15}$$

  - Crear conexiones entre u y r, entre r y v; remover la conexión entre las neuronas u y v.
  - Reducir el número de victorias de las neuronas u y v multiplicado sus contadores de victorias por una constante α, fijar el número de victorias de la neurona r como el nuevo número de victorias de u.
- Por último, en el paso 10 en lugar de reducir el error acumulado, se reduce el número de victorias de todas las neuronas multiplicando el contador de victorias por β.

2.4 Reconstrucción del espacio de estado.

El espacio de estado es el conjunto de todos los posibles estados de un sistema dinámico. Si bien no siempre es posible tener acceso a las variables de estado de un sistema, bajo ciertas condiciones es posible realizar una reconstrucción del espacio de estado a partir de una sola variable. Una técnica para la reconstrucción del espacio de estados es la pro-

puesta por Takens [8], que permite obtener el comportamiento de la dinámica interna del sistema a partir de la serie de tiempo de sólo una variable de estado.

Si $x(t) \in \mathbb{R}$ es una serie de tiempo, el vector $s(t)$ corresponde a una reconstrucción de dimensión m y retardo τ del espacio de estado original:

$$s(t) = [x_i(t), x_i(t-\tau), \ldots, x_i(t-(m-1)\times\tau)]. \quad (2.16)$$

El interés por estudiar sistemas dinámicos no lineales es una de las motivaciones para desarrollar redes neuronales capaces de incorporar la dependencia temporal de los datos. Métodos como SOM, NG y GNG cuantifican los datos sólo espacialmente, por lo tanto es conveniente extenderlos al tiempo.

En la literatura, métodos como el Mapa Temporal de Kohonen (TKM) [9], Mapa Auto-organizativo Recurrente (RSOM) [10] [11], SOMTAD(*Self-Organizing Maps with Temporal Activity Diffusion*), GASTAD (*Neural Gas with Temporal Activity Diffusion*) [12], Mapa Auto-organizativo Recursivo (RecSOM) [13], MSOM (*Merge SOM*) junto con MNG (*Merge NG*) [14] y MGNG (*Merge Growing Neural Gas*) [7]; intentan incorporar el tiempo a los métodos de cuantificación espacial tales como SOM, NG y GNG.

2.5 Error de Cuantificación Temporal

El Error de Cuantificación Temporal (TQE) [13] [15], es una medida de distorsión espacio temporal. El TQE es una métrica que indica el grado de dispersión (desviación estándar) de las secuencias de datos asociadas a cada neurona.

Como ejemplo en la Figura 6, se considera una neurona que cuantifica los 4 puntos mostrados en $\tau = 0$, hacia la derecha se muestran los datos de la serie para distintos retardos de tiempo. Si se considera un cierto retardo τ, el TQE de la neurona corresponde a la desviación estándar entre los puntos A, B, C, D.

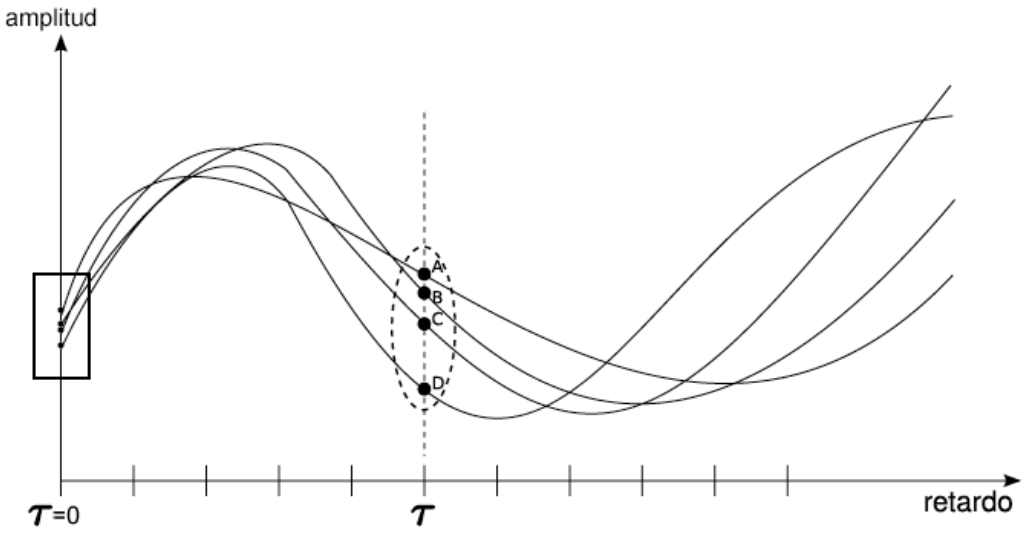


Figura 6: Ejemplo para mostrar los puntos sobre los cuales el TQE calcula la desviación estándar.

Para el cálculo del TQE primero se obtiene $A_i(\tau)$, el valor promedio de los datos ubicados a un retardo τ de aquellos datos cuantificados por una neurona i. Para esto es necesario definir un conjunto Θ_i, el cual contiene todos los tiempos en los que la neurona i es la BMU. Con esto $A_i(\tau)$ se calcula a partir de

$$A_i(\tau) = \frac{1}{|\Theta_i|} \sum_{j \in \Theta_i} x(j - \tau), \tag{2.17}$$

en donde $x(j - \tau)$ es el valor de la serie de tiempo en $j - \tau$ y $|\Theta_i|$ la cardinalidad del conjunto Θ_i. Así se puede obtener el TQE de la neurona i para un retardo τ

$$E_i(\tau) = \sqrt{\frac{1}{|\Theta_i|} \sum_{j \in \Theta_i} \big(A_i(\tau) - x(j - \tau)\big)^2}. \tag{2.18}$$

Al obtener el promedio del TQE entre todas las neuronas, se obtiene el error del mapa $e(\tau)$ en el retardo τ, es decir

$$e(\tau) = \frac{1}{M} \sum_{j \in \mathcal{N}} E_i(\tau). \tag{2.19}$$

Algoritmos como SOM, NG y GNG; obtienen un error $e(0)$ muy bajo, pues $e(0)$ corresponde al error de cuantificación espacial. Sin embargo, al considerar retardos mayores el error de cuantificación temporal crece pues SOM, NG y GNG no fueron diseñados para la cuantificación espacio temporal.

2.6 Merge SOM (MSOM)

Este algoritmo extiende SOM con un modelo de contextos que le permite cuantificar secuencias temporales. El contexto en MSOM se refiere a la fusión de dos propiedades que caracterizan al último ganador de la red: el vector de referencia w_i y el contexto c_i de la última neurona ganadora. Así el contexto es el resultado de unir, a través de una combinación lineal, w_i y c_i de la neurona ganadora.

En MSOM una neurona i está definida en el espacio de entrada por un par $(w_i,\ c_i) \in \mathbb{R}^d \times \mathbb{R}^d$, en donde w_i representa la entrada esperada por la red en ese instante, mientras que c_i representa el contexto esperado por la red. Dada una entrada $x(t)$, la neurona ganadora es la neurona que minimiza el siguiente funcional de distancia:

$$d_i(t) = (1 - \alpha_m) \cdot \|x(t) - w_i\|^2 + \alpha_m \cdot \|c(t) - c_i\|^2, \tag{2.20}$$

donde ambas contribuciones ($\|x(t) - w_i\|^2$ y $\|c(t) - c_i\|^2$) están balanceadas por el parámetro α_m. El descriptor de contexto $c(t)$ se define según

$$c(t) = (1 - \beta_m) \cdot w^{I_{t-1}} + \beta_m \cdot c_{I_{t-1}} \tag{2.21}$$

con $c_{I_{t-1}}$ corresponde al contexto de la neurona con el mejor ranking en el tiempo $t-1$. La condición inicial es un valor fijo $c(0)=0$ y $\beta_m \in (0,1)$, típicamente 0.5 [14].

El entrenamiento de la red corresponde al ajuste de los vectores de peso hacia la entrada actual y del contexto de las neuronas hacia el descriptor de contexto de la red. Según una función de vecindad, $d_{\mathcal{N}}: A \times A \to \mathbb{R}$, centrada en la neurona ganadora, el ajuste en MSOM viene dado por

$$\begin{aligned} \Delta w_i &= \epsilon \cdot h_\sigma\big(d_A(i, I_t)\big) \cdot (x(t) - w_i), \\ \Delta c_i &= \epsilon \cdot h_\sigma\big(d_A(i, I_t)\big) \cdot (c(t) - c_i), \end{aligned} \tag{2.22}$$

donde ϵ es la tasa de aprendizaje y h_σ es típicamente una función gaussiana [14].

2.7 Merge NG

El modelo de contextos se puede aplicar también a NG. El funcional de distancia es el mismo de MSOM pero al no haber restricciones topológicas, se realiza un ranking de proximidad

$$k = rnk(i) = \big|\{j \big| d_j(t) < d_i(t)\}\big|, \tag{2.23}$$

donde $d_i(t)$ es el expuesto en la ecuación (2.20). La ecuación (2.23) mide el ranking de la neurona i con respecto a las otras neuronas en el tiempo t. Se utiliza este ranking para el cálculo de la adaptación

$$\begin{aligned} \Delta w_i &= \gamma_1 \cdot exp\left(-\frac{rnk(i)}{\sigma}\right) \cdot (x(t) - w_i), \\ \Delta c_i &= \gamma_2 \cdot exp\left(-\frac{rnk(i)}{\sigma}\right) \cdot (c(t) - c_i). \end{aligned} \tag{2.24}$$

El descriptor de contexto es equivalente al de MSOM y se debe actualizar para ir manteniendo la información de la última neurona con el mejor ranking.

2.8 Gamma GNG

Gamma GNG (γ-GNG) [16] [17] corresponde al resultado de introducir memorias Gamma al algoritmo GNG. Es un tipo de RNA auto-organizativa dinámica la cual posee un modelo de contextos que le permite recordar valores a corto plazo, el modelo de contextos que se utiliza se basa en las memorias Gamma de De Vries y Principe [18] [19] [20]. Si bien en este trabajo se muestra sólo γ-GNG cabe notar que también existe γ-NG [21] y γ-SOM [22].

2.8.1 Memorias Gamma

Corresponde a un filtro Gamma que combina las propiedades de los filtros FIR con la potencia de los filtros IIR. Se define en el dominio del tiempo según

$$y(t) = \sum_{k=0}^{K} w_k \cdot c_k(t),$$
$$c_k(t) = \beta c_k(t-1) + (1-\beta)c_{k-1}(t-1). \tag{2.25}$$

Con $c_0(t) \equiv x(t)$ la señal de entrada, $y(t)$ la salida del filtro, $w_0, \dots, w_K$ los pesos del filtro. $\beta \in (0,1)$ es un parámetro utilizado para separar las características de profundidad (D) y resolución (R) del filtro. La profundidad representa la capacidad de las memorias para retener valores pasados, mientras que la resolución específica la precisión con la que se almacenaron estos valores. Los valores de la profundidad y resolución vienen dados por

$$D = \frac{K}{(1-\beta)},$$
$$R = 1 - \beta. \tag{2.26}$$

2.8.2 Modelo de contextos de γ-GNG

Se considera una red neuronal con *N* neuronas, cada neurona con su vector $w_i \in \mathbb{R}^D$ y con un conjunto de contextos $\mathcal{C}_i = \{c_1^i, \dots, c_K^i\}$, $c_k^i \in \mathbb{R}^d, k = 1, \dots, K$, con K el orden del filtro Gamma. Los contextos de cada neurona se inicializan en 0.
Tomando una señal de entrada $x(t)$ la BMU, I_t, corresponderá a la neurona que minimice el siguiente criterio de distancia

$$d_i(t) = \alpha_w \|x(t) - w_i\|^2 + \sum_{k=1}^{K} \alpha_k \left\|c_k(t) - c_k^i\right\|^2. \tag{2.27}$$

Los parámetros α_w y α_k controlan el balance entre $\|x(t) - w_i\|^2$ y $\left\|c_k(t) - c_k^i\right\|^2$. El descriptor de contextos $c_k(t)$ se calcula a partir de

$$c_k(t) = \beta c_k^{I_{t-1}} + (1-\beta)c_{k-1}^{I_{t-1}} \quad \forall k = 1, \dots, K, \tag{2.28}$$

en donde $c_k^{I_{t-1}}$ es el contexto k de la BMU del tiempo $t-1$. Además $c_0^{I_{t-1}} \equiv w_{I_{t-1}}$ y los valores iniciales de $c_k^{I_0}$ son inicializados de manera aleatoria $\forall k = 1, \dots, K$.

Debido a la naturaleza recursiva en la construcción del contexto es recomendable que $\alpha_w > \alpha_1 > \cdots > \alpha_K > 0$, de lo contrario es muy probable que la mala cuantificación en

las primeras etapas de filtrado produzca errores en la construcción de los contextos de orden superior [17].

2.8.3 Algoritmo γ-GNG

Al considerar el modelo de contextos presentado anteriormente en GNG se obtiene γ-GNG. El sistema comienza con 2 neuronas. Cada neurona $i \in A$ tiene asociada un vector prototipo d -dimensional, w_i y un conjunto de contextos d -dimensionales c_k^i con $k \in \{1, \dots, K\}$.

El algoritmo γ-GNG es el siguiente:

1. Inicializar aleatoriamente los vectores prototipos, w_1 y w_2. Inicializar sus contextos c_k^i en cero $k = 1, \dots, K,\ i = 1, 2$. Conectar ambas neuronas con una conexión de edad cero y fijar sus contadores de triunfos en cero, $wcount_i = 0$.
2. Presentar un vector de entrada, $x(t)$, a la red γ-GNG.
3. Calcular el descriptor de contexto $c_k(t)$ de la red utilizando la ecuación (2.28).
4. Encontrar la BMU de $x(t)$, I_t y la segunda neurona más cercana J_t mediante

$$I_t = \underset{i \in A}{argmin}\, d_i(t), \qquad J_t = \underset{i \in A \setminus I_t}{argmin}\, d_i(t), \tag{2.29}$$

   con $d_i(t)$ la distancia definida en la ecuación (2.27).
5. Incrementar en uno la edad de todas las conexiones que salen de I_t.
6. Actualizar el contador de triunfos de la BMU I_t, $wcount_{I_t} = wcount_{I_t} + 1$
7. Adaptar los vectores prototipos y contextos de la BMU utilizando

$$\Delta w_{I_t} = \epsilon_w(t)\big(x(t) - w_{I_t}\big), \qquad \Delta c_k^i = \epsilon_w(t)\big(c_k(t) - c_k^i\big)\ \ \forall k \in \{1, \dots, K\}, \tag{2.30}$$

   y luego adaptar los vecinos de la BMU utilizando

$$\Delta w_n = \epsilon_n(t)(x(t) - w_n), \qquad \Delta c_k^j = \epsilon_n(t)\big(c_k(t) - c_k^j\big)\ \ \forall k \in \{1, \dots, K\}, \tag{2.31}$$

   para todo $j \in N_{I_t}$, donde N_{I_t} es el conjunto de índices de todos los vecinos de la BMU I_t.
8. Si no existe una conexión entre I_t y J_t crearla. Fijar la edad de la conexión entre ambas neuronas en cero.
9. Remover las conexiones cuya edad sea mayor a la edad máxima a_{max}. Eliminar las neuronas sin conexiones.
10. Si la iteración actual t es un entero múltiplo del parámetro λ, entonces insertar una nueva neurona a la red. La inserción de una nueva neurona es según:

a. Buscar en la red GNG la neurona u con la mayor cantidad de victorias.
b. Entre los vecinos de u, encontrar aquella neurona v con la mayor cantidad de victorias.
c. Insertar una neurona r entre u y v según

$$w^r = 0.75w^u + 0.25w^v,$$
$$c_k^r = 0.75c_k^u + 0.25c_k^v. \tag{2.32}$$

d. Crear conexiones entre u y r, entre r y v; remover la conexión entre las neuronas u y v
e. Reducir el número de victorias de las neuronas u y v multiplicado sus contadores de victorias por una constante α, fijar el número de victorias de la neurona r como el nuevo número de victorias de u.

11. Reducir el número de victorias de todas las neuronas multiplicando el contador de victorias de cada una por β.
12. Incrementar el número de la iteración ($t = t + 1$) y volver al punto 2 si no se ha alcanzado algún criterio de término (e.g., número máximo de neuronas).

2.9 Neurona segmento

En [23] se propone una extensión de SOM para la utilización del algoritmo en el seguimiento de manos y cuerpos humanos. Parte de la extensión realizada consiste en el uso de segmentos como unidad básica.

2.9.1 Definición de un segmento

Un segmento s_i está definido por el trazo entre dos vectores w_1^i, $w_2^i \in \mathbb{R}^d$, es decir

$$s_i = \{\, \hat{x} \mid \hat{x} = \alpha \cdot w_1^i + (1-\alpha) \cdot w_2^i \quad \forall \alpha \in [0,1] \}. \tag{2.33}$$

2.9.2 Medida de distancia

La distancia $\|d(t)\|^2$ entre un punto $x \in \mathbb{R}^d$ y un segmento s definido por los vectores $w_1, w_2 \in \mathbb{R}^d$, se obtiene proyectando el punto x sobre el segmento, y calculando la distancia entre x y su proyección p, ver Figura 7. Definiendo el vector $d_{12} = w_1 - w_2$ y $d_{21} = w_2 - w_1$, entonces la proyección p de x sobre el segmento viene dada por

$$p = w_2 + \eta_{12} \cdot d_{12}, \qquad 0 \le \eta_{12} \le 1, \tag{2.34}$$

ó por

$$p = w_1 + \eta_{21} \cdot d_{21}, \qquad 0 \le \eta_{21} \le 1, \tag{2.35}$$

con $\eta_{12} + \eta_{21} = 1$. Dados los vectores unitarios $\hat{d}_{12}$ y $\hat{d}_{21}$, los coeficientes η_{12} y η_{21} vienen dados mediante

$$\eta_{12} = \frac{(x - w_2) \cdot \hat{d}_{12}}{\|d_{12}\|}, \qquad \eta_{12} = \frac{(x - w_1) \cdot \hat{d}_{21}}{\|d_{12}\|}. \tag{2.36}$$

Así la distancia cuadrática $\|d(t)\|^2$ de x al segmento es

$$\|d(t)\|^2 = \|x - p\|^2 = \|x(t) - w_2\|^2 - \eta_{12}^2 \cdot \|w_1 - w_2\|^2. \tag{2.37}$$

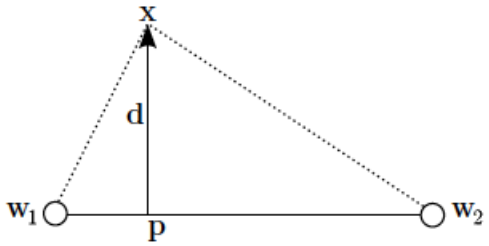


Figura 7: Se muestra la distancia entre el punto x y su proyección p en el segmento definido por los vectores w_1 y w_2.

Cabe destacar que la proyección p siempre pertenecerá al segmento, ya que los coeficientes están limitados, $\eta_{12}, \eta_{21} \in [0,1]$.

2.9.3 Regla de ajuste

Para derivar la regla de ajuste se calcula el gradiente de la medida de distancia $\|d(t)\|^2$ mostrada en la ecuación (2.37). El procedimiento es como sigue

$$\begin{aligned} \frac{\partial d^2(t)}{\partial w_1} &= \frac{\partial}{\partial w_1}(x - p)^T(x - p), \\ &= \frac{\partial p^T}{\partial w_1}(x - p), \\ &= \frac{\partial (w_2 + \eta_{12} d_{12})^T}{\partial w_1}(x - p), \\ &= \frac{\partial \eta_{12}}{\partial w_1} d_{12}^T (x - p) + \eta_{12} \frac{\partial d_{12}^T}{\partial w_1}(x - p), \end{aligned} \tag{2.38}$$

dado que d_{12} es perpendicular a $(x - p)$ y que $\frac{\partial d_{12}^T}{\partial w_1} = 1$ se tiene que

$$\frac{\partial d^2(t)}{\partial w_1} = \eta_{12}(x - p). \tag{2.39}$$

Con el gradiente obtenido se derivan las reglas de ajuste para ambos pesos asociados al segmento s

$$\begin{aligned} \Delta w_1 &= \epsilon_w \eta_{12}(x-p), \\ \Delta w_2 &= \epsilon_w \eta_{21}(x-p), \end{aligned} \tag{2.40}$$

donde $\epsilon_w \in \mathbb{R}$ corresponde a la tasa de aprendizaje. En la Figura 8 se ilustra el ajuste de un segmento frente a un punto x, el desplazamiento de los vectores que definen al segmento es ortogonal al trazo entre ellos.

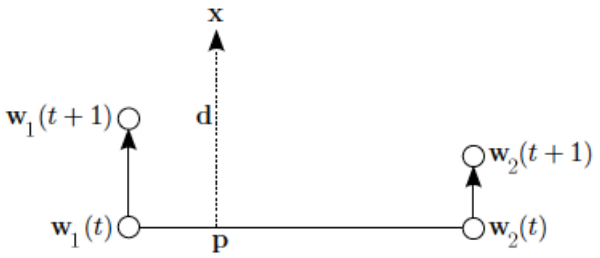


Figura 8: Los extremos del segmento se ajustan hacia x de acuerdo a la ecuación (2.40), el ajuste es ortogonal al segmento.

2.9.4 Parámetro de resorte

Como ya se ilustró en la Figura 8, los extremos de un segmento se ajustan de forma ortogonal al trazo entre ellos, esto produce que después de cada ajuste el segmento crezca y en ocasiones estos pueden escapar de los limites del espacio de entrada. Para evitar esto, se introduce un factor de resorte en la regla de ajuste con el fin de reducir la distancia entre los extremos de los segmentos cada paso. Específicamente a los nodos actualizados se les aplica

$$\begin{aligned} \Delta w_i &= \delta_s(w_j - w_i), \\ \Delta w_j &= \delta_s(w_i - w_j), \end{aligned} \tag{2.41}$$

donde $\delta_s \in \mathbb{R}$ es el parámetro de resorte.

3 Desarrollo de una red neuronal auto-organizativa con la capacidad de cuantificar espacial y temporalmente

Este trabajo busca desarrollar un algoritmo inspirado en GNG, que logre cuantificar espacial y temporalmente una serie de tiempo. En este capítulo se muestran los pasos con los cuales se desarrolla la extensión temporal de GNG. Cabe destacar que se utiliza la variante de GNG que maximiza la entropía.

Se propone usar segmentos en lugar de puntos o vectores, con el fin de cuantificar trayectorias de datos consecutivos en el tiempo. Esto requiere modificar la medida de distancia junto con modificar el ajuste, inserción y remoción de las neuronas. Se comienza con el algoritmo tradicional de GNG, luego se extiende con el fin de usar segmentos y posteriormente se cambia la regla de ajuste de los segmentos. En cada extensión se busca modificar lo menos posible el algoritmo original GNG.

A lo largo de este capítulo se utiliza un atractor artificial, que se observa en la Figura 9, con el fin de mostrar que hace cada extensión del algoritmo.

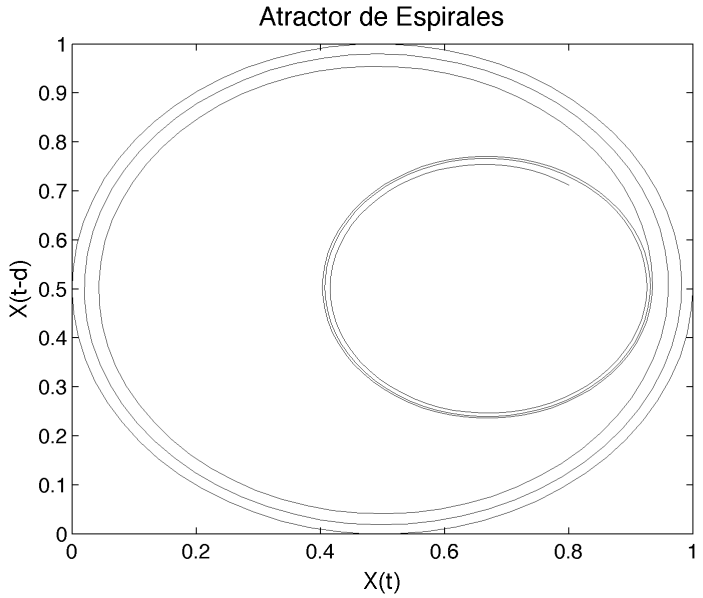


Figura 9: Atractor artificial, consistente de dos espirales conectadas y de distinto radio.

3.1 Integración de segmentos en GNG

Para utilizar segmentos en GNG se hace uso del segmento desarrollado en [23] para el algoritmo SOM. Sin embargo, es necesario añadir etapas de inicialización, inserción y conexión de segmentos.

3.1.1 Inicialización

La inicialización es similar a la de GNG. Se inicializan 2 segmentos aleatorios cada uno compuesto de 2 vectores en el espacio de entrada.

3.1.2 Inserción de segmentos

El criterio de inserción de GNG busca la neurona con más victorias y entre sus vecinos, se busca la neurona con más victorias. Posteriormente, entre estas dos se inserta una nueva neurona. En este trabajo se utiliza el mismo método, se seleccionan dos segmentos en base al mismo criterio y luego se inserta un nuevo segmento en medio de estos dos.

Sea u el segmento con mayor número de victorias y v el segmento vecino con mayor número de victorias, la inserción de un nuevo segmento r se hace como sigue

$$\begin{aligned} w_1^r &= 0.5 \cdot (w_1^u + w_1^v), \\ w_2^r &= 0.5 \cdot (w_2^u + w_2^v). \end{aligned} \tag{3.1}$$

En la Figura 10 se muestra como es la inserción de un segmento r entre dos segmentos u y v.

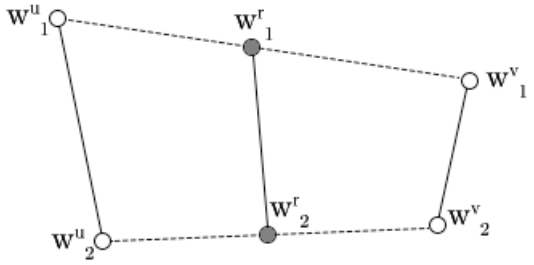


Figura 10: La inserción de un segmento r entre dos segmentos u y v. La línea punteada entre w_1^u y w_1^v se utiliza para ilustrar que w_1^r se inserta a la mitad de la distancia entre w_1^u y w_1^v, tal como se indica en la ecuación (3.1). Asimismo la línea punteada entre w_2^u y w_2^v se utiliza para ilustrar el posicionamiento de w_2^r.

3.1.3 Conexiones

Las conexiones en GNG establecen relaciones de vecindad entre las neuronas de la red, además éstas se pueden visualizar como en la Figura 5, en donde se muestran conexiones entre neuronas de la red GNG. Al introducir la utilización segmentos, las conexiones siguen estableciendo relaciones de vecindad. Sin embargo, con fines de visualización, las conexiones entre segmentos se dibujan desde los centros de los segmentos, conectados tal como se muestra en la Figura 11. Esto entrega información visual sobre la estructura espacial de la red construida.

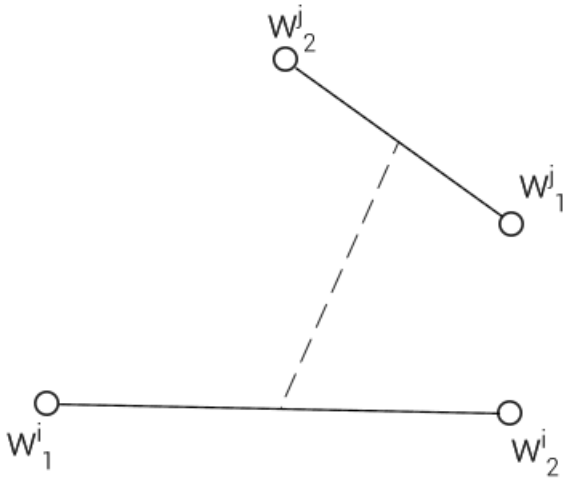


Figura 11: Un segmento *i* conectado con un segmento *j*. La conexión se dibuja entre los centros de los segmentos, en este caso la conexión corresponde a la línea punteada.

3.1.4 GNG extendido al uso de segmentos

El algoritmo de GNG extendido al uso de segmentos es el siguiente:

1. Comenzar con 2 segmentos desconectados, $A = \{1, 2\}$, inicializados aleatoriamente en el espacio de entrada $\mathbb{R}^d$. Además inicializar el contador de victorias de ambos segmentos en cero, $wcount_1 = wcount_2 = 0$.
2. Presentar un vector de entrada, $x(t)$, a la red GNG extendida al uso de segmentos.
3. Encontrar la BMU[2] de $x(t)$, I_t y el segundo segmento más cercano J_t mediante

$$I_t = \underset{i \in A}{argmin} \|d_i(t)\|^2,$$
$$J_t = \underset{i \in A \setminus I_t}{argmin} \|d_i(t)\|^2, \tag{3.2}$$

   utilizando $\|d_i(t)\|^2$ como la distancia cuadrática presentada en la ecuación (2.37) y A el conjunto de índices de los segmentos de la red GNG extendida.
4. Incrementar en uno la edad de todas las conexiones que salen de I_t.
5. Actualizar el contador de victorias de I_t mediante

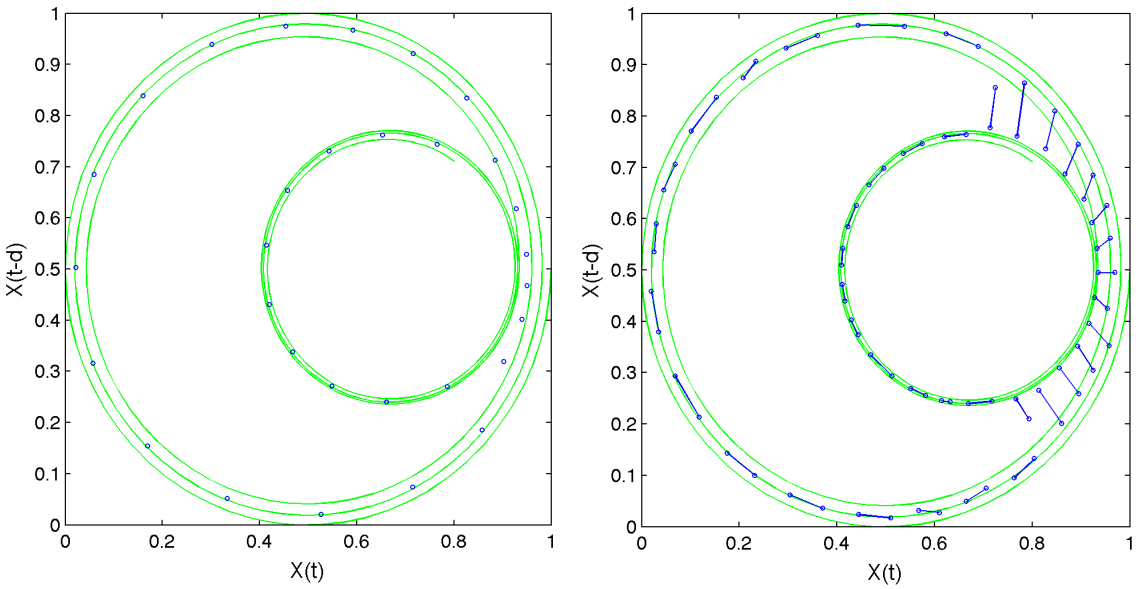
$$wcount_{I_t} = wcount_{I_t} + 1. \tag{3.3}$$

6. Ajustar la BMU utilizando la ecuación (2.40). Además ajustar los vecinos de la BMU utilizando $\epsilon_n \in \mathbb{R}$ en lugar de ϵ_w en la ecuación (2.40), donde ϵ_n corresponde a la tasa de aprendizaje de los vecinos de la BMU. Aplicar la ecuación (2.41) sobre todos los segmentos ajustados en esta etapa.
7. Si no existe una conexión entre I_t y J_t crearla. Fijar la edad de la conexión entre ambos segmentos en cero.

[2] Se mantiene la nomenclatura de GNG, la BMU en el caso de segmentos corresponde al segmento más cercano.

8. Remover las conexiones cuya edad sea mayor a la edad máxima a_{max}. Eliminar los segmentos sin conexiones.
9. Si la iteración actual t es un entero múltiplo del parámetro λ, entonces insertar un nuevo segmento a la red. La inserción de un nuevo segmento es según:
    a. Buscar en la red GNG extendida, el segmento u con el mayor número de victorias.
    b. Entre los vecinos de u, encontrar aquel segmento v con el mayor número de victorias.
    c. Insertar un segmento r entre u y v según la ecuación (3.1).
    d. Crear conexiones entre u y r, entre r y v; remover la conexión entre los segmentos u y v.
    e. Reducir el contador de victorias de los segmentos u y v multiplicado sus contadores de victorias por una constante α, fijar el contador de victorias del segmento r como el nuevo valor del número de victorias de u.
10. Reducir el contador de victorias de todos los segmentos multiplicando sus contadores de victorias por una constante β.
11. Incrementar el número de la iteración ($t = t + 1$) y volver al punto 2 si no se ha alcanzado algún criterio de término (e.g., número máximo de segmentos).

En la Figura 12 se muestra la cuantificación de la red en el atractor artificial. A la izquierda GNG y a la derecha GNG extendida con segmentos. Notar la dirección de los segmentos, buscando cubrir la mayor cantidad de muestras en su vecindad.



(a) (b)

Figura 12: Con fines ilustrativos a) Cuantificación entregada por GNG las neuronas están dibujadas como círculos. b) Cuantificación entregada por GNG extendido al uso de segmentos dibujados como trazos azules. No se muestran las conexiones.

3.2 Modificaciones temporales

Hasta el momento, la propuesta de GNG extendida al uso de segmentos realiza un ajuste de los segmentos basados en la distancia espacial, sin considerar el tiempo asociado a los datos. En esta sección se realizan 2 modificaciones para incluir el tiempo en el algoritmo propuesto. Primero se modifica el ajuste de los segmentos y luego se realizan conexiones temporales entre los segmentos.

3.2.1 Ajuste temporal

Con el fin de representar trayectorias de datos consecutivos en el tiempo, se propone que los segmentos se ajusten hacia un trazo de curva entre el dato actual $x(t)$ y un dato de n muestras anteriores $x(t-n)$. En la Figura 13 se muestra el ajuste de un segmento definido por $w_1, w_2 \in \mathbb{R}^d$ hacia un trazo $[x(t-n),\ x(t)]$.

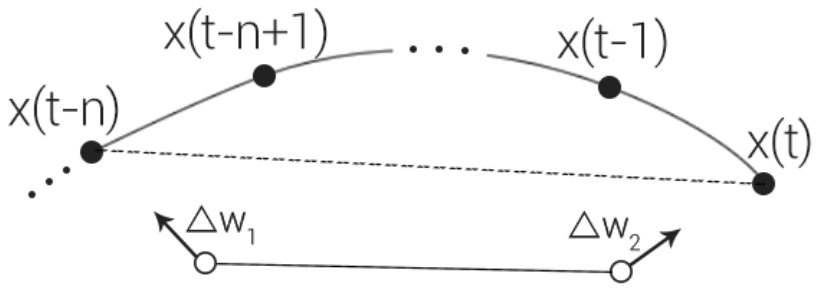


Figura 13: Ajuste de un segmento definido por $w_1, w_2 \in \mathbb{R}^d$ hacia el trazo de curva compuesto por $x(t-n)$ y $x(t)$, e ilustrado como una línea punteada.

Los extremos del segmento w_1 y w_2 se ajustan hacia los extremos del trazo $x(t-n)$, $x(t)$ según

$$\begin{aligned} \Delta w_1 &= \epsilon_w(t)(x(t) - w_1), \\ \Delta w_2 &= \epsilon_w(t)(x(t-n) - w_2), \end{aligned} \tag{3.4}$$

en donde n es el retardo que determina el largo del trazo $[x(t-n),\ x(t)]$, este valor se determina de manera adaptiva según el error acumulado entre la sección de la curva de datos $\{x(t-n), \ldots, x(t)\}$ y el trazo recto $[x(t-n),\ x(t)]$. En la Figura 14 se muestran los errores $e_j(t,n) \in \mathbb{R}$ correspondientes a la distancia cuadrática entre el punto $x(t-j)$ y el trazo $[x(t-n),\ x(t)]$. El error $e_j(t,n)$ se determina a partir de

$$e_j(t,n) = \|x(t-j) - x(t)\|^2 - \left\|\left(x(t-j) - x(t)\right) \cdot \hat{d}_{t-n,t}\right\|^2, \tag{3.5}$$

en donde $\hat{d}_{t-n,t}$ es el vector unitario de $d_{t-n,t} = x(t-n) - x(t)$.

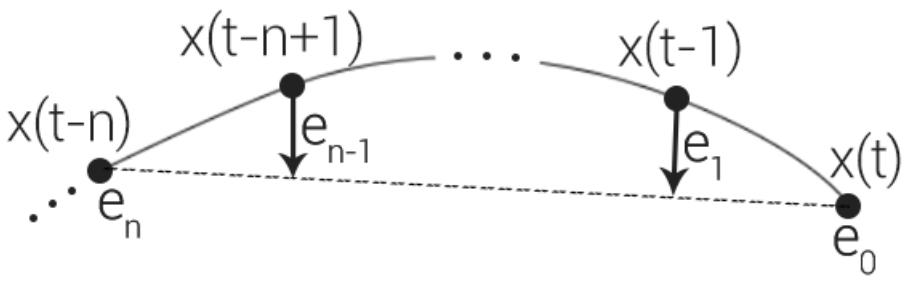


Figura 14: Una sección de la curva de datos, compuesta por los puntos $\{x(t), \dots, x(t-n)\}$. Se muestran los errores e_j como la distancia cuadrática entre los puntos $x(t-j)$ y su proyección sobre el trazo recto entre $x(t)$ y $x(t-n)$. La suma de los e_j entrega el error bajo la curva considerada.

Así, utilizando la ecuación (3.5) y notando que $e_0 = e_n = 0$, el error total bajo la sección de curva compuesta por los datos $\{x(t-n), \dots, x(t)\}$, $E(t,n)$, está dado por

$$E(t,n) = \sum_{j=1}^{n-1} e_j(t,n). \tag{3.6}$$

Con el error acumulado $E(t,n)$ es posible determinar n a partir de

$$n = \underset{n \in \mathbb{N}}{argmin} |E(t,n) - E_{max}|^2, \tag{3.7}$$

en donde $E_{max} \in \mathbb{R}$ es el máximo error permitido bajo una sección de curva, es un parámetro de la red, ajustado por el usuario.

Esta nueva manera de adaptar los trazos evita el uso del factor de resorte. En la Figura 15 se puede ver el resultado de esta propuesta de adaptación de segmentos para el caso del atractor artificial. Todos los segmentos ahora se alinean temporalmente con los datos.

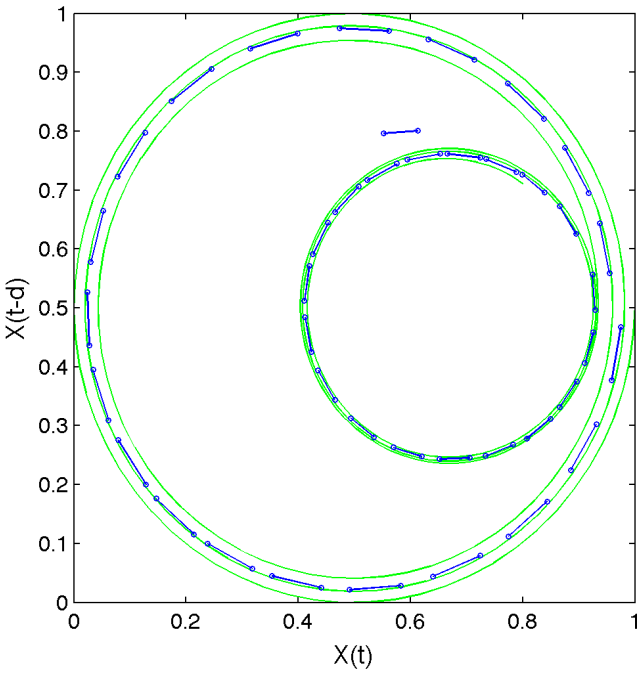


Figura 15: Red GNG extendida al uso de segmentos junto con el nuevo ajuste temporal. Se observa como los segmentos (trazos azules) se alinean a la curva. Por limpieza se omiten las conexiones.

Es importante notar que si $n = 0$, los segmentos se convierten en puntos y su comportamiento es idéntico al de una neurona de GNG.

3.2.2 Conexiones temporales

Se introduce el uso de conexiones temporales para mantener un registro de la relación temporal que existe entre los segmentos de la red. Las conexiones temporales se crean entre segmentos ganadores consecutivos en el tiempo, es decir se conecta el último segmento ganador con el actual segmento ganador. La inclusión de conexiones temporales tiene como propósito identificar el comportamiento temporal de la serie. Así, en relación a las conexiones, la red GNG extendida al uso de segmentos tendrá 2 conjuntos de conexiones:

- **Conexiones espaciales:** utilizadas para el ajuste de segmentos vecinos, como también para la inserción de nuevos segmentos.
- **Conexiones temporales:** utilizadas para identificar el comportamiento temporal de la serie.

Por último, una conexión temporal que une a 2 segmentos se dibuja entre los 2 extremos más cercanos entre los segmentos.

3.3 GSG

Como resultado de las extensiones de GNG expuestas durante este capítulo se presenta, GSG (*Growing Segment Gas*) el primer algoritmo de cuantificación espacio temporal propuesto en este trabajo. El algoritmo GSG es como sigue:

1. Comenzar con 2 segmentos, $A = \{1, 2\}$, desconectados espacial y temporalmente. Los segmentos se inicializan aleatoriamente en el espacio de entrada y son definidos por los vectores $w_1^1, w_2^1 \in \mathbb{R}^d$ para el segmento 1 y $w_1^2, w_2^2 \in \mathbb{R}^d$ para el segmento 2. Además inicializar el contador de victorias de ambos segmentos en cero, $wcount_1 = wcount_2 = 0$.
2. Presentar un vector de entrada, $x(t)$, a la red GSG.
3. Encontrar la BMU de $x(t)$, I_t y el segundo segmento más cercano J_t mediante

$$I_t = \underset{i \in A}{argmin} \|d_i(t)\|^2,$$
$$J_t = \underset{i \in A \setminus I_t}{argmin} \|d_i(t)\|^2, \tag{3.8}$$

   utilizando $\|d_i(t)\|^2$ como la distancia cuadrática presentada en la ecuación (2.37) y A el conjunto de índices de los segmentos de la red GSG.
4. Incrementar en uno la edad de todas las conexiones que salen de I_t.
5. Actualizar el contador de victorias de I_t mediante

$$wcount_{I_t} = wcount_{I_t} + 1. \tag{3.9}$$

6. Encontrar n utilizando la ecuación (3.7) y ajustar la BMU utilizando la ecuación (3.4). Además, ajustar los vecinos de la BMU utilizando $\epsilon_n \in \mathbb{R}$ en lugar de ϵ_w en la ecuación (3.4), donde ϵ_n corresponde a la tasa de aprendizaje de los vecinos de la BMU.
7. Si no existe una conexión espacial entre I_t y J_t crearla. Fijar la edad de la conexión entre ambos segmentos en cero.
8. Si la BMU en $t - 1$, I_{t-1}, es distinta a la BMU actual, I_t, conectarlas con una conexión temporal.
9. Remover las conexiones espaciales cuya edad sea mayor a la edad máxima a_{max}. Eliminar los segmentos sin conexiones espaciales.
10. Si la iteración actual t es un entero múltiplo del parámetro λ, entonces insertar un nuevo segmento a la red. La inserción de un nuevo segmento es según:
    a. Buscar en la red GSG, el segmento u con el mayor número de victorias.
    b. Entre los vecinos de u, encontrar aquel segmento v con el mayor número de victorias.
    c. Insertar un segmento r entre u y v según (3.1).
    d. Crear conexiones espaciales entre u y r, entre r y v; remover las conexiones entre los segmentos u y v.
    e. Reducir el contador de victorias de los segmentos u y v multiplicado sus contadores de victorias por una constante α, fijar el contador de victorias del segmento r como el nuevo valor del número de victorias de u.
11. Reducir el contador de victorias de todos los segmentos multiplicando sus contadores de victorias por una constante β.
12. Incrementar el número de la iteración ($t = t + 1$) y volver al punto 2 si no se ha alcanzado algún criterio de término (e.g., número máximo de segmentos).

3.4 MGSG

Incorporando un modelo de contexto en GSG se obtiene MGSG (*Merge Growing Segment Gas*), el segundo algoritmo propuesto en este trabajo. Se utiliza el mismo modelo de contextos utilizado en MSOM, MNG y MGNG, sobre GSG.

Para esto, la medida de distancia cuadrática de la ecuación (2.37) se reemplaza por

$$\|d(t)\|^2 = (1-\alpha_m)\|x-p\|^2 + \alpha_m\|c(t)-c_i\|^2, \tag{3.10}$$

donde $c(t) \in \mathbb{R}^d$ es el descriptor de contexto de la red MGSG y $c_i \in \mathbb{R}^d$ el contexto del segmento i. Se definen las reglas de ajuste para $c(t)$ y c_i mediante

$$\Delta c_i = \epsilon_w \cdot (c(t) - c_i), \tag{3.11}$$

$$c(t+1) = (1-\beta_m)\cdot \widehat{w}^{I_t} + \beta_m \cdot c_{I_t}, \tag{3.12}$$

con $\epsilon_w \in \mathbb{R}$ la tasa de aprendizaje de la BMU, I_t el índice de la BMU y $\widehat{w}^i \in \mathbb{R}^d$ el centro del segmento i. Los vectores que definen al segmento se ajustan de igual manera que en GSG.

El algoritmo MGSG es como sigue:

1. Comenzar con 2 segmentos, $A = \{1, 2\}$, desconectados espacial y temporalmente. Los segmentos se inicializan aleatoriamente en el espacio de entrada y son definidos por los vectores y contextos $w_1^1, w_2^1, c_1 \in \mathbb{R}^d$ para el segmento 1 y por $w_1^2, w_2^2, c_2 \in \mathbb{R}^d$ para el segmento 2. Fijar el contador de victorias de ambos segmentos en cero, $wcount_1 = wcount_2 = 0$. Inicializar en cero el descriptor de contexto de la red MGSG $c(t) = 0$.
2. Presentar un vector de entrada, $x(t)$, a la red MGSG.
3. Encontrar la BMU de $x(t)$, I_t y el segundo segmento más cercano J_t mediante

$$\begin{aligned} I_t &= \underset{i\in A}{argmin}\|d_i(t)\|^2, \\ J_t &= \underset{i\in A\setminus I_t}{argmin}\|d_i(t)\|^2, \end{aligned} \tag{3.13}$$

   utilizando $\|d_i(t)\|^2$ como la distancia cuadrática presentada en la ecuación (3.10) y A el conjunto de índices de los segmentos de la red MGSG.
4. Incrementar en uno la edad de todas las conexiones que salen de I_t.
5. Actualizar el contador de victorias de I_t según

$$wcount_{I_t} = wcount_{I_t} + 1. \tag{3.14}$$

6. Encontrar n utilizando la ecuación (3.7), ajustar los vectores de la BMU utilizando la ecuación (3.4) y ajustar el contexto de la BMU utilizando (3.11). Además,

ajustar los vecinos de la BMU utilizando $\epsilon_n \in \mathbb{R}$ en lugar de ϵ_w en la ecuación (3.4) y (3.11), donde ϵ_n corresponde a la tasa de aprendizaje de los vecinos de la BMU.

7. Actualizar el descriptor de contexto utilizando la ecuación (3.16).
8. Si no existe una conexión espacial entre I_t y J_t crearla. Fijar la edad de la conexión entre ambos segmentos en cero.
9. Si la BMU en $t-1$, I_{t-1}, es distinta a la BMU actual, I_t, conectarlas con una conexión temporal.
10. Remover las conexiones espaciales cuya edad sea mayor a la edad máxima a_{max}. Eliminar los segmentos sin conexiones espaciales.
11. Si la iteración actual t es un entero múltiplo del parámetro λ, entonces insertar un nuevo segmento a la red. La inserción de un nuevo segmento es según:
    a. Buscar en la red MGSG, el segmento u con el mayor número de victorias.
    b. Entre los vecinos de u, encontrar aquel segmento v con el mayor número de victorias.
    c. Insertar un segmento r entre u y v según

$$w_1^r = 0.5w_1^u + 0.5w_1^v,$$
$$w_2^r = 0.5w_2^u + 0.5w_2^v, \tag{3.15}$$
$$c_r = 0.5c_u + 0.5c_v.$$

    d. Crear conexiones entre u y r, entre r y v; remover la conexión entre los segmentos u y v.
    e. Reducir el contador de victorias de los segmentos u y v multiplicado sus contadores de victorias por una constante α, fijar el contador de victorias del segmento r como el nuevo valor del número de victorias de u.
12. Reducir el contador de victorias de todos los segmentos multiplicando sus contadores de victorias por una constante β.
13. Incrementar el número de la iteración ($t = t + 1$) y volver al punto 2 si no se ha alcanzado algún criterio de término (e.g., número máximo de segmentos).

3.5 Metodología

Para evaluar el desempeño de los algoritmos propuestos, GSG y MGSG, estos se comparan con otros dos algoritmos: GNG y MGNG. Los algoritmos son evaluados en 4 bases de datos distintas: Mackey-Glass, Lorenz, Rössler y Laser NH3-FIR.

Se mide la cuantificación espacio temporal de los algoritmos sobre las bases de datos utilizando el TQE con retardos de $\tau = 0, \dots, 50$. El desempeño de cada algoritmo se mide como el TQE promedio de 10 ejecuciones independientes en cada una de las bases de datos mencionadas anteriormente. Se realiza una evaluación estadística utilizando la Prueba t de Stundent con un 5% de tolerancia.

Los algoritmos GNG, MGNG, GSG y MGSG se implementaron en el software MATLAB.

3.5.1 Mackey-Glass

La serie de tiempo Mackey-Glass [24] está definida por la ecuación diferencial

$$\frac{\partial x}{\partial \tau} = bx(\tau) + \frac{a\, x(\tau - d)}{1 + x(\tau - d)^{10}}. \tag{3.16}$$

Se utiliza como base de datos *benchmark* sobre la cual se prueban varios algoritmos orientados al estudio de series de tiempo. Utilizando $d = 17$, $a = 0.2$ y $b = -0.1$ se obtiene la serie mostrada en la Figura 16. En la Figura 17 se muestra el atractor de la serie en el espacio de estado.

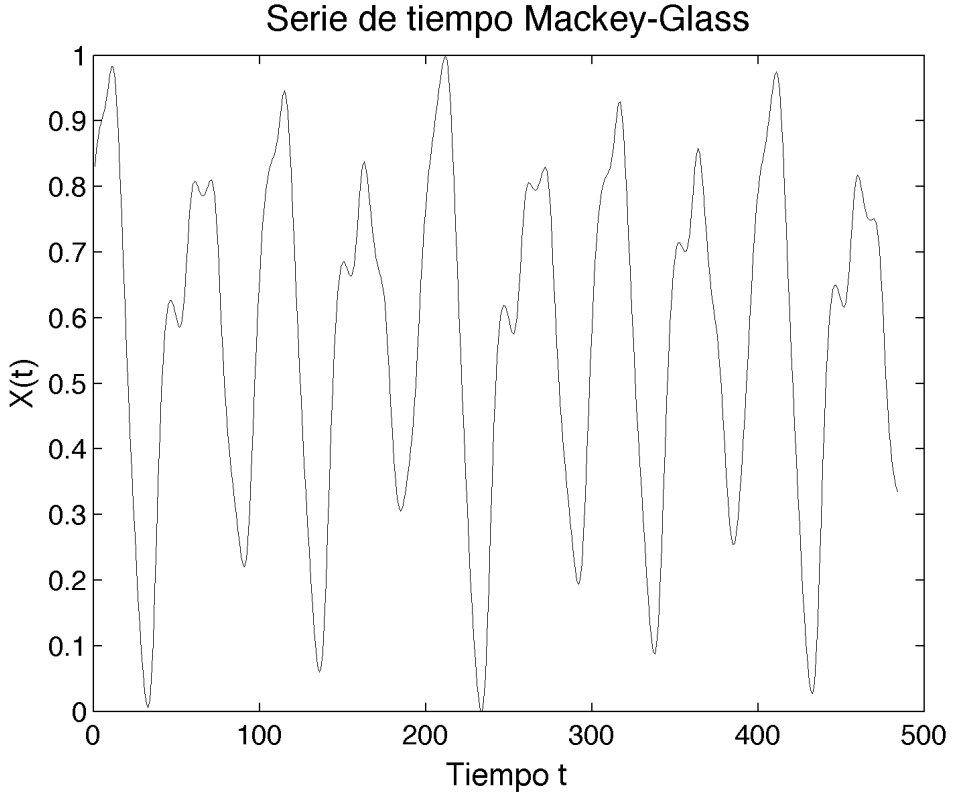


Figura 16: Serie de tiempo Mackey-Glass, con $d = 17$, $a = 0.2$ y $b = -0.1$ en la ecuación (3.16).

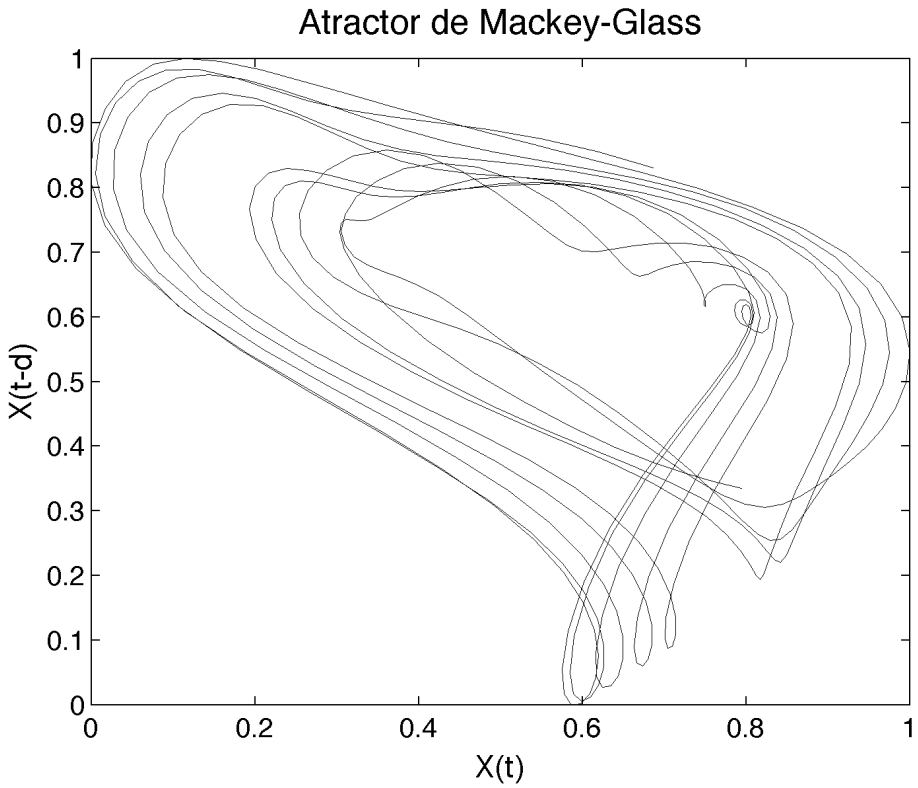


Figura 17: Atractor de la serie de tiempo Mackey-Glass mostrada en la Figura 16: Serie de tiempo Mackey-Glass, con $d = 17$, $a = 0.2$ y $b = -0.1$ en la ecuación (3.16). con un retardo de $d = 17$.

3.5.2 Lorenz

El atractor de Lorenz es un modelo físico de la convección térmica [25], corresponde a un atractor caótico tridimensional, cuyas variables de estado están dadas por la solución del siguiente sistema de ecuaciones

$$\begin{aligned} \dot{x} &= -\sigma x + \sigma y, \\ \dot{y} &= Rx - y - xz, \\ \dot{z} &= -Bz + xy. \end{aligned} \tag{3.17}$$

En este trabajo se utilizan los siguientes parámetros para el atractor de Lorenz [25]

$$\sigma = 10, \qquad R = 28, \qquad B = \frac{8}{3},$$

con condiciones iniciales

$$x_0 = -1, \qquad y_0 = 0, \qquad z_0 = 0.$$

Sobre la solución de (3.17) se utiliza una proyección PCA unidimensional la cual se observa en la Figura 18. Luego, utilizando un retardo de $d = 16$ sobre la serie proyectada se obtiene el atractor de la Figura 19.

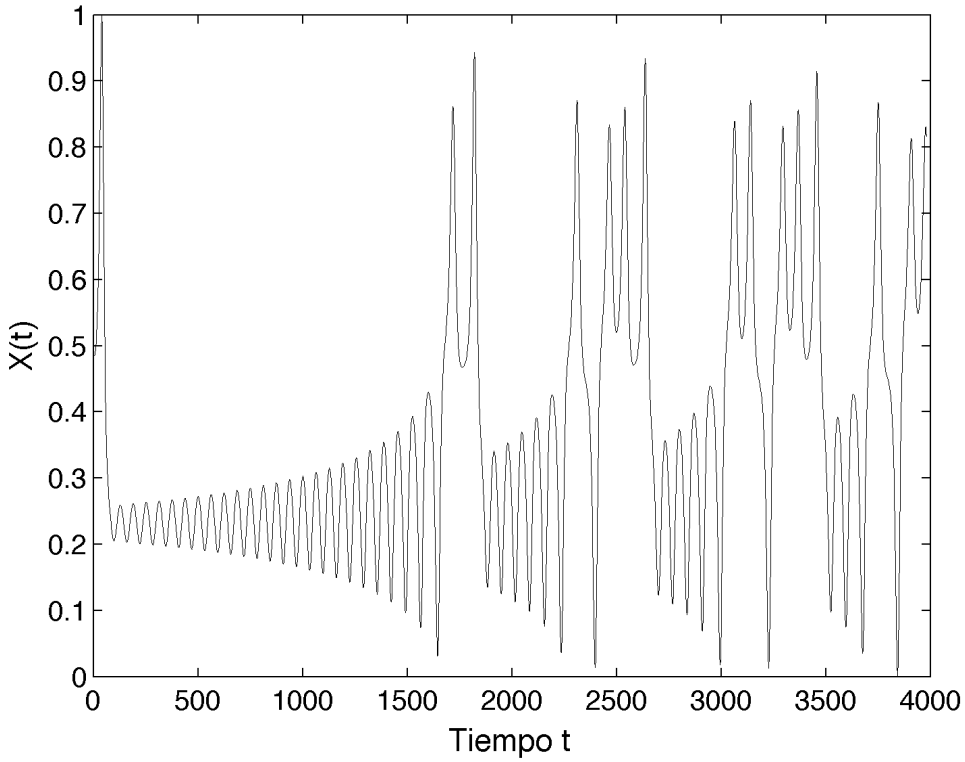


Figura 18: Proyección unidimensional del atractor de Lorenz.

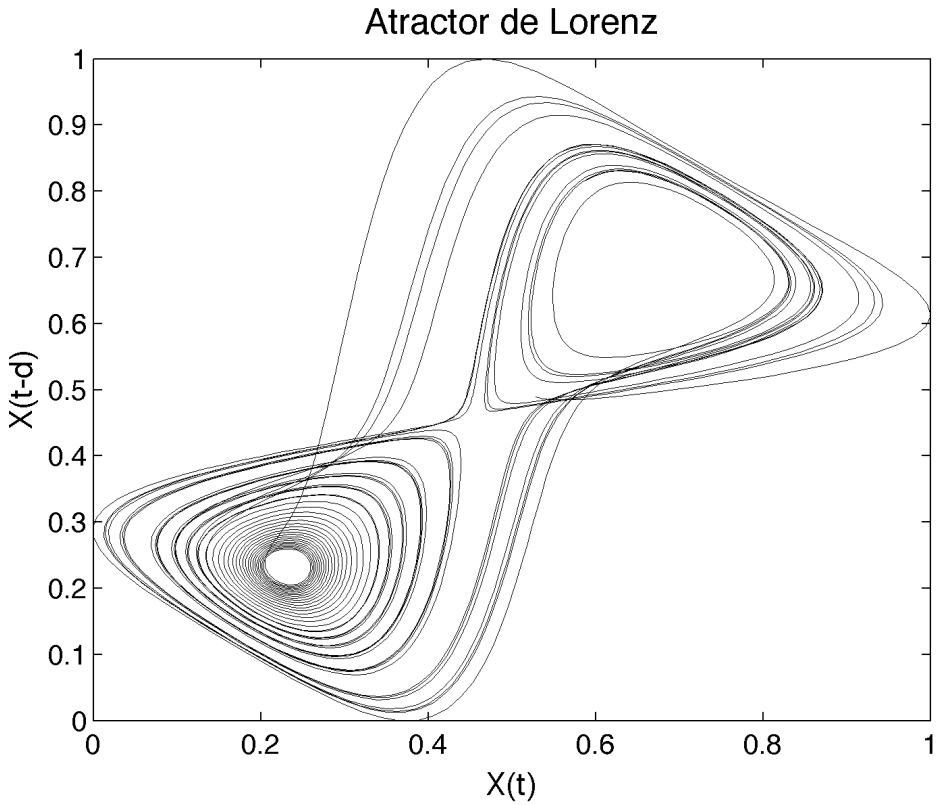


Figura 19: Atractor de Lorenz con un retardo de $d = 16$.

3.5.3 Rössler

El atractor de Rössler [26] se obtiene como la solución del siguiente sistema de ecuaciones diferenciales

$$\begin{aligned} \dot{x} &= -y - z, \\ \dot{y} &= x + ay, \\ \dot{z} &= b + xz - cz. \end{aligned} \tag{3.18}$$

Se resuelven las ecuaciones de (3.18) utilizando los parámetros $(a, b, c) = (0{,}2; 0{,}2; 5{,}7)$ y las condiciones iniciales $(x_0, y_0, z_0) = (-1{,}0{,}0)$. Luego se realiza una proyección PCA sobre $\mathbb{R}$ obteniendo la serie mostrada en la Figura 20. En la Figura 21 se muestra el atractor de la serie proyectada con un retardo de $d = 33$.

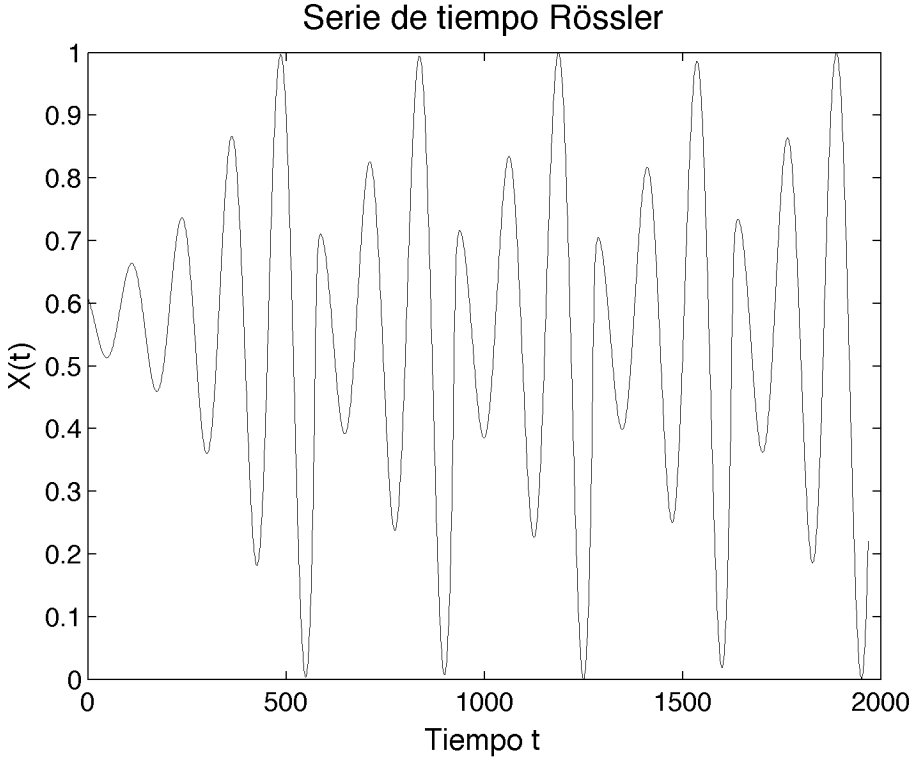


Figura 20: Serie de tiempo de Rössler.

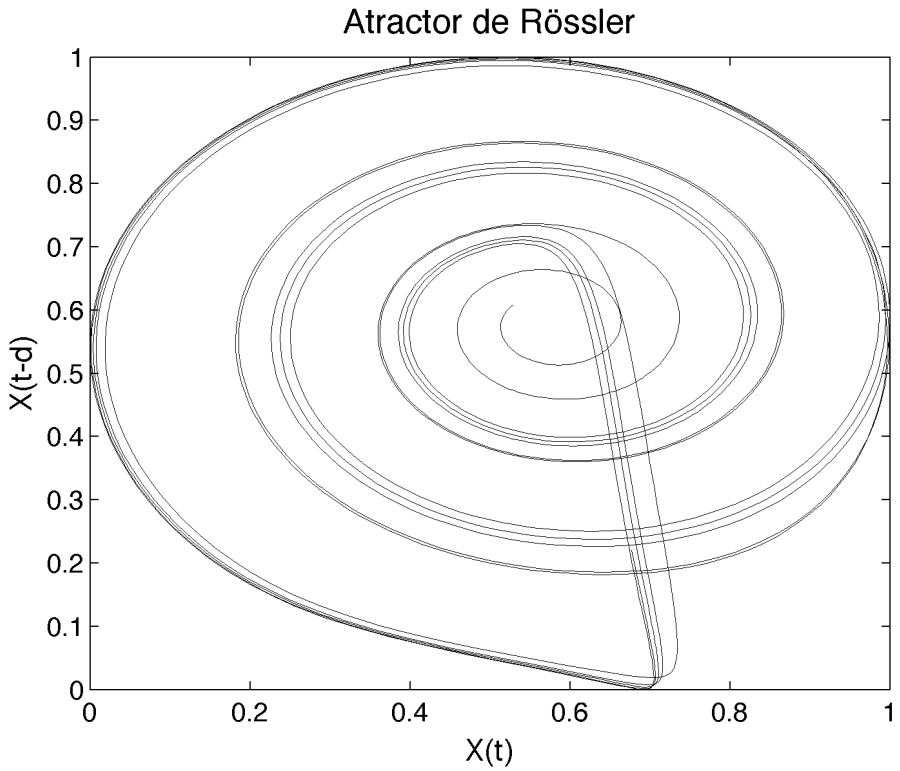


Figura 21: Atractor de Rössler con un retardo de $d = 33$.

3.5.4 Laser NH3-FIR

La serie de tiempo Laser NH3-FIR consiste en una secuencia de enteros de 8-bits. En [27] se ha demostrado que la pulsaciones caóticas del laser, siguen aproximadamente el modelo teórico de Lorenz. En la Figura 22 se muestra el atractor de la serie de tiempo Laser NH3-FIR mostrada en la Figura 23.

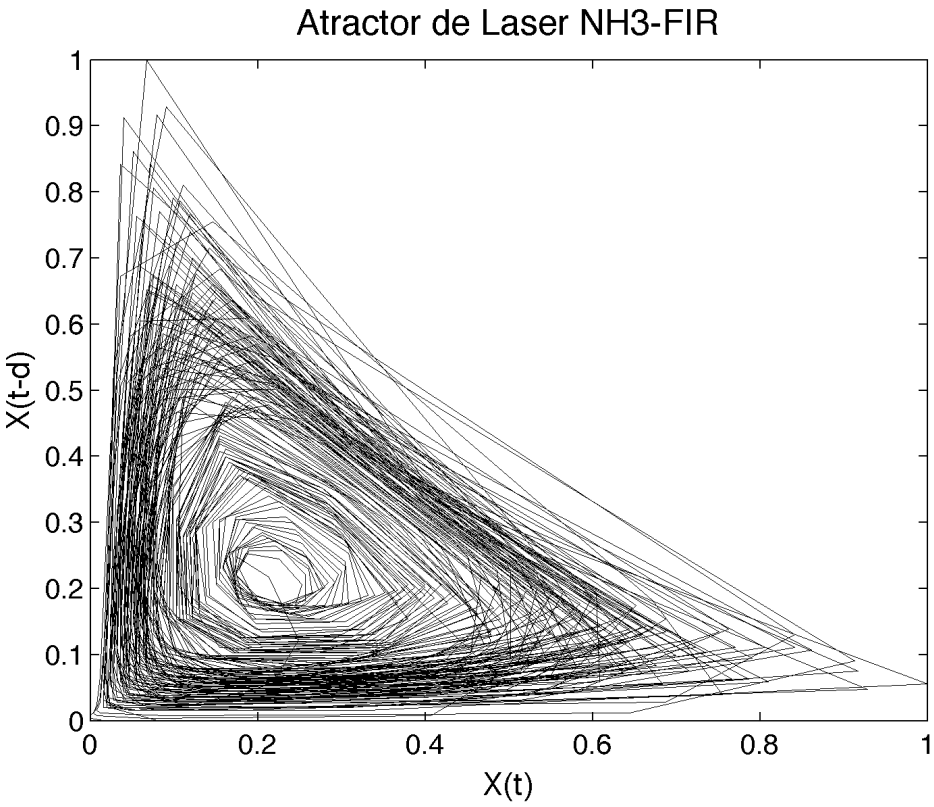


Figura 22: Atractor de la serie de tiempo Laser NH3-FIR con un retardo $d = 3$.

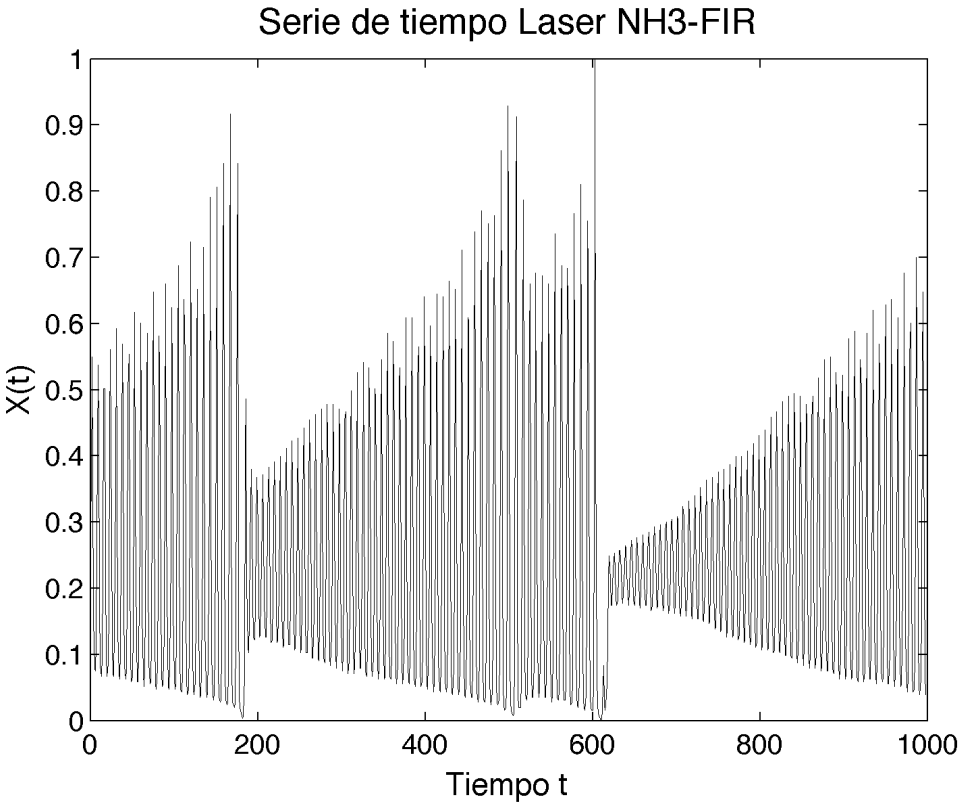


Figura 23: Serie de tiempo de la base de datos Laser NH3-FIR.

En la Tabla 3.1 se hace un resumen de las bases de datos a utilizar.

Base de Datos	Datos	Retardo	Características
Mackey-Glass	484	18	El atractor 2D tiene curvas de datos que se superponen unas con otras.
Lorenz	3981	17	Atractor con zonas muy densas.
Rössler	1968	33	El atractor 2D tiene curvas de datos que se superponen unas con otras.
Laser NH3-FIR	998	3	Atractor con muy pocas muestras, saltos abruptos y con curvas que se superponen unas con otras en 2D.

Tabla 3.1: Resumen de las cuatro bases de datos que se utilizan en este trabajo.

4 Resultados

Los parámetros utilizados por cada algoritmo para cada base de datos se encuentran detallados en el Anexo I, mientras que en el Anexo II se muestran los resultados gráficos de los algoritmos GNG, MGNG, GSG y MGSG sobre los atractores de las series utilizadas.
Para todas las redes el número máximo de neuronas o segmentos a utilizar es igual al 15% de las muestras de la serie a cuantificar.
Cabe destacar que GNG y GSG no hacen uso de contextos, lo cual es una desventaja frente a los métodos MGNG y MGSG. Esto queda en evidencia en los resultados presentados.
Por último es importante tener en consideración que el TQE en $\tau = 0$, corresponde al error de cuantificación espacial y es cercano pero no igual a cero.

4.1 Rössler

Se fija el máximo número de neuronas o segmentos utilizados en esta serie en 295. En la Figura 24 se muestra el desempeño de los algoritmos GNG, MGNG, GSG y MGSG de acuerdo al TQE en retardos desde 0 hasta 50.

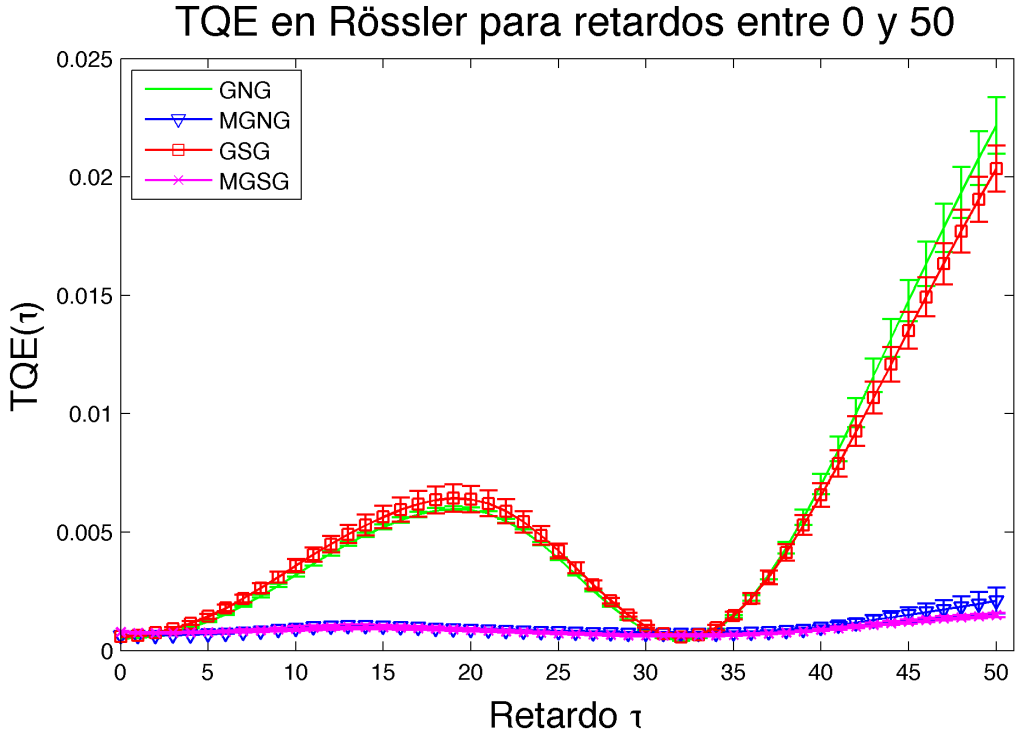


Figura 24: TQE de GNG, MGNG, GSG y MGSG en Rössler, para retardos de $\tau = 0 \ldots 50$.

Como era de esperar GNG y GSG presentan un peor desempeño que MGNG y MGSG, al no hacer uso de contextos. Entre GNG y GSG hay leves diferencias en términos del TQE (Ver Anexo III). A pesar de esto, se destaca la mejor visualización que ofrece GSG por sobre GNG en la Figura 25, donde se muestran las redes resultantes y el atractor de Rössler. Por un lado los segmentos, a diferencia de las neuronas, entregan información sobre

la dirección de las trayectorias de los datos. Por otro lado, las conexiones temporales de GSG, a diferencia de las conexiones espaciales de GNG, representan de mejor manera la dinámica de la serie, un análisis más detallado de este aspecto se realiza en la sección 4.1.1.

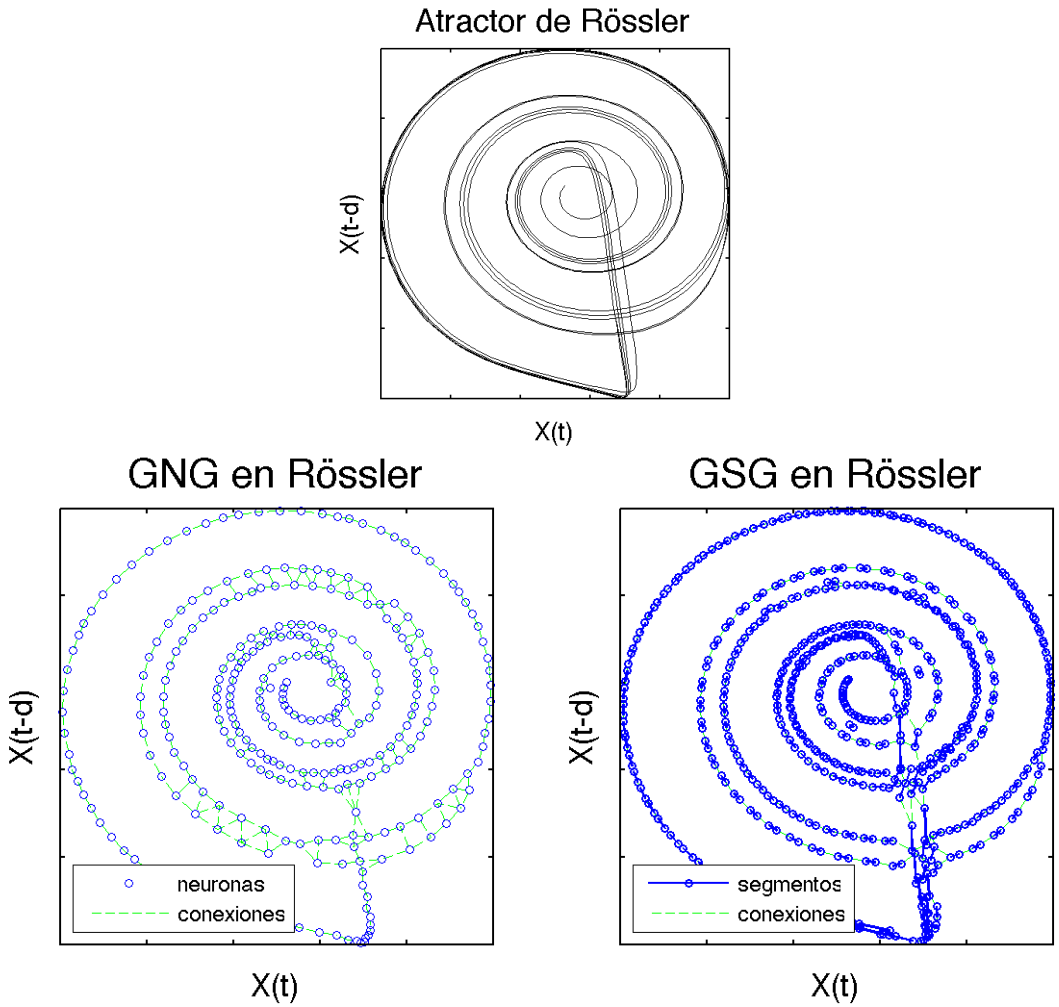


Figura 25: Redes GNG y GSG resultantes en el atractor de Rössler (mostrado arriba). Abajo a la izquierda GNG con sus neuronas y conexiones espaciales. Abajo a la derecha GSG con sus segmentos y conexiones temporales.

Siguiendo con las observaciones sobre el TQE medido en la Figura 24, se aprecia que los mejores algoritmos de acuerdo a esta métrica son MGNG y MGSG, la diferencia entre estos 2 algoritmos se comienza a notar con retardos altos cercanos a 50. En aquella zona MGSG muestra un desempeño estadísticamente significativo[3] por sobre todos los algoritmos evaluados. Además MGSG presenta una ventaja en términos de visualización, incluso por sobre GSG ya que realiza una cuantificación espacio temporal mucho más fina. Las redes resultantes para MGNG y MGSG se muestran en la Figura 26.

[3] Para los intervalos de retardos 24-30, 34-50 muestra ventajas sobre todos los algoritmos. Ver Anexo III.

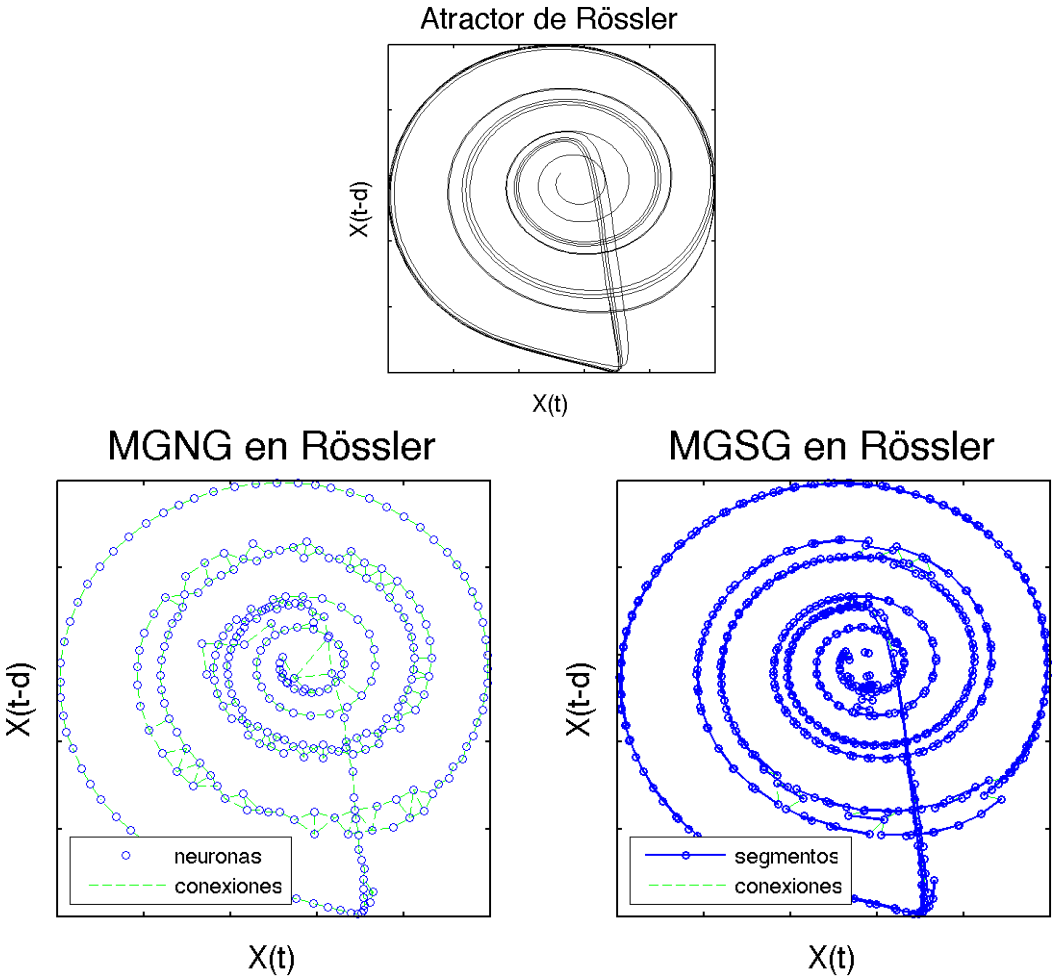


Figura 26: Redes MGNG y MGSG resultantes en el atractor de Rössler (mostrado arriba). Abajo a la izquierda MGNG con sus neuronas y conexiones espaciales. Abajo a la derecha MGSG con sus segmentos y conexiones temporales.

4.1.1 Análisis de la cuantificación espacio temporal de MGSG sobre la base de datos Rössler

MGSG en Rössler muestra la mejor cuantificación espacio temporal en términos del TQE, además muestra una red cuyas conexiones temporales pueden ser útiles en trabajos futuros que busquen desarrollar predictores en base a la red MGSG.

El análisis se realiza utilizando la red MGSG ya entrenada sobre la base de datos Rössler. Se muestra como se activa la red entrenada frente a una muestra en $t = 565$ y $t = 572$, se identifica el segmento activado, las muestras asociadas a ese segmento, los segmentos que son vecinos temporales del segmento activado y por último las muestras asociadas a estos segmentos vecinos. Los tiempos seleccionados para realizar este análisis son elegidos en una zona compleja del atractor de Rössler, precisamente donde hay cruce de curvas en el atractor de dos dimensiones. El cruce de curvas es complejo y se debe cuantificar de manera correcta, pues a pesar de poseer puntos cercanos espacialmente, los valores pasados y futuros siguen dinámicas muy distintas.

En la Figura 27 se muestra la red MGSG entrenada sobre el atractor de Rössler a la cual se le presenta el dato $x(t = 565)$ denotado por una X. El segmento activado y sus vecinos temporales se dibujan en morado.

El efecto que tiene esto en la serie de tiempo se ilustra en la Figura 28, en donde se muestran:

- La serie de tiempo del atractor de Rössler.
- El punto $x(t = 565)$, ilustrado con una x.
- Los datos cuantificados por el segmento activado por $x(t = 565)$, dibujados como círculos azules.
- Los datos cuantificados por los vecinos temporales del segmento activado por $x(t = 565)$, dibujados como círculos morados.

En la Figura 28 se nota que los datos cuantificados por el segmento activado (como también los datos de los segmentos que son vecinos temporales) poseen la misma dinámica. Todos estos datos vienen después de los ciclos de mayor amplitud de la serie y también poseen valores futuros bastante parecidos. Esto es esencial en escenarios de predicción, pues se puede combinar la información presente para proyectar valores futuros de datos.

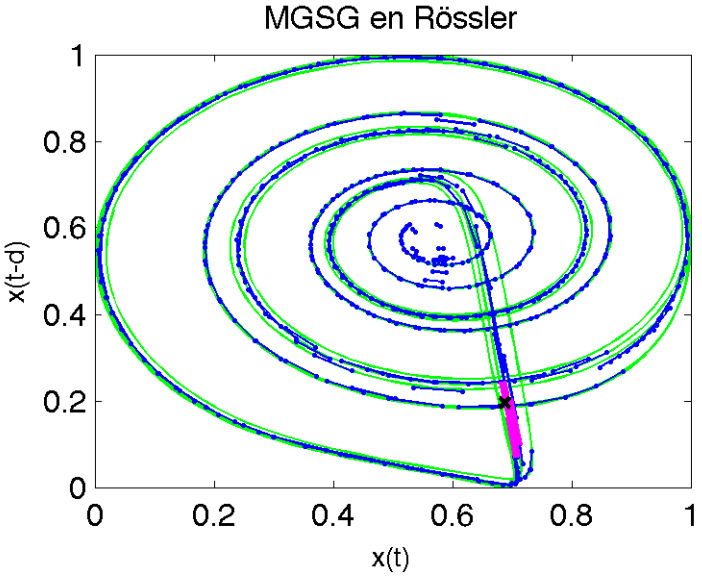


Figura 27: Red MGSG ya entrenada sobre Rössler, el dato $x(t = 565)$ se muestra con una X y el segmento activado, junto con sus vecinos temporales se ilustran en morado.

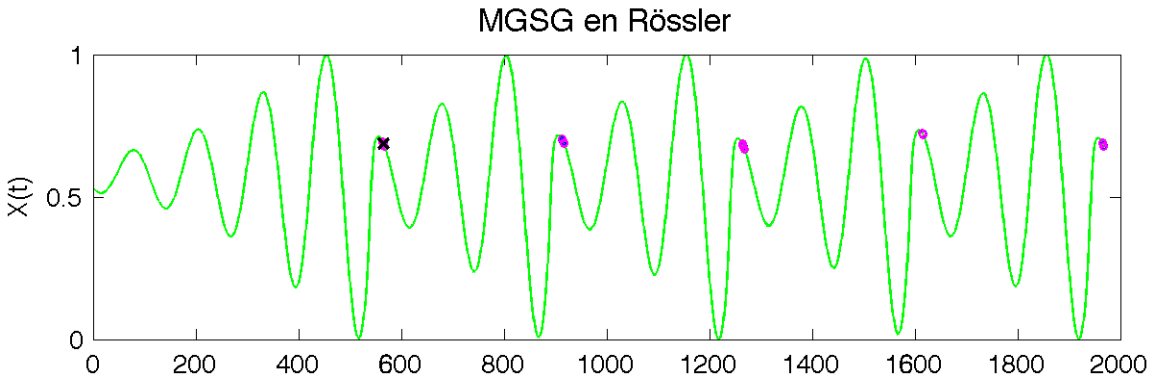


Figura 28: Dato en el tiempo $t = 565$ ilustrado con una X. Como círculos se dibujan los datos cuantificados por el segmento activado y los datos activados por los vecinos temporales del segmento activado.

En otro análisis, se introduce la muestra $x(t = 572)$, la cual activa la red MGSG como se muestra en Figura 29. Las apreciaciones son las mismas que en el caso anterior y no sólo se resalta la correcta cuantificación para ese dato, si no que también se resalta el hecho de que las conexiones temporales del segmento son hechas con segmentos que cuantifican datos consecutivos en el tiempo.

En la Figura 30 se nota nuevamente que a pesar de haber varios puntos que están espacialmente muy cercanos, se privilegia la dinámica en torno a esa zona al momento de cuantificarlos con un segmento.

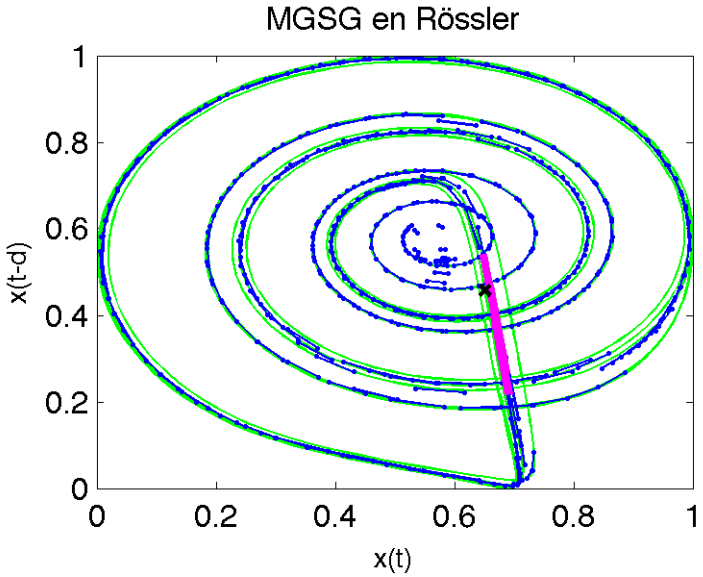


Figura 29: Red MGSG ya entrenada sobre Rössler, el dato $x(t = 572)$ se muestra con una X y el segmento activado, junto con sus vecinos temporales se ilustran en morado.

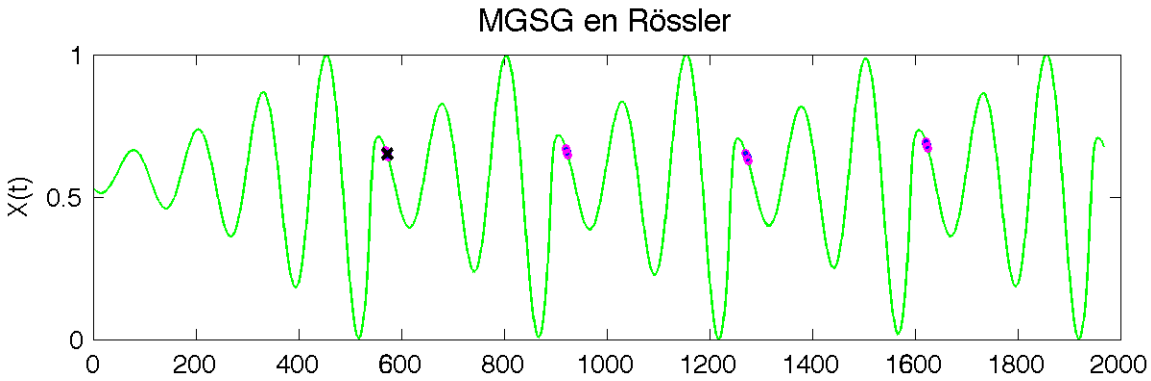


Figura 30: Dato en el tiempo $t = 572$ ilustrado con una X. Como círculos se dibujan los datos cuantificados por el segmento activado y los datos activados por los vecinos temporales del segmento activado.

4.2 Mackey-Glass

Se fija el máximo número de neuronas o segmentos utilizados en esta serie en 72. En la Figura 31 se muestra el desempeño de los algoritmos GNG, MGNG, GSG y MGSG de acuerdo al TQE en retardos desde 0 hasta 50.

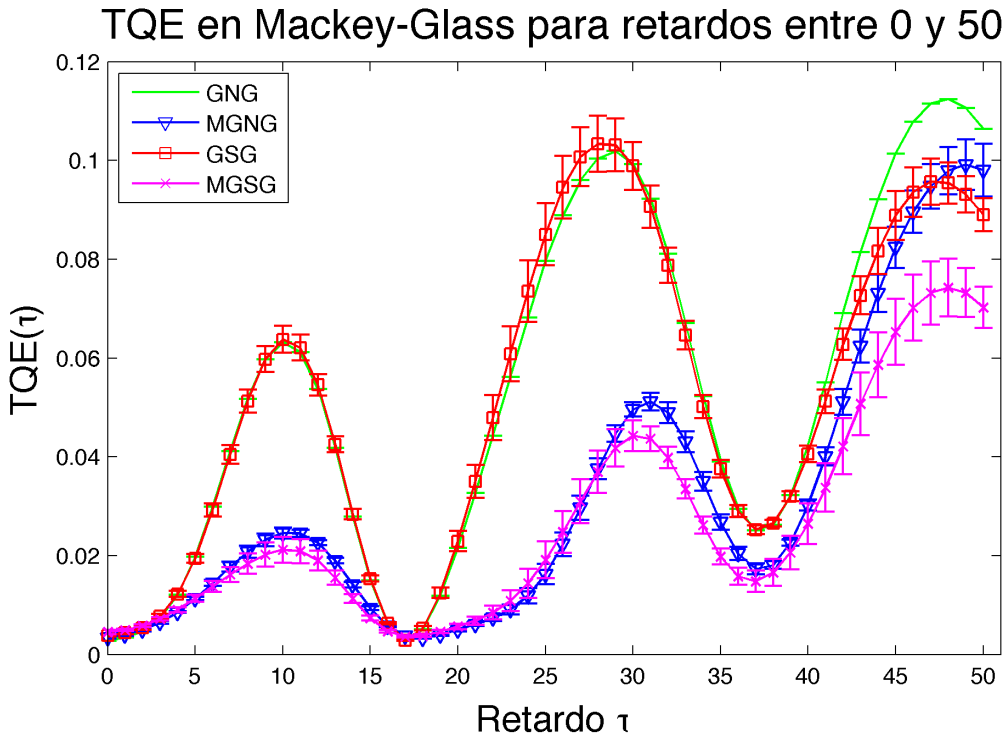


Figura 31: TQE de GNG, MGNG, GSG y MGSG en Mackey-Glass, para retardos de $\tau = 0 \ldots 50$.

El TQE para Mackey-Glass mostrado en la Figura 31 muestra mínimos y máximos locales, los cuales se encuentran en retardos aproximadamente iguales para todos los algoritmos, esto por la cuasi-periodicidad de la serie.

Si bien GNG y GSG no logran realizar una cuantificación espacio temporal eficiente en términos del TQE, cabe resaltar una vez más las ventajas en términos visuales de GSG. Gracias al uso de segmentos y conexiones temporales, GSG entrega una red más cercana

a la realidad del atractor. Esto se nota en la Figura 32, en donde se grafican las redes resultantes de GNG y GSG para Mackey-Glass.

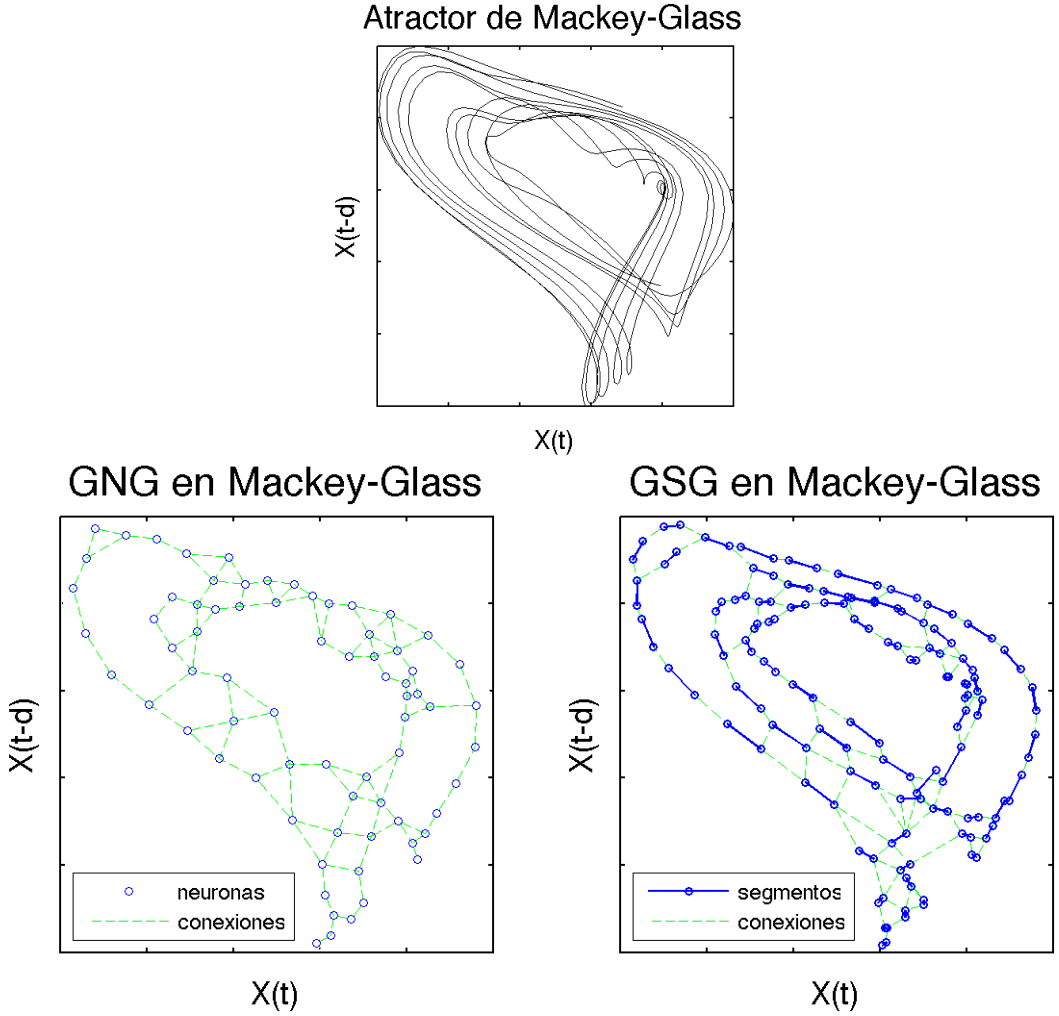


Figura 32: Resultado de las redes GNG y GSG en Mackey-Glass (mostrado arriba). Abajo a la izquierda se muestra GNG y sus conexiones espaciales. Abajo a la derecha los segmentos y las conexiones temporales de GSG entregan una visualización más completa de dinámica del atractor.

En términos de cuantificación espacio temporal los algoritmos con mejor desempeño (menor TQE) son las redes MGNG y MGSG. Sin embargo, MGSG tiene mejor desempeño que MGNG, esto se nota en las zonas donde el TQE presenta máximos locales, en especial en retardos $\tau \in \{9,10,11\} \cup \{30,\ldots,36\} \cup \{42,\ldots,50\}$, en los otros retardos se observa un comportamiento igual a MGNG. Por otro lado, MGSG presenta una ventaja sobre MGNG en términos de visualización, pues los segmentos y sus conexiones temporales entregan una red más cercana a la realidad del atractor. Esto último se muestra en la Figura 33, donde se observa además como MGNG y MGSG manejan de manera correcta la superposición de curvas presentes en el atractor en dos dimensiones.

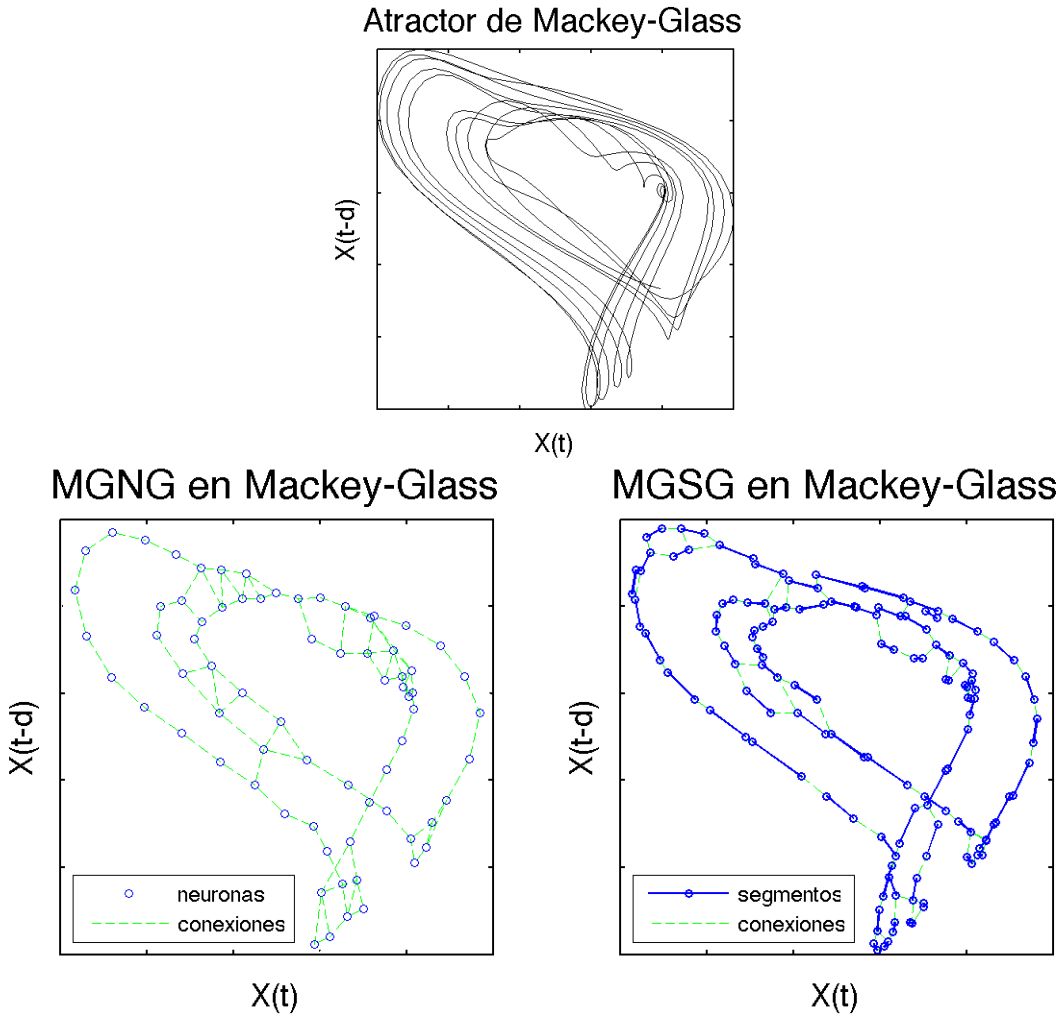


Figura 33: Redes resultantes de MGNG y MGSG en Mackey-Glass (mostrado arriba). Abajo a la izquierda MGNG con sus neuronas y conexiones espaciales. Abajo a la derecha MGSG con sus segmentos y conexiones temporales.

4.3 Lorenz

Se fija el máximo número de neuronas o segmentos utilizados en esta serie en 597. En la Figura 34 se muestra el desempeño de los algoritmos GNG, MGNG, GSG y MGSG de acuerdo al TQE en retardos desde 0 hasta 50.

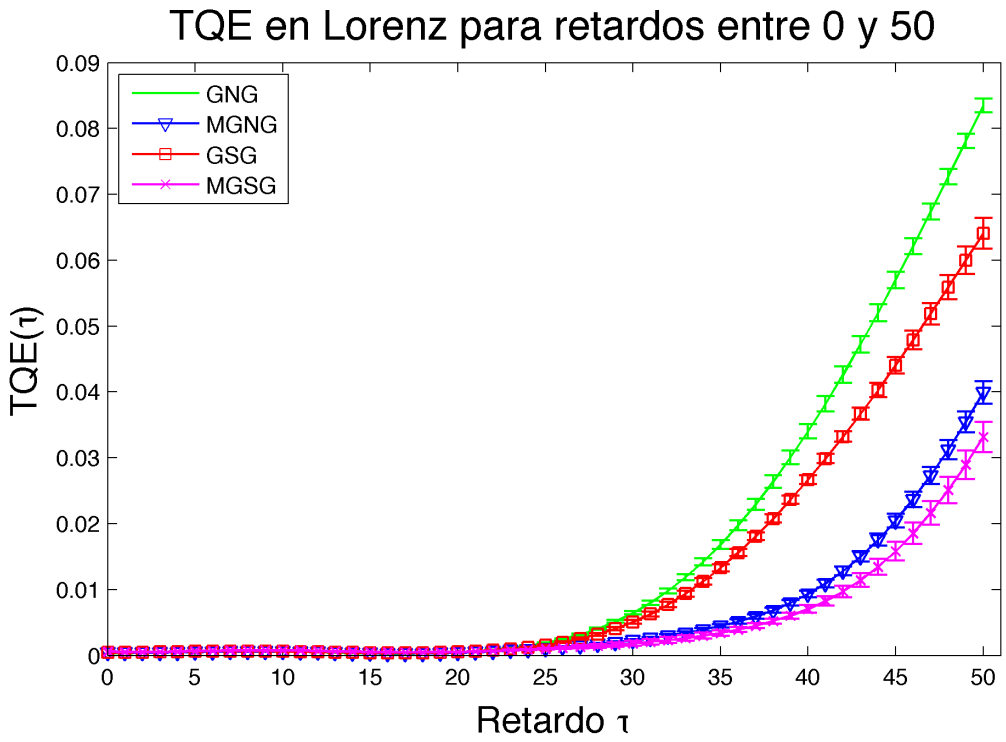


Figura 34: TQE de GNG, MGNG, GSG y MGSG en Lorenz, para retardos de $\tau = 0 \ldots 50$.

El TQE mostrado en la Figura 34 muestra diferencias en la cuantificación espacio temporal en retardos $\tau \geq 29$. GNG siendo un algoritmo de cuantificación espacial muestra el peor desempeño y es seguido por GSG, el cual si bien muestra mejor desempeño que GNG realiza un peor trabajo que MGNG. Por su lado MGSG realiza una cuantificación mejor que los otros 3 algoritmos, mostrando un buen desempeño en una base de datos densa.

4.4 Laser NH3-FIR

Se fija el máximo número de neuronas o segmentos utilizados en esta serie en 149. En la Figura 35 se muestra el desempeño de los algoritmos GNG, MGNG, GSG y MGSG de acuerdo al TQE en retardos desde 0 hasta 50.

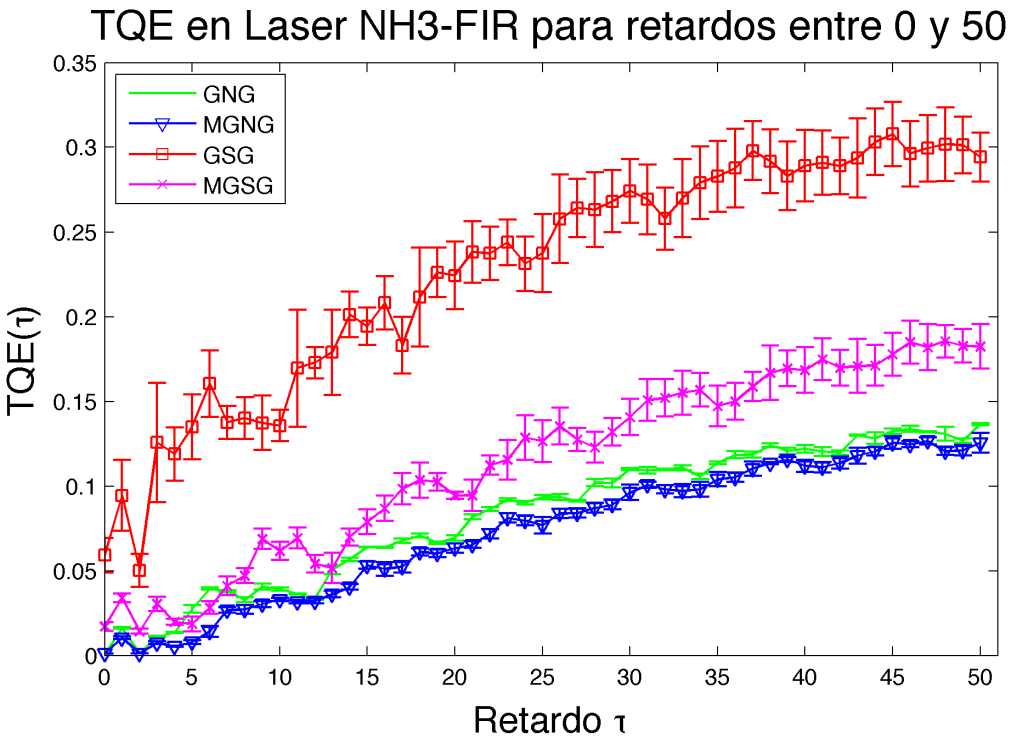


Figura 35: TQE de GNG, MGNG, GSG y MGSG en Laser NH3-FIR, para retardos de $\tau = 0 \ldots 50$.

En la Figura 35 el TQE muestra el pobre desempeño de GSG y MGSG incluso peor que la cuantificación realizada por GNG. Por su lado MGNG muestra un rendimiento levemente mejor que GNG.

Hay 2 factores que pueden explicar el desempeño de GSG y MGSG en la serie Laser NH3-FIR, por un lado no se ha realizado una selección de parámetros exhaustiva y por otro lado tampoco se ha estudiado la convergencia del algoritmo, la cual no necesariamente fue alcanzada por los algoritmos propuestos luego de las 60 épocas de entrenamiento. Este resultado sugiere una posible modificación al algoritmo la cual es mencionada en 5.1.4.

5 Conclusiones

Los algoritmos GSG y MGSG fueron puestos a prueba en 4 bases de datos de distinta naturaleza. Se evaluó su cuantificación espacio temporal utilizando la métrica TQE y sus resultados se compararon con los obtenidos por las redes GNG y MGNG.

Los resultados muestran que el algoritmo propuesto MGSG es un aporte en términos de la cuantificación espacio temporal, ya que mostró ventajas estadísticamente significativas[4] sobre las bases de datos de Mackey-Glass, Rössler y Lorenz. A pesar de sus buenos resultados en 3 bases de datos, su desempeño en la base de datos Laser NH3-FIR es de bajo rendimiento con respecto al resto de los algoritmos, esto indica una debilidad del algoritmo en series cuyos cambios son abruptos y poseen un muestreo bajo.

Por otro lado GSG no resaltó por sobre MGNG pero si lo hizo sobre GNG en las bases de datos Mackey-Glass y Lorenz. Sin embargo al igual que MGSG no mostró buenos resultados en la serie del Laser NH3-FIR.

La utilización de segmentos fue diseñada para describir de mejor manera la trayectoria de la serie, esta nueva unidad básica de cuantificación no sólo cuantifica si no que también indica la dirección temporal que poseen los datos cuantificados. Esto último, sumado a las conexiones temporales, le dio a GSG y MGSG la habilidad de generar redes visualmente más cercanas a la dinámica real de los atractores cuantificados.

También se resalta que GSG y MGSG son una generalización de GNG y MGNG, respectivamente. En particular, un segmento es la generalización de la neurona puntual pues esta última se puede entender como un segmento de largo nulo. Las reglas de ajuste, inserción y eliminación no varían notablemente pues los algoritmos propuestos se construyeron sobre GNG realizando la menor cantidad de cambios posibles.

MGSG, gracias a la correcta cuantificación espacio temporal y a la utilización de conexiones temporales, podría ser utilizado en la predicción de series de tiempos en trabajos futuros.

En conclusión el algoritmo MGSG desarrollado en este trabajo, es un algoritmo competitivo en problemas de cuantificación espacio temporal. Naturalmente, por lo nuevo del algoritmo, hay muchos espacios para mejorar MGSG y GSG, en 5.1 se exponen algunos cambios que pueden mejorar el rendimiento de los algoritmos propuestos.

[4] Ver Anexo III.

5.1 Trabajo Futuro

5.1.1 Criterio de eliminación e inserción de segmentos

Se propone introducir un criterio de eliminación de segmentos como el de Utility usado para la eliminación de neuronas en GNG [28]. Este criterio propone eliminar neuronas (o segmentos en el caso de los algoritmos propuestos) cuyo aporte a la reducción del error global de la red es bajo, al eliminar estas neuronas es posible reubicarlas en sectores donde su aporte sea mayor. Además, como se menciona en [28] este criterio le otorga a la red neuronal la habilidad de seguir series de tiempo no estacionarias.

También es posible desarrollar un criterio de inserción de segmentos basado en el error temporal.

5.1.2 Independización de las conexiones espaciales

Se propone evaluar el desempeño de la red si se elimina el uso de conexiones espaciales, cambiando principalmente 2 puntos:

- Inserción de segmentos: En lugar de insertar segmentos entre vecinos espaciales, utilizar los vecinos temporales.
- Ajuste de segmentos: En la etapa de ajuste, en lugar de ajustar los vecinos espaciales, se propone ajustar los vecinos temporales.

5.1.3 Estudiar convergencia de los algoritmos propuestos

Es necesario estudiar la convergencia de los algoritmos propuestos para definir criterios de termino y lograr evaluar de manera precisa su desempeño. En este trabajo se hizo uso de un número constante de épocas de entrenamiento, lo cual no es justo pues hay algoritmos que convergen antes que otros.

5.1.4 Hacer explicita la generalización de los algoritmos propuestos

Los algoritmos propuestos GSG y MGSG poseen una restricción que no les permite considerar, en la etapa de ajuste, trazos de curva $[x(t-n), x(t)]$ cuyo n sea nulo. Es necesario notar que si $n = 0$ entonces GSG y MGSG son equivalentes a GNG y MGNG, respectivamente. Por lo tanto se motiva a modificar el criterio de ajuste para lograr tener $n = 0$, haciendo que GSG y MGSG sean una generalización de GNG y MGNG.

5.1.5 Predicción

Como se mencionó en la sección 1.3, uno de los alcances de este trabajo es que los algoritmos propuestos abran las puertas a trabajos futuros que implementen predictores. Dados los resultados presentados se motiva a evaluar estos algoritmos en escenarios orientados a la predicción de series de tiempo.

5.1.6 Optimización de Parámetros

Los algoritmos propuestos son altamente sensibles a E_{max}. Por el lado de MGSG α_m y β_m son parámetros importantes en la cuantificación espacio temporal del algoritmo. Para un buen funcionamiento de GSG y MGSG es importante tener en cuenta estos parámetros dentro de cualquier trabajo futuro que se realice.

5.1.7 Utilización de memorias Gamma

En la sección 2.8 se realizo una introducción al algoritmo γ-GNG el cual combina GNG con un modelo de contextos basado en memorias Gamma. Esta combinación se puede realizar también entre el algoritmo propuesto GSG y el modelo de contextos de memorias Gamma.

5.1.8 Adaptar para tres dimensiones

Los algoritmos propuestos fueron desarrollados para ser utilizados con atractores bidimensionales, se propone extender GSG y MGSG para su uso con entradas tridimensionales.

6 Bibliografía

[1] Jim Holmström, "Experiments with GNG, GNG with Utility and Supervised GNG," Department of Information Technology, Uppsala University, Uppsala, Master's Thesis.

[2] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 9, pp. 1464-1480, 1990.

[3] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, and Jorma Laaksonen, "Som pak: The self-organizing map program package," University of Technology, Laboratory of Computer and Information Science, Helsinki, Finland, 1996.

[4] T. Martinetz and K. Schulten, "A 'neural-gas' network learns topologies," *Artificial Neural Networks*, vol. I, pp. 397-402, 1991.

[5] Hartmut S. Loos and Bernd Fritzke. DemoGNG. [Online]. http://sund.de/netze/applets/gng/full/GNG-U_0.html

[6] B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems*, no. 7, pp. 625-632, 1995.

[7] Andreas Andreakis, Nicolai v. Hoyningen-Huene, and Michael Beetz, "Incremental Unsupervised Time Series Analysis using Merge Growing Neural Gas," in *Advances in Self-Organizing Maps*, Risto Miikkulainen José C. Príncipe, Ed. St. Augustine, FL, USA: Springer, 2009, pp. 10-18.

[8] F. Takens, "Detecting strange attractors in turbulence," *Dynamical systems and turbulence (D. A. Rand and L. S. Young, eds.)*, pp. 366-381, 1980.

[9] G. J. Chappell and J. G. Taylor, "The temporal Kohonen map.," *Neural Networks*, vol. 6, no. 3, pp. 441-445, 1993.

[10] J. Heikkonen, and J. del R. Millan M. Varsta, "Context learning with the self organizing map," *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland*, pp. 197-202, June 1997.

[11] J. Heikkonen, J. Lampinen, and J. D. R. Millán M. Varsta, "Temporal kohonen map and the recurrent self-organizing map: Analytical and experimental comparison," *Neural Process. Lett.*, vol. 13, no. 3, pp. 237-251, 2001.

[12] N. Euliano, and S. Garani J. Principe, "Principles and networks for self-organization in space- time," *Neural Networks*, vol. 15, no. 8-9, pp. 1069-1083, 2002.

[13] T. Voegtlin, "Recursive self-organizing maps," *Neural Networks*, vol. 15, no. 8-9, pp. 979-991, 2002.

[14] M. Strickert and B. Hammer, "Merge SOM for temporal data," *Neurocomputing*, vol. 64, pp. 39-71, 2005.

[15] M. Strickert and B. Hammer, "Unsupervised recursive sequence processing," *Neurocomputing*, pp. 433-439, 2003.

[16] P.A. Estévez and R. Hernández, "Gamma-Filter Self-Organizing Neural Networks for Time Series Analysis," *WSOM 2011 (Laaksonen, J., Honkela, T. eds.)*, vol. 6731,

pp. 151-159, 2011.

[17] Jorge R. Vergara Pablo A. Estévez, "Nonlinear Time Series Analysis by Using Gamma Growing Neural Gas," in *Advances in Self-Organizing Maps*.: Springer Berlin Heidelberg, vol. 198.

[18] J. C. Principe, and Guedes B. De Vries, "Adaline with adaptive recursive memory," *Neural Networks for Signal Processing [1991]., Proceedings of the 1991 IEEE Workshop*, pp. 101-110, 1991.

[19] B. de Vries, and P. G. de Olivieira J. C. Principe, "The gamma filter – A new class of adaptive IIR filters with restricted feedback," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 649-656, 1993.

[20] B. de Vries and J. C. Principe, "The gamma model – a new neural network for temporal processing," *Neural Networks*, vol. 5, no. 4, pp. 565-576, 1992.

[21] P.A. Estévez, R. Hernández, C.A. Perez, and Held C.M., "Gamma-filter Self-Organizing Neural Networks for Unsupervised Sequence Processing," *Electronics Letters47(8)*, pp. 494-496, 2011.

[22] Pablo A. Estévez and Rodrigo Hernández, "Gamma SOM for Temporal Sequence Processing," *Advances in Self-Organizing Maps*, vol. 5629, pp. 63-71, June 2009.

[23] Foti Coleca, Andreea State, Sascha Klement, Erhardt Barth, and Thomas Martinetz, "Self-organizing maps for hand and full body tracking," *Neurocomputing*, vol. 147, pp. 174-184, January 2015.

[24] L Glass MC Mackey, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287-289, July 1977.

[25] Edward N. Lorenz, "Deterministic Nonperiodic Flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, p. 130, March 1963.

[26] O.E. Rössler, "An equation for continuous chaos," *Physics Letters A*, vol. 57, no. 5, pp. 397-398, July 1976.

[27] U., Weiss, C.O., Abraham, N.B., Tang, D. Huebner, "Lorenz-like Chaos in NH3-FIR Lasers (Data Set A)," in *Time Series Prediction: Forecasting The Future And Understanding The Past*. Addison-Wesley, FL: Elsevier, 1994, vol. 36, pp. 73-104.

[28] Bernd Fritzke, "A self-organizing network that can follow non-stationary distributions.," *Artificial Neural Networks — ICANN'97*, vol. 1327, pp. 613-618, 1997.

Anexo I

A continuación se presentan las tablas que especifican los parámetros utilizados por cada algoritmo para cada base de datos. Para todos $\alpha = 0.5$ y $\beta = 0.9995$. El número de épocas es de 60 para los algoritmos cuyo $\lambda = 300$ y 36 para aquellos en donde $\lambda = 150$.

Mackey-Glass				
Parámetros	GNG	MGNG	GSG	MGSG
ϵ_w	0.05	0.05	0.05	0.05
ϵ_n	0.0005	0.0006	0.0006	0.0006
a_{max}	60	88	88	88
λ	300	300	150	150
α_m	-	0.5	-	0.25
β_m	-	0.75	-	0.75
E_{max}	-	-	10^{-8}	10^{-7}

Tabla 0.1: Parámetros utilizados por los algoritmos sobre la serie de Mackey-Glass.

Lorenz				
Parámetros	GNG	MGNG	GSG	MGSG
ϵ_w	0.05	0.05	0.05	0.05
ϵ_n	0.0005	0.0006	0.0006	0.0006
a_{max}	60	88	88	88
λ	300	300	150	150
α_m	-	0.5	-	0.5
β_m	-	0.75	-	0.75
E_{max}	-	-	10^{-8}	10^{-6}

Tabla 0.2: Parámetros utilizados por los algoritmos sobre la serie de Lorenz.

Rössler				
Parámetros	GNG	MGNG	GSG	MGSG
ϵ_w	0.05	0.05	0.05	0.05
ϵ_n	0.0005	0.0006	0.0006	0.0006
a_{max}	60	88	88	88
λ	300	300	150	150
α_m	-	0.5	-	0.5
β_m	-	0.75	-	0.75
E_{max}	-	-	10^{-8}	10^{-6}

Tabla 0.3: Parámetros utilizados por los algoritmos sobre la serie de Rössler.

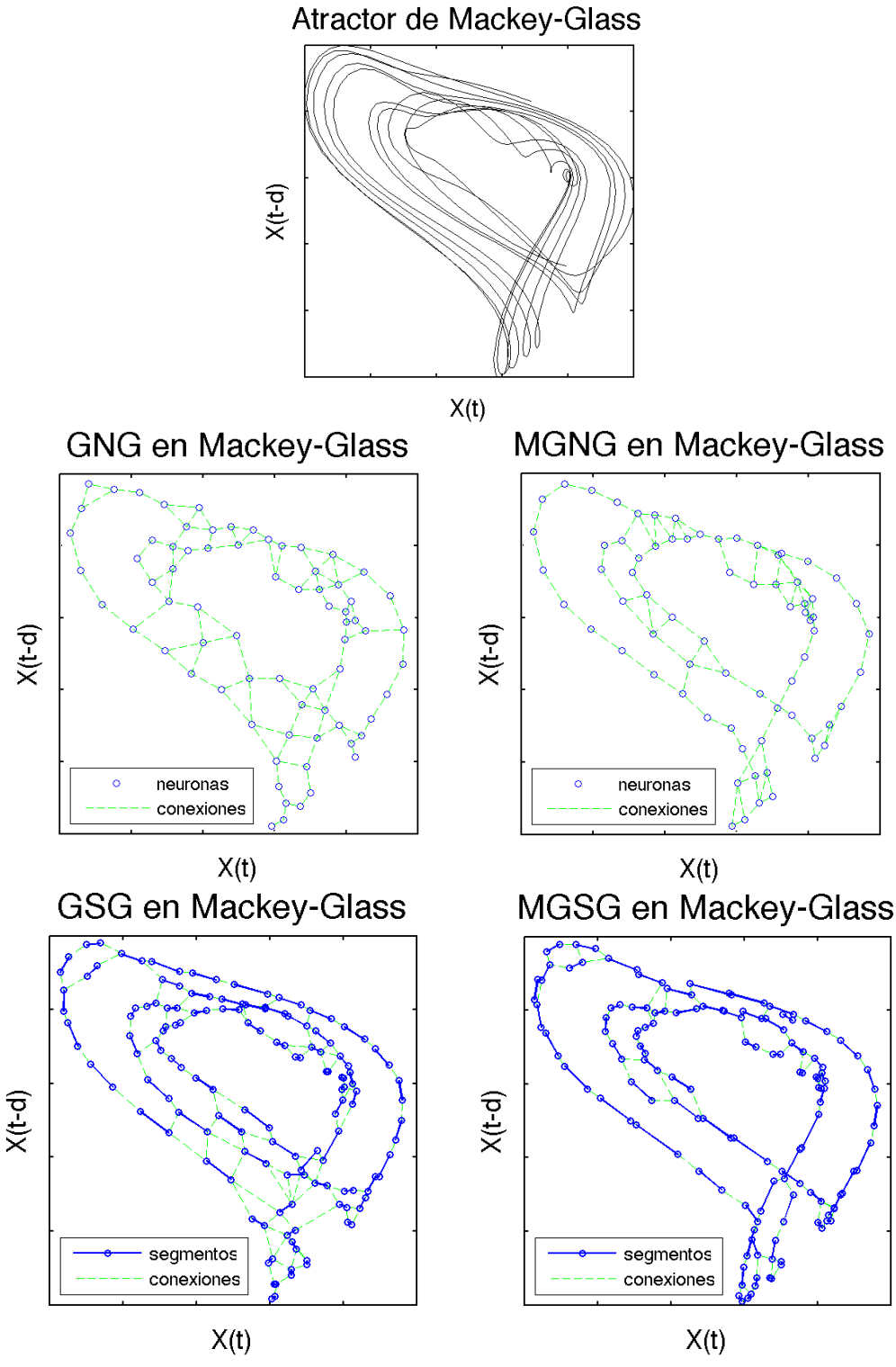
Laser NH3-FIR				
Parámetros	GNG	MGNG	GSG	MGSG
ϵ_w	0.05	0.05	0.1	0.05
ϵ_n	0.0005	0.0006	0.0006	0.0006

a_{max}	60	88	88	88
λ	300	300	150	300
α_m	-	0.5	-	0.5
β_m	-	0.75	-	0.75
E_{max}	-	-	10^{-8}	10^{-10}

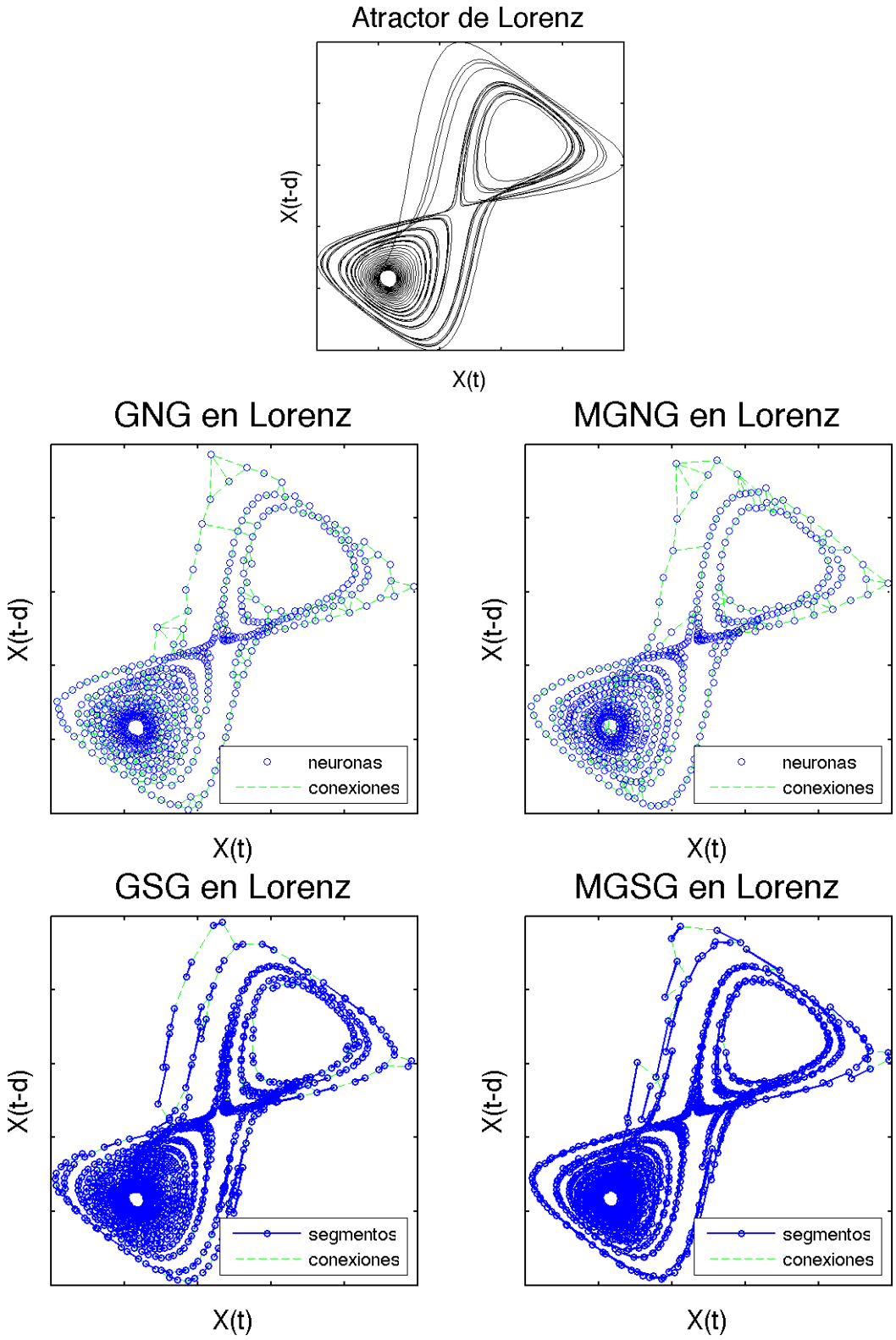
Tabla 0.4: Parámetros utilizados por los algoritmos sobre la serie de Laser NH3-FIR.

Anexo II

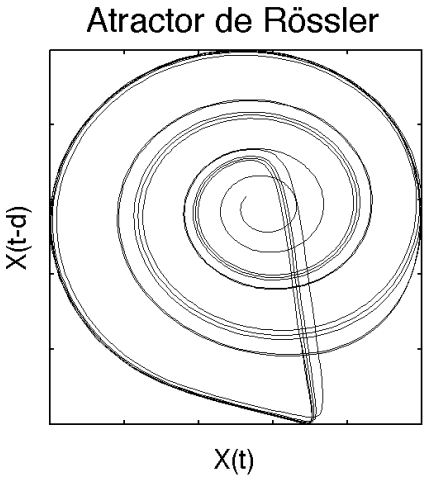
Mackey-Glass

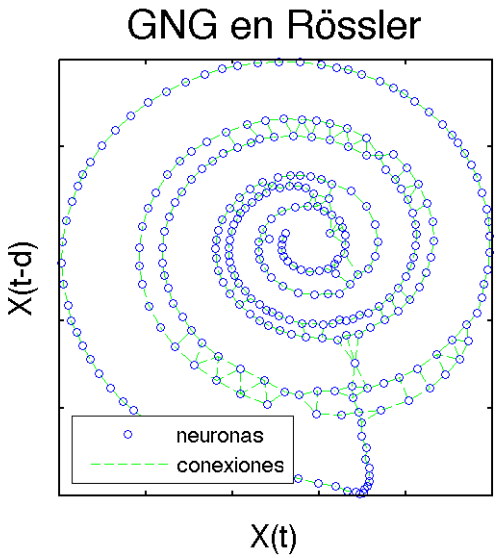


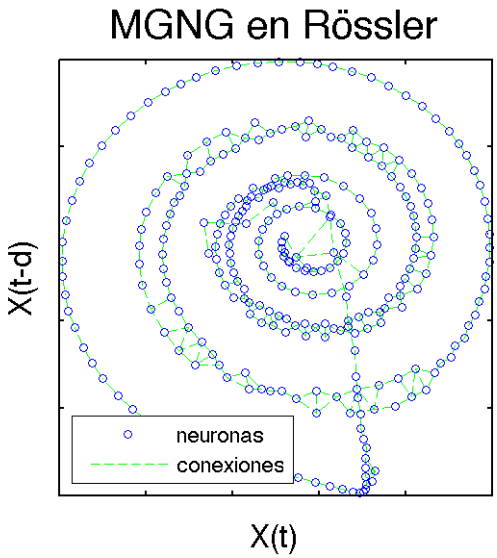
Lorenz

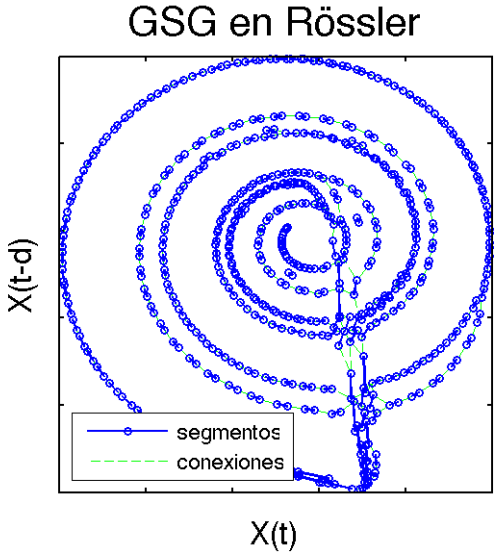

Atractor de Lorenz
X(t-d)
X(t)
GNG en Lorenz
X(t-d)
neuronas
conexiones
X(t)
MGNG en Lorenz
X(t-d)
neuronas
conexiones
X(t)
GSG en Lorenz
X(t-d)
segmentos
conexiones
X(t)
MGSG en Lorenz
X(t-d)
segmentos
conexiones
X(t)

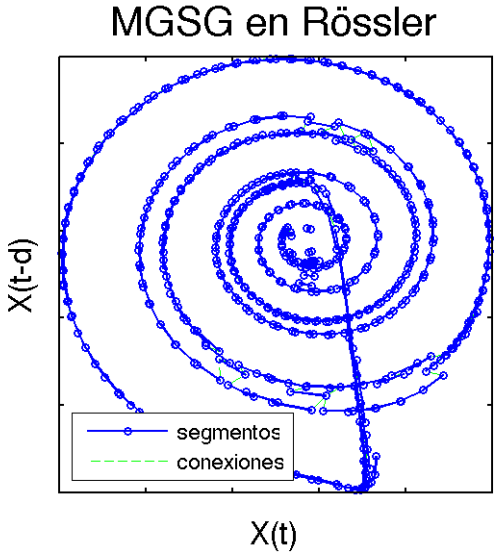
Rössler


Atractor de Rössler
X(t-d)
X(t)

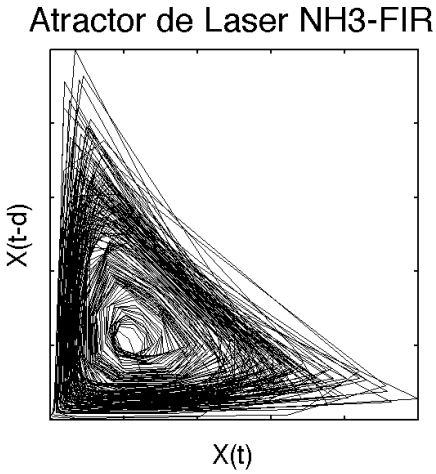

GNG en Rössler
X(t-d)
neuronas
conexiones
X(t)

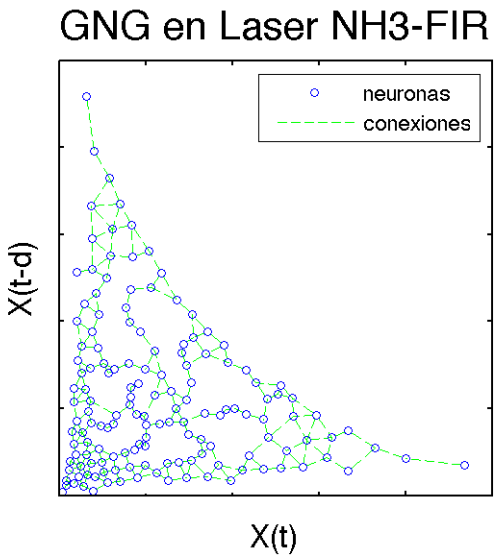

MGNG en Rössler
X(t-d)
neuronas
conexiones
X(t)

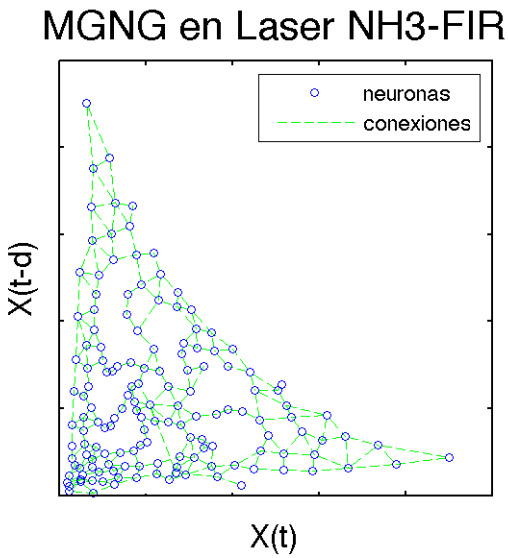

GSG en Rössler
X(t-d)
segmentos
conexiones
X(t)

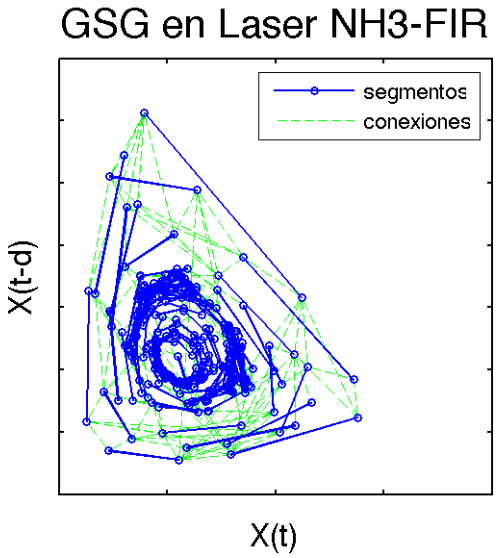

MGSG en Rössler
X(t-d)
segmentos
conexiones
X(t)

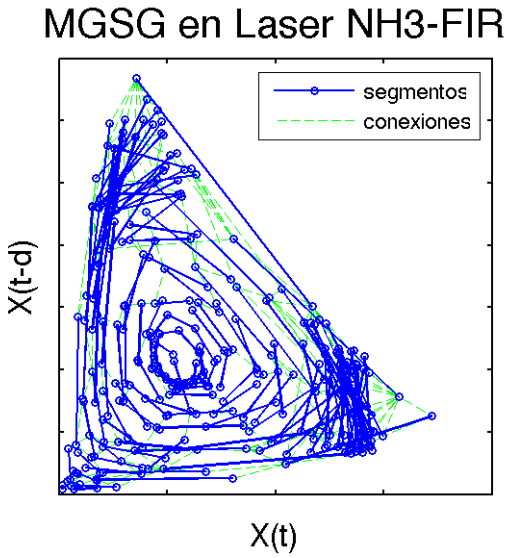
Laser NH3-FIR











Anexo III

Desempeño de MGSG

A continuación se muestran los intervalos de retardos en los que MGSG demostró una ventaja estadísticamente significativa sobre los otros algoritmos. Se hizo uso de la Prueba t de Student con un 5% de tolerancia sobre el TQE medido 10 veces por algoritmo. En el total se indican los intervalos en los que MGSG fue superior a todos los algoritmos.

Rössler			
GNG	GSG	MGNG	Total
4-30, 34-50	3-31, 34-50	24-50	24-30, 34-50

Mackey-Glass			
GNG	GSG	MGNG	Total
4-16, 18-50	3-16, 18-50	7-17, 30-37, 41-50	7-16, 30-37, 41-50

Lorenz			
GNG	GSG	MGNG	Total
21-50	22-50	27-50	27-50

Laser NH3-FIR			
GNG	GSG	MGNG	Total
-	0-50	-	-

Desempeño de GSG

A continuación se muestran los intervalos de retardos en los que GSG demostró una ventaja estadísticamente significativa sobre los otros algoritmos. Se hizo uso de la Prueba t de Student con un 5% de tolerancia sobre el TQE medido 10 veces por algoritmo. En el total se indican los intervalos en los que GSG fue superior a todos los algoritmos.

Rössler			
GNG	MGNG	MGSG	Total
36-50	0, 32, 33	0, 1, 32	-

Mackey-Glass			
GNG	MGNG	MGSG	Total
5-7, 33-50	18, 49, 50	0-2, 17	-

Lorenz			
GNG	MGNG	MGSG	Total
5-11, 22-50	0	0-2, 5-18	-

Laser NH3-FIR			
GNG	MGNG	MGSG	Total
-	-	-	-