Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Control Automático

"Redes neuronales LSTM para modelado difuso de sistemas no lineales"

T E S I S

Que presenta

Jose Francisco Vega Lopez

Para obtener el grado de

Doctor en Ciencias

En la especialidad de

Control Automático

Director de Tesis

Dr. Wen Yu Liu

Ciudad de México Diciembre 2022

Agradecimientos

En principio, doy las gracias al CONACyT (Consejo Nacional de Ciencia y Tecnología) por su apoyo para la realización de esta tesis.

Agradezco a mi director de tesis, el Dr. Wen Yu Liu, por la atención y el tiempo brindados durante este proyecto.

También, hago mención de los sinodales que evaluaron esta tesis, cuyos comentarios me ayudaron a enriquecer mi trabajo.

Y finalmente, a mi familia, les agradezco el apoyo incondicional que siempre me dan.

Índice general

Índice de figuras

Índice de tablas

Resumen

Los sistemas inteligentes, como las redes neuronales artificiales y los sistemas difusos, al día de hoy están presentes en varios aspectos de la sociedad como las comunicaciones, la economía, la industria y la ciencia. En el control automático, estos sistemas son de utilidad en la caracterización de sistemas no lineales, aprovechando sus cualidades de adaptabilidad y robustez. Pero, mayormente en aplicaciones en tiempo real y en sistemas con presencia de ruido y cambios abruptos de comportamiento, el uso de estos sistemas aún presenta los siguientes inconvenientes: la dificultad de obtener un error de aproximación mínimo, tiempos de entrenamiento considerablemente altos, los algoritmos resultantes no son factibles (computacionalmente hablando) en la mayoría de las ocasiones, y la falta de condiciones de estabilidad útiles para un correcto funcionamiento. Para hacer frente a ello, en esta tesis se propuso una red difuso-neuronal, la Red Recurrente LSTM Difusa (LSTM son las siglas de “long-short term memory”, término de memoria grande-corta). Esta red, junto con su algoritmo de entrenamiento, logra sobrellevar los inconvenientes mencionados dada su estructura interna. Lo anterior, se muestra en tareas de control e identificación de sistemas que se desarrollan a lo largo del documento, en las cuales se hicieron comparaciones con otros algoritmos semejantes para sustentar las ventajas de la red propuesta.

Abstract

Intelligent systems, such as artificial neural networks and fuzzy systems, are nowadays present in various aspects of society such as communications, economy, industry and science. In automatic control, these systems are useful in the characterization of non-linear systems, taking advantage of their adaptability and robustness. But, mostly in real-time applications and in systems with the presence of noise and abrupt changes in behavior, the use of these systems still presents the following drawbacks: the difficulty of obtaining a minimum approximation error, considerably high training times, the resulting algorithms are not feasible (computationally speaking) on most occasions, and the lack of useful stability conditions for correct operation. To address this, in this thesis a fuzzy-neural network, the Fuzzy LSTM Recurrent Network (LSTM from "long-short term memory"), was proposed. This network, together with its training algorithm, manages to overcome the mentioned drawbacks given its internal structure. This is shown in control and system identification tasks that are developed throughout the document, in which comparisons with other similar algorithms were made to sustain the advantages of the proposed network.

Capítulo 1

Introducción

Para introducir la temática del presente trabajo, se mencionan a continuación las características sobresalientes de la llamada “computación inteligente”. Además, se habla de la relación existente entre la computación inteligente con el control e identificación de sistemas. De esta forma, se justifica el diseño del algoritmo propuesto en la tesis.

1.1. Motivaciones

Actualmente, vivimos en un mundo de cambio constante, aludiendo a lo dicho por Zadeh [1], en donde los cambios más relevantes están relacionados con la informática. Dentro de la informática se distinguen tres partes principales: los sistemas de información, los sistemas inteligentes y los sistemas de información inteligentes. Los sistemas mencionados están involucrados con el manejo de información, la cual es obtenida de datos recabados de cualquier fenómeno o situación.

Un sistema de información se refiere a las tecnologías relacionadas con la adquisición y organización de datos, ejemplos destacados de estos sistemas son la Internet, los teléfonos celulares y las computadoras convencionales. Por otro lado, un sistema inteligente es una máquina, o algoritmo, que tiene la capacidad de analizar datos, los llamados “dispositivos inteligentes” y el software especializado se cuentan entre los principales ejemplos de estos sistemas.

No obstante, conforme han pasado los años, la relación entre los sistemas de información y los sistemas inteligentes ha aumentado; la inteligencia requiere de información y viceversa. Como consecuencia de lo anterior, el desarrollo de aquellos sistemas ya no es individual sino en

conjunto, dando lugar al auge de los sistemas de información inteligentes. Entonces, los sistemas de información inteligentes son tecnologías con la capacidad de adquirir, organizar y analizar datos.

Además, el desarrollo de los sistemas de información inteligentes está ligado a la computación flexible, ya que estos sistemas aprovechan las bondades de este tipo de computación. La computación flexible, también conocida como computación inteligente, es una asociación de metodologías entre las que resaltan: la lógica difusa, la neurocomputación, la computación evolutiva y la computación probabilista. La mayoría de estas metodologías tienen en común que son complementarias y simbióticas en vez de ser competitivas y exclusivas entre sí. Por lo que son frecuentes los trabajos que combinan varias metodologías de la computación inteligente, como es el caso de los sistemas neuro-difusos, difuso-genéticos, neuro-genéticos y neuro-difuso-genéticos.

Cada metodología que integra a la computación inteligente aporta algo diferente, tales aportaciones se resumen de la siguiente manera: la lógica difusa ofrece una base, conocida como computación de estilo granular, para las operaciones con palabras; la neurocomputación se encarga del análisis y diseño de las redes neuronales; el desarrollo de algoritmos de búsqueda atañe a la computación evolutiva, esta búsqueda usualmente va dirigida hacia la solución óptima de algún problema; y la computación probabilista indaga sobre las técnicas que se relacionan con la teoría de probabilidad para la toma de decisiones.

Entonces, la computación inteligente es capaz de lidiar con datos que conlleven incertidumbres, verdades parciales y aproximaciones. Además, los resultados obtenidos al aplicar sus metodologías son localizables, es decir, se puede entender el origen dichos resultados. Esto, acorde a la filosofía de la computación inteligente, que es la emulación de la mente humana, en específico, de los procesos del pensamiento humano que lidia con información incompleta e imprecisa para aplicarlos en la resolución de diversos problemas computacionales.

Las características anteriores se pueden resumir como adaptabilidad y robustez, lo cual es la principal distinción y ventaja de la computación inteligente con respecto de la computación convencional, como se detalla en los trabajos de Zadeh [2], Zadeh [3] y Magdalena [4]. La computación convencional trabaja con información exacta, sus objetivos son la precisión y la certeza. Pero, en las problemáticas que acontecen en la realidad, es complicado elaborar representaciones exactas o captar todas las propiedades de cualquier fenómeno en el que se tenga interés.

Por todo lo anterior, se entiende el porqué la computación inteligente ha tenido un mayor desarrollo en comparación a la computación convencional. En consecuencia, los sistemas de información inteligentes están envueltos en diversos campos de la ingeniería y las ciencias, como lo es el área de control automático, donde estos sistemas se usan para tareas de control e identificación de sistemas.

1.1.1. La computación inteligente en el control e identificación de sistemas

En el control, así como en la identificación de sistemas, se trabaja con modelos que describen las cualidades de un sistema o un conjunto de sistemas. Un sistema, a grandes rasgos, se define como la combinación de elementos que actúan juntos hacia un objetivo determinado. Los sistemas no se limitan solamente a fenómenos físicos o químicos, también pueden referirse a abstracciones, tal es el caso de los fenómenos que se estudian en áreas como la economía. Comúnmente, el modelo de un sistema solo representa algunas propiedades de éste, incluidas las de su entorno, dependiendo de para qué se va a usar este modelo. Por ejemplo, el modelo requerido para la construcción de un motor es diferente al modelo que se usa para diseñar su control, en el primer caso se requiere de una mayor representación de variables a diferencia del segundo caso. Un modelo no es exclusivo de un solo sistema, es decir, pueden existir diferentes modelos para un sistema.

Por otra parte, ocurre que el modelo de un sistema se adapte para representar a otros sistemas. Un modelo puede ser el resultado de cambiar parámetros en determinadas estructuras ya establecidas, como lo son las ecuaciones o conjuntos de datos, en función de alguna parte o comportamiento que es de interés del sistema a modelar. El ejemplo más notorio de ello son los sistemas inteligentes, cuando se usan para generar modelos de otros sistemas, ya que estos sistemas tienen una estructura fija, pero modifican sus parámetros para imitar otros sistemas.

Debido al gran número de técnicas para construir modelos, conocidas como modelados, es oportuno tener una clasificación de éstas para agilizar la elección de alguna. Una clasificación que sobresale es la expuesta por Amara et al. [5], donde los modelados se agrupan en tres tipos:

- Modelado “caja blanca”: el sistema en cuestión, así como su entorno, se consideran conocidos por completo a partir de las leyes (ecuaciones) que los explican.
- Modelado “caja negra”: todo el conocimiento de un sistema se deriva a partir de las mediciones

de sus parámetros y de la salida del sistema ante una entrada determinada.

- Modelado “caja gris”: se combinan las leyes o ecuaciones correspondientes a un sistema y las mediciones que se pueden obtener de éste.

Se hace alusión a cajas por la comparación esencial del modelado caja negra con las cajas negras de los aviones. Esta metáfora se usa para designar aquel elemento estructural de un modelo sobre el funcionamiento de un sistema.

En el modelado caja blanca, rara vez, las ecuaciones se acercan a lo que se pretende expresar y cuando es el caso se tiene que usar una gran cantidad de ecuaciones para obtener una caracterización satisfactoria; en el peor de los casos, se hacen suposiciones al no encontrar ecuaciones que encajen. Por su parte, en el modelado caja gris se combinan ecuaciones e información obtenida del sistema, ganando robustez, pero perdiendo parte de la claridad de los componentes del sistema en el modelo. Asimismo, el modelado caja negra usa solo datos obtenidos de un sistema, usualmente mediciones, para caracterizar a éste, lo que implica una dependencia total a los medios que recopilan los datos.

Entonces, se puede decir que no es conveniente aplicar el modelado caja blanca, salvo las ocasiones en que las ecuaciones capten la estructura de un sistema sin mayor dificultad. En general, es mejor emplear el modelado caja gris y caja negra ya que se ajustan mejor a los problemas de la vida real. No obstante, la selección de cualquier técnica dependerá de los requerimientos individuales en cada caso; por ejemplo, a veces será mejor usar el modelado caja gris porque se desea conocer parte de la conformación del sistema en cuestión, mientras que otras veces esto no es relevante y se puede usar el modelado caja negra en su lugar.

También resalta la similitud del modelado caja gris y caja negra con la computación inteligente, según lo descrito en esta sección. Ciertamente, la computación inteligente tiene similitudes con lo que se plantea en estos modelados, especialmente las metodologías de la lógica difusa y la neurocomputación. Algunos artículos resaltan y explican en detalle esta relación, como lo hacen Zadeh [2] y Ibrahim [6]; se tiene que tanto la neurocomputación, así como la lógica difusa, buscan crear modelos adaptables, basados en análisis de datos a través del estudio de las redes neuronales y sistemas difusos, respectivamente.

En el control automático los modelos difusos son apreciados porque, a diferencia de los modelos que se obtienen con redes neuronales, estos modelos representan a un sistema en términos

de variables lingüísticas o lenguaje natural. Estos modelos son tan válidos como aquellos modelos obtenidos convencionalmente con ecuaciones matemáticas que describen fenómenos físicos y químicos. Los modelos difusos tienen un gran respaldo teórico, además se prestan a trabajar con redes neuronales fácilmente, por lo que es posible ajustar los parámetros de estos modelos con algoritmos de aprendizaje y así acercar el modelo difuso aún más al modelo deseado.

Los sistemas difusos, dados sus principios, pueden clasificarse dentro del modelado caja gris o incluso dentro del modelado caja blanca. Mientras que las redes neuronales se encuentran dentro del modelado caja gris y caja negra, porque éstas trabajan con datos ajenos a ellas para ajustar sus parámetros y a veces se pueden adaptar ecuaciones matemáticas con las redes para trabajar conjuntamente.

Aunque hay otros algoritmos inteligentes que se pueden emplear en algún tipo de modelado, como la computación evolutiva, éstos por lo general son auxiliares a las redes neuronales y sistemas difusos. Algo a tener en cuenta es que estos algoritmos pueden dar origen a sistemas híbridos con facilidad, aprovechando la buena compatibilidad entre ellos por su origen. En torno a todo esto existe una enorme cantidad de trabajos y teoría, de lo cual se habla más en el próximo capítulo.

Hablando del siguiente capítulo, al revisar los trabajos que se citan ahí, es evidente que las redes neuronales, sistemas difusos y sistemas difuso-neuronales tienen una gran presencia dentro de la ciencia y tecnología. Particularmente, en el control automático, se han obtenido buenos resultados en la obtención de modelos de sistemas, debido a las características fundamentales de estos algoritmos como su fácil aplicación, adaptabilidad y robustez. Aunque estos algoritmos tienen sus desventajas, como que los parámetros aprendidos no son fáciles de interpretar, sus estructuras son relativamente grandes, tienen tiempos de entrenamiento largos, y la falta condiciones de estabilidad que garanticen su buen funcionamiento. Por lo que no es de extrañarse el encontrar propuestas de nuevas estructuras, las cuales pueden atender las necesidades de una situación en concreto.

1.2. Planteamiento del problema

La problemática que inspiro la creación de este trabajo se sintetiza de la siguiente forma: los modelos, que caracterizan a funciones no lineales en tiempo discreto equivalentes a sistemas

no lineales, obtenidos con redes neuronales artificiales, sistemas difusos y redes difuso-neuronales en el área de control automático, a pesar de las ventajas que ofrecen, principalmente en términos de adaptabilidad y robustez para sistemas con cambios de comportamiento abruptos, aún presentan los siguientes inconvenientes: la dificultad de obtener un error de aproximación mínimo, tiempos de entrenamiento considerablemente altos, los algoritmos resultantes no son factibles (computacionalmente hablando) en la mayoría de las ocasiones, y la falta de condiciones de estabilidad útiles para el correcto funcionamiento de los algoritmos. Estos inconvenientes se ven reflejados mayormente en aplicaciones en tiempo real.

Al hablar de funciones no lineales en tiempo discreto se hace alusión a sistemas no lineales en tiempo discreto, los cuales se conforman con datos de mediciones, en determinados periodos de tiempo, de la respuesta del sistema en el que se tiene interés en tiempo real. Razón por la cual, los algoritmos inteligentes se conforman a partir de funciones no lineales discretas. A lo largo de este trabajo, se habla regularmente de esto, además, solo se trabajó con funciones y sistemas en tiempo discreto.

Se entiende por sistemas con cambios de comportamiento abruptos, o bruscos, a los sistemas que describen un comportamiento variable a través del tiempo, por ejemplo, la respuesta de un sistema que sigue una determinada trayectoria descrita por diferentes formas y tipos de señales. Además, los sistemas con perturbaciones en su ambiente, las cuales afectan el comportamiento de la respuesta de dichos sistemas, también se consideran sistemas con cambios abruptos.

Un error de aproximación mínimo, se refiere a que se quiere que el error, el cual surge entre las respuestas del modelo generado con algún algoritmo inteligente y el sistema en cuestión, se mantenga en valores cercanos a cero. Con esto, se mide la eficiencia de los modelos generados con algoritmos inteligentes: el error de aproximación obtenido con determinado algoritmo inteligente, comparado con el de otros algoritmos, que se mantenga más cercano a cero durante un determinado lapso de tiempo es considerado el más eficiente.

Con tiempo de entrenamiento razonable, se quiere decir que el algoritmo no le tome tanto tiempo aprender la dinámica del sistema de interés, por ejemplo, es preferible que este proceso dure unos pocos minutos en vez de un par de horas. Aunque, el tiempo de entrenamiento de los algoritmos inteligentes suele variar conforme a las aplicaciones y condiciones en que se usan, a

veces el proceso puede ser corto y otras veces este proceso es largo porque la información que se emplea no se puede conseguir de inmediato. El fin de este proceso, generalmente, se determina cuando se consiguen errores de aproximación satisfactorios (cercanos a cero).

Un algoritmo factible, computacionalmente hablando, se refiere a que el algoritmo que resulte como solución a la problemática pueda ser reproducido en equipos de cómputo con diferentes capacidades de cálculo, o diferentes capacidades operacionales. En otras palabras, que los equipos de cómputo con capacidades limitadas sean capases de ejecutar el algoritmo.

1.2.1. Solución propuesta a la problemática planteada

A partir de la problemática presentada, surgió la idea de que un algoritmo híbrido entre redes neuronales artificiales y sistemas difusos, adecuado a los avances recientes en esos campos, podría cubrir las desventajas que se mencionaron anteriormente. Una red neuronal artificial puede modificar sus parámetros en función de procesos de aprendizaje, mientras que un sistema difuso puede ofrecer una interpretación estructurada, con reglas causa-efecto, a lo aprendido por la red.

Con los avances recientes que se han hecho, en torno a los algoritmos inteligentes mencionados, se tiene la capacidad de afrontar los inconvenientes planteados anteriormente. La justificación de la estructura que da solución a la problemática se encuentra en la Subsección 2.2.4, y su desarrollo se muestra a lo largo del Capítulo 3. Esto, teniendo en cuenta que se quiere verificar la siguiente hipótesis, que fue la guía principal de este trabajo.

De antemano, se denominó a la estructura propuesta, como solución a la problemática, “Red Recurrente LSTM Difusa”. De ahí el título de este trabajo, “Redes neuronales LSTM para modelado difuso de sistemas no lineales”, ya que se presentan dos variaciones de esta red en el documento.

1.2.2. Hipótesis propuesta

Considerando a la problemática presentada y su posible solución, se estableció la hipótesis base de la investigación: Se puede diseñar una estructura de red difuso-neuronal para estimar sistemas, para aplicaciones en tiempo real relacionadas con el control e identificación de sistemas con variaciones de comportamiento bruscas, la cual presente un error mínimo de modelado, en comparación a otros algoritmos semejantes establecidos con anterioridad, aplicando conceptos actuales de la computación inteligente.

El error de modelado se refiere al error que se presenta, en el contexto de control automático, entre la respuesta de un algoritmo inteligente y el sistema que el algoritmo pretende emular. Igual al error de aproximación descrito con anterioridad.

Los conceptos actuales de la computación inteligente, con respecto a las redes neuronales y sistemas difusos, hacen referencia al "aprendizaje de la máquina" (machine learning) y "aprendizaje profundo" (deep learning). Estos términos se detallan en el Capítulo 2 de este trabajo.

1.3. Objetivos de la tesis

En relación con lo anterior, se definió al objetivo principal de la tesis como: diseñar una nueva estructura de red difuso-neuronal que sea capaz de modelar sistemas no lineales, en tiempo real, con un error mínimo de modelado, en específico, a aquellos sistemas que presentan cambios bruscos en su comportamiento, así como perturbaciones en su ambiente.

Para alcanzar este objetivo, en adición, se establecieron objetivos particulares que dictaron las actividades que se realizaron para conformar esta tesis:

- Revisar la teoría, de redes neuronales y sistemas difusos, para diseñar la estructura de red difuso-neuronal adecuada para modelar sistemas no lineales, conforme a lo requerido.
- Desarrollar la estructura de la red difuso-neuronal, que se mencionó con anterioridad, así como un algoritmo de entrenamiento para ajustar sus parámetros.
- Establecer maneras de emplear a la nueva red en la identificación y control de sistemas no lineales, conforme a lo requerido.
- Hacer modificaciones a la estructura de la nueva red, con el fin de minimizar aún más al error de modelado originado entre la respuesta de la red y el sistema a estimar.

1.4. Organización de la tesis

La estructura de la presente tesis se divide en seis capítulos. El contenido los otros capítulos se resume de la siguiente manera:

- En el Capítulo 2, se revisa la teoría que antecede al trabajo de esta tesis.
- En el Capítulo 3, se describe la estructura de la red difuso-neuronal que satisface los objetivos

de este trabajo, incluyendo su uso en la identificación de sistemas no lineales.

- En el Capítulo 4, se explica como usar la nueva red difuso-neuronal para el control de sistemas no lineales a través de varios esquemas de control.
- En el Capítulo 5, se muestra el desarrollo de una aplicación en el mundo real con la red propuesta, la cual consiste en la identificación de daño en edificios provocado por sismos.
- En el Capítulo 6, se modifica a la estructura de la red que se estableció en el Capítulo 3, con lo que se logró una reducción mayor del error de modelado en la identificación y control de sistemas no lineales.

Al final del documento, se dan conclusiones generales con base en lo expuesto, además de posible trabajo a futuro, y se lista la bibliografía usada.

1.5. Contribuciones

Las contribuciones que se obtuvieron, como consecuencia de la realización de esta tesis, se listan brevemente como:

- La estructura de la red difuso-neuronal, denominada como Red Recurrente LSTM Difusa, en conjunto con su algoritmo de entrenamiento. Esto corresponde al contenido del Capítulo 3, al igual que al contenido de los siguientes artículos:
  - Wen Yu and Francisco Vega, Nonlinear System Modeling Using the Takagi-Sugeno Fuzzy Model and Long-Short Term Memory Cells, Journal of Intelligent & Fuzzy Systems, 2020, vol. 39, no 3, p. 4547-4556.
  - Francisco Vega and Wen Yu , Fuzzy Modeling Using LSTM Cells for Nonlinear Systems, IEEE World Congress on Computational Intelligence (WCCI 2020), Glasgow, UK, July 19-24, 2020.
- Formas de aplicar la Red Recurrente LSTM Difusa en la identificación y control de sistemas no lineales, que corresponde a parte del contenido del Capítulo 3 y al contenido del Capítulo 4, respectivamente.
- El desarrollo de una metodología de detección de daño estructural en edificios debido a sismos, la cual involucra el uso de datos obtenidos de teléfonos celulares y a la Red Recurrente

LSTM Difusa. Esto corresponde al contenido del Capítulo 5, al igual que al contenido de los siguientes artículos:

- Francisco Vega and Wen Yu , Smartphone Based Structural Health Monitoring Using Deep Neural Networks, Sensors and Actuators A: Physical, 2022, vol. 346, p. 113820.
- Francisco Vega and Wen Yu , Building Health Monitoring Using Smartphone and Deep Neural Networks, 4th International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI' 2022).

- La modificación de la estructura de la Red Recurrente LSTM Difusa, de un sistema difuso de tipo 1 a un sistema difuso de tipo 2, con lo que se logró reducir el error de modelado presente entre la red y el sistema a estimar. Lo anterior corresponde a lo mostrado en el Capítulo 6.

Los artículos reportados se han publicado antes de la fecha de presentación de este trabajo.

1.6. Conclusiones del capítulo

Se mencionaron las motivaciones que justificaron la realización de esta tesis. Se estableció la problemática de interés, así como la solución que se consideró adecuada para ella. Asimismo, se presentaron los objetivos de la investigación que, además, son la base de la estructura de este trabajo.

Capítulo 2

Antecedentes

La propuesta que se plantea en este trabajo proviene del desarrollo que han tenido las redes neuronales artificiales y los sistemas difusos. A continuación, estos algoritmos inteligentes se abordan desde su presentación inicial hasta sus últimos desarrollos notables.

2.1. Marco teórico

Antes de hablar del desarrollo actual de las redes neuronales artificiales y sistemas difusos, en esta sección se revisan los conceptos fundamentales de la teoría de estos algoritmos inteligentes, iniciando por las redes neuronales artificiales.

2.1.1. Redes neuronales

Las redes neuronales artificiales, mejor conocidas como redes neuronales, son percibidas como sistemas compuestos por una cantidad considerable de elementos (neuronas) con conexiones ponderadas entre ellos dentro de una estructura. Estos sistemas son bastante conocidos en varios campos, como el procesamiento de imágenes y el control automático, por lo que su teoría es muy amplia. Entre las diversas fuentes de información, Haykin [7] ofrece un basto referente, en donde se muestra la historia, los conceptos básicos, análisis y aplicaciones de las redes neuronales.

Desde su inicio, en la década de los 40, el desarrollo de redes neuronales artificiales busca imitar a sus homónimos biológicos; bajo la premisa de crear sistemas que emulen a las neuronas en los seres vivos, con la habilidad de obtener experiencia (conocimiento) de una situación y aplicarla en

otras situaciones similares. Las neuronas son células que se encuentran en el cerebro de un ser vivo, Figura 2.1 (a), cuyo funcionamiento es complejo, pero se simplifica de la siguiente manera: el núcleo procesa la información que recolectan las dendritas, ya procesada la información se envía a través del axón a otras neuronas usando la sinapsis (conexiones entre terminales del axón y dendritas).

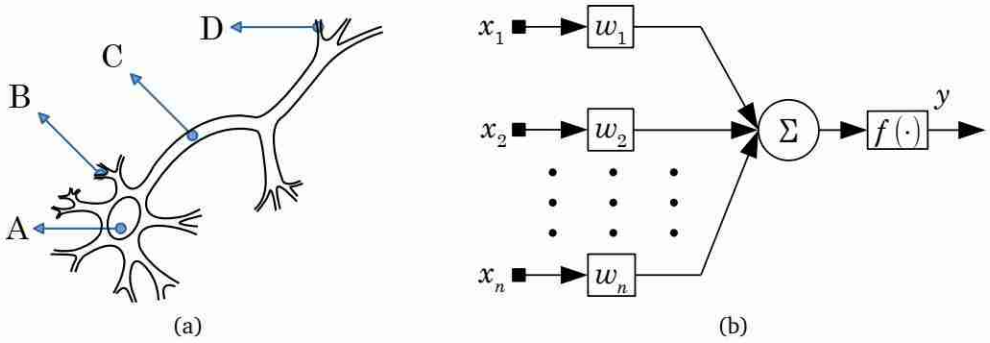


Figura 2.1. Tipos de neuronas: (a) bosquejo de las partes de una neurona biológica (A es el núcleo, B las dendritas, C el axón y D las terminales del axón); (b) esquema de una neurona artificial.

A partir del funcionamiento de las neuronas biológicas se estableció un esquema para las neuronas artificiales, Figura 2.1 (b) y (2.1), este esquema refleja la esencia de la neurona biológica matemáticamente hablando: un conjunto de datos asignados a n variables, definidas como x_n, se ponderan por los correspondientes w_n pesos para sumarse entre ellos; el resultado de esta suma se usa como el argumento de una función $y = f(\cdot)$, llamada función de activación, que arroja un determinado resultado.

$$y = f\left(\sum_{i=1}^{n} w_i x_i\right) \tag{2.1}$$

Entonces, se entiende por red neuronal biológica al cúmulo de neuronas biológicas conectadas unas a otras que comparten información para realizar una función del sistema nervioso, se involucra además la adquisición de experiencia (conocimiento) por medio de procesos biológicos exhibiendo la habilidad de la red para aprender y adaptarse a su entorno. Análogamente, una red neuronal artificial es un conjunto de neuronas artificiales conectadas entre sí, en una determinada estructura, con el fin de efectuar alguna tarea; para lograr esta tarea se deben ajustar los pesos de las neuronas artificiales (conocimiento), por medio de algún algoritmo, capacidad similar al aprendizaje en las neuronas biológicas. El conocimiento del que se habla se refiere a información

almacenada o modelos usados por un ser vivo o máquina para interpretar, predecir y responder apropiadamente al mundo exterior.

Desde este punto en adelante se denotará por neurona a la neurona artificial y por red neuronal a la red neuronal artificial. El conocimiento es adquirido por la red neuronal de su ambiente a través de un proceso de aprendizaje, en este proceso los parámetros de la red son ajustados. Los principales parámetros de cualquier red neuronal son los pesos de las neuronas que la integran, la función de estos pesos es guardar o servir de representación del conocimiento adquirido. Más características de las redes neuronales se mencionan en las siguientes subsecciones.

2.1.2. Características de las redes neuronales

La organización de las neuronas, Figura 2.2, dentro de una red neuronal está definida en capas o bloques. Las capas son agrupaciones de neuronas cuyas entradas provienen de una misma fuente y sus salidas tienen el mismo destino. Cada neurona en una capa está conectada a las neuronas de las capas anterior y posterior, con excepción de las capas de entrada y salida que solo se conectan a una capa posterior o anterior, respectivamente. Además, las redes neuronales se pueden clasificar, de acuerdo al flujo de las conexiones de las neuronas, en tres tipos: redes neuronales consecutivas (o de conexiones hacia adelante), redes neuronales recurrentes (con realimentación de entradas) y redes neuronales híbridas (combinación los dos tipos anteriores).

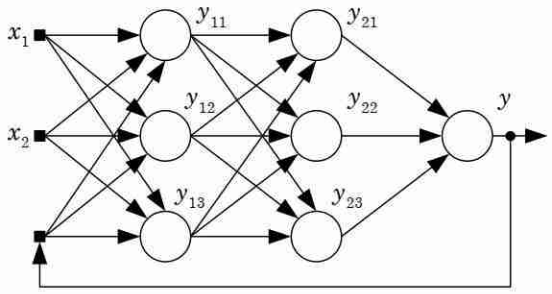


Figura 2.2. Ejemplo de una red neuronal: se tienen 3 entradas (una de ellas recurrente), una capa de entrada con tres elementos, una capa interna u oculta con tres elementos y una capa de salida con un solo elemento. Las neuronas de esta red, en forma de círculos, son versiones resumidas de la Figura 2.1 (b).

Como cualidades sobresalientes de las redes están: su estructura representa un mapeo entre datos de entrada y salida de un sistema, gran adaptabilidad, respuestas evidentes e información contextual asociada a la estructura de la red, tolerancia a fallos. Estas particularidades se engloban

en los procesos de entrenamiento y generalización de las redes neuronales.

El proceso de entrenamiento, más conocido como entrenamiento o aprendizaje, es el ajuste de los pesos de las neuronas de la red, por medio de cualquier método o algoritmo que sigue un determinado índice de desempeño o guía. El proceso de generalización, o simplemente generalización, de una red neuronal es la capacidad de la red para generar resultados aceptables ante datos de entrada no presentes durante su aprendizaje. La generalización es totalmente dependiente del entrenamiento, ya que muestra directamente el rendimiento de este último; entre más adecuada sea la generalización mayor fue la eficacia del entrenamiento.

En la literatura existe un gran número de metodologías para el aprendizaje de las redes neuronales, estas metodologías se han desarrollado a lo largo del tiempo inspirándose en distintos puntos de vista. Los enfoques de los algoritmos de entrenamiento para redes neuronales se dividen en supervisado, no supervisado y reforzado. Estos paradigmas pueden aplicarse a cualquier arquitectura de red neuronal, por lo que cada enfoque inspira diferentes tipos de algoritmos. Los procedimientos de aprendizaje y generalización de las redes neuronales, además de los paradigmas de aprendizaje y algunos ejemplos de estos son explicados brevemente en Musumeci et al. [8].

Para los algoritmos basados en el enfoque supervisado, es necesaria una figura externa que controle el proceso e incorpore información al mismo, esta figura puede ser un conjunto de datos o un observador que califica el desempeño del algoritmo. El propósito de estos algoritmos es que la red cambie sus pesos conforme a ejemplos con entradas y salidas de datos; los datos deben cubrir todos los aspectos de lo que se desea aprender. Ya que se ha logrado el ajuste de los pesos, con un error mínimo definido, el proceso se ha completado y la red puede generalizarse. Entre ejemplos de estos algoritmos se cuentan al algoritmo de mínimos cuadrados, el algoritmo de propagación hacia atrás y las redes de funciones de base radial.

De forma opuesta, los algoritmos de entrenamiento no supervisado se elaboran sin considerar a una figura externa que los califique, en su lugar los algoritmos se deben organizar a sí mismos con base en un criterio interno e información local designada en la red neuronal. Solo se disponen de muestras de datos de entrada para el entrenamiento, las cuales la red neuronal clasificará en diferentes grupos. Como ejemplos de este algoritmo se tienen a las redes basadas en estadísticas, las redes de Kohonen y otros tipos similares mencionados por Miljković [9].

Los algoritmos de entrenamiento basados en el enfoque de refuerzo no cuentan con datos de lo que se quiere aprender, en su lugar se usan datos recabados de las interacciones de la red neuronal con su ambiente. La filosofía de estos algoritmos, se basa en cómo una red neuronal debe tomar medidas de su entorno para maximizar alguna noción de recompensa a largo plazo, es decir, buscar la relación entre situaciones y acciones a partir de procedimientos de prueba y error. Por sus características, es el enfoque que es más complicado de aplicar, por lo que este enfoque no es muy utilizado. El aprendizaje por refuerzo es particularmente adecuado para problemas que incluyen un intercambio de recompensas a largo plazo contra corto plazo.

Asimismo, los algoritmos de entrenamiento pueden catalogarse según los datos que utilizan, ya sea en línea (datos obtenidos en tiempo real) o fuera de línea (datos obtenidos de un historial). Algo que ha resaltado a lo largo de la historia de las redes neuronales, sin considerar los enfoques de aprendizaje, es que el algoritmo de propagación hacia atrás (conocido por su nombre en inglés como backpropagation) es el algoritmo de entrenamiento de redes con mayor popularidad, debido a su simpleza y versatilidad. Tanto así que este algoritmo ha servido de referencia para crear otros más sofisticados, como es explicado por Rehman y Nawi [10], Saduf y Wani [11] y Chaudhary et al. [12], y por tanto es conveniente hacer una descripción de éste.

Algoritmo de propagación hacia atrás

El algoritmo de propagación hacia atrás se basa en la aplicación de la técnica de gradiente descendiente, el cual es un algoritmo de optimización iterativo de primer orden para encontrar el mínimo de una función; en redes neuronales es utilizado para minimizar un término de error, cambiando cada peso de las neuronas de una red en proporción de la derivada del error con respecto a ese peso, lo que implica que la función de activación de la red debe de ser diferenciable.

Además, el algoritmo de propagación hacia atrás puede ser local en espacio o local en tiempo. Se dice que el algoritmo es local en espacio, porque un peso puede ajustarse, o actualizarse, usando solamente información almacenada en todos los pesos de la red, incluido ese mismo peso. Local en tiempo se refiere a que la actualización de los pesos de la red es continua, es decir, que la actualización depende la información de la iteración anterior del proceso de entrenamiento. Independientemente del caso, la obtención de las derivadas para el gradiente es igual en ambas situaciones.

Hay que considerar que (2.1) se puede replantear de la siguiente forma:

$$v = W^T X \tag{2.2}$$

$$y = f(v) \tag{2.3}$$

con: $X = [1, x_1, \cdots, x_n]^T \in \mathbb{R}^n$ como el vector de $n+1$ datos de entrada, $W = [w_0, w_1, \cdots, w_n]^T \in \mathbb{R}^n$ como el vector de pesos en donde cada elemento de W corresponde a un elemento de X, $w_0 \in \mathbb{R}$ es un peso que pondera a una entrada constante $x_0 = 1$ y en conjunto se llama bias, $v \in \mathbb{R}$ es el argumento de la función de activación $f(\cdot) : \mathbb{R} \to \mathbb{R}$ ligada a la salida $y \in \mathbb{R}$ de la neurona. Por conveniencia, todos los elementos están definidos dentro del conjunto de los números reales.

Se tienen diferencias entre (2.1), (2.2) y (2.3), a pesar de que representan lo mismo. Básicamente, (2.2) y (2.3) son la representación vectorial de una neurona, con el agregado término bias. El bias puede, o no, estar presente en las neuronas, en la teoría es muy empleado porque este término beneficia al entrenamiento de la red al ofrecer una mayor cantidad de parámetros ajustables. Por mencionar una ventaja, entre las varias que existen, el bias ofrece información adicional cuando los datos de entrada de la red son cero.

Continuando con la descripción, se considera una red similar a la de la Figura 2.2, pero sin realimentar por lo que se tiene un solo elemento de salida, como en (2.2) y (2.3), aunque puede extenderse al caso de varias salidas. Comenzando por la capa de salida, se define la función del error que se quiere minimizar, (2.5), formada a partir de (2.4). Siguiendo, (2.5) es conocida como la energía del error instantánea y (2.6) como la energía promedio del error sobre el entrenamiento, esta última sirve para medir la eficacia del entrenamiento.

$$e = d - y \tag{2.4}$$

$$\xi = \frac{1}{2} e^T e \tag{2.5}$$

$$\Xi = \frac{1}{N} \sum_{q=1}^{N} \xi(q) \tag{2.6}$$

siendo: $d \in \mathbb{R}$ el valor que se desea aprender, $N \in \mathbb{N}$ el total de iteraciones del proceso, $\xi \in \mathbb{R}$ y $\Xi \in \mathbb{R}$. Además, (2.5) considera la iteración “actual” del entrenamiento, mientras que (2.6) considera

todas las iteraciones. Asimismo, (2.6) se usa para medir la eficacia del entrenamiento; entre menor sea su valor mejor se realizó el entrenamiento de la red.

Teniendo presente lo establecido anteriormente, el entrenamiento se inicia asignando un valor inicial a los pesos de las neuronas de la red para que ésta pueda comenzar a trabajar. Una vez que se tiene una salida por parte de la red, se procede a calcular (2.4) y (2.5), (2.6) puede calcularse al final del proceso porque solo sirve para hacer un análisis de éste. A partir de estos cálculos, se pueden obtener los gradientes para la actualización de los pesos, como se muestra en las siguientes ecuaciones:

$$\frac{\partial \xi}{\partial w_j} = \frac{\partial \xi}{\partial e}\frac{\partial e}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_j}, \qquad \text{con } j=\overline{0,n} \tag{2.7}$$

$$\frac{\partial \xi}{\partial w_j} = -ef'(v)x_j \tag{2.8}$$

y la corrección de los pesos queda definida, por la regla delta, como:

$$\triangle w_j = -\eta\frac{\partial \xi}{\partial w_j}, \qquad \text{con } j=\overline{0,n} \tag{2.9}$$

con $0<\eta\in\mathbb{R}$ como una constante que regula la velocidad del entrenamiento, (2.9) se reescribe siguiendo a (2.8) y empleando el término δ_j conocido como gradiente local:

$$\triangle w_j = \eta\delta_j x_j, \qquad \text{con } j=\overline{0,n} \tag{2.10}$$

$$\delta_j = \frac{\partial \xi}{\partial v} \tag{2.11}$$

$$\delta_j = \frac{\partial \xi}{\partial e}\frac{\partial e}{\partial y}\frac{\partial y}{\partial v} \tag{2.12}$$

$$\delta_j = -ef'(v) \tag{2.13}$$

Los pesos de la red se modifican, sumando a su valor actual la corrección, tal cual se expresa en (2.14). Los pesos modificados se sustituyen en la red y se repite todo el proceso desde el inicio. El proceso puede detenerse en un determinado número de iteraciones, o cuando (2.5) y (2.6) han alcanzado un valor satisfactorio. Entonces, se tiene:

$$w_{j_+} = w_j + \triangle w_j, \qquad \text{con } j=\overline{0,n} \tag{2.14}$$

donde el subíndice + hace alusión al cambio del valor de los pesos para la siguiente iteración de la red.

El signo en (2.9) es negativo porque se desea que el valor del gradiente descienda en el espacio de pesos, dicho de otra forma, se busca la dirección del cambio de pesos que reduzca a (2.5), la cual es indicada por (2.8) cuando se multiplica por menos uno. Esta descripción puede aplicarse a las neuronas de las capas antes de la capa de salida, cambiando el término de error en (2.8) por la suma de los gradientes δ_j (multiplicados por sus respectivos pesos) de las neuronas en la capa siguiente a las que llega la salida de la neurona que se está tomando en cuenta.

Si las neuronas son del tipo recurrentes (con entradas realimentadas) también es posible aplicar este algoritmo de aprendizaje, en el entendido de que se deben de hacer algunas modificaciones para ello, tal como es explicado por Pearlmutter [13] y Jaeger [14]. Comúnmente, las modificaciones consisten en hacer algunos cambios en la estructura de las neuronas recurrentes, para tratarlas como neuronas consecutivas. Una de las desventajas del algoritmo de propagación hacia atrás es que el entrenamiento puede durar mucho tiempo; se requieren de cambios pequeños en el gradiente, debido a que los cambios grandes pueden generar ruido en la respuesta final de la red, lo que provoca que el proceso sea en ocasiones relativamente lento.

Otra desventaja del algoritmo de propagación hacia atrás es la desaparición de los gradientes (en inglés esto es conocido como “vanishing gradient problem”), que significa que la red neuronal no puede propagar información útil, del gradiente del error, desde la capa de salida a las capas cercanas a la entrada de datos de la red. El problema es que, en ocasiones, el gradiente será muy pequeño dada la función de activación (que tiene valores de frontera entre 0 y 1), evitando que los pesos cambien su valor. En el peor de los casos, esto puede detener completamente a la red neuronal de seguir aprendiendo. Algunas soluciones implican el cambio de función de activación, por alguna más elaborada que las funciones de saturación y sigmoidea, así como dar prioridad a ciertas regiones de la red neuronal en el entrenamiento.

Para tratar el problema del ruido, el parámetro η de (2.10) puede variar según el comportamiento de la red durante su entrenamiento; entre más pequeño sea el valor habrá menor ruido pero el entrenamiento será lento, si el valor es muy grande la presencia de ruido aumentará y la red puede volverse inestable. Una solución a esto es alterar a (2.14), como se muestra abajo. El término $0 \leq \alpha < 1 \in \mathbb{R}$, en (2.15), es llamado término momento, el subíndice $-$ significa que se

usa el valor anterior del ajuste de pesos.

$$w_{j_+} = w_j + \triangle w_j + \alpha \triangle w_{j_-}, \qquad \text{con } j = \overline{0,n} \tag{2.15}$$

Continuando, (2.15) se interpreta como (2.14) más un término nuevo. Esta modificación actúa como un amortiguamiento al ruido generado por aumentar el valor de η, permitiendo que el entrenamiento sea rápido y evitando que se llegue a estancar en algún mínimo local de la superficie formada en el espacio del error, ya que se usa el gradiente en función del error para la actualización de los pesos de la red neuronal.

Retomando a las redes neuronales, por mencionar algunas desventajas en general de todos los tipos de redes, éstas dependen mucho de la experimentación, o mejor dicho de la prueba y error. No existe una metodología exacta para determinar su tamaño, llegando en ocasiones el caso de que una red neuronal tenga un exceso de capas y de neuronas. Los algoritmos de entrenamiento son lentos porque implican, muchas veces como en el algoritmo de propagación hacia atrás, el cálculo de derivadas parciales y ajuste de parámetros externos. La mayoría de los modelos obtenidos con redes, a pesar de aprender un fenómeno físico, son simplemente no casuales físicamente hablando.

2.1.3. Aprendizaje de la máquina y aprendizaje profundo

En las últimas décadas, a las redes neuronales se les ha asociado con el concepto del aprendizaje de la máquina (del término en inglés “machine learning”). Según Jordan y Mitchell [15], el aprendizaje de la maquina es la disciplina encargada de dar respuesta a las siguientes preguntas: ¿cómo construir sistemas de cómputo que se mejoren automáticamente a través de la experiencia? y ¿cuáles son las leyes fundamentales de la teoría de información, cómputo y estadística que gobiernan a los sistemas de aprendizaje en los que se encuentran las computadoras, los humanos y las organizaciones?.

Dentro de la inteligencia artificial, el aprendizaje de la máquina ha progresado como una metodología de creación de algoritmos inteligentes. Dado que para muchas situaciones, es mejor entrenar sistemas inteligentes por medio de ejemplos del comportamiento de datos entrada-salida deseados (experiencia), en vez de imponer de antemano la posible respuesta para todos los datos de entrada en dichos sistemas. Además, estos métodos permiten analizar una gran cantidad de

datos gracias a las mejoras tecnológicas que ha tenido en la computación a lo largo del tiempo.

Un problema de aprendizaje se define como mejorar el desempeño en la ejecución de una tarea, por medio de un algoritmo que emplee algún tipo de experiencia. Existen una gran variedad de métodos para afrontar esta problemática, que tienen que ver principalmente con el manejo de datos según sus características. El aprendizaje de la máquina se puede entender como un buscador de métodos, guiado por la experiencia, que elige al método que optimiza la ejecución de determinada tarea.

En la selección del método se consideran criterios como el volumen de datos, el tipo de datos, la robustez ante errores en las suposiciones en el modelado, errores en los datos de entrada, y si para determinado problema de aprendizaje una cantidad limitada de datos es suficiente para desarrollar un algoritmo o no. Tal caracterización teórica de los métodos del aprendizaje de la máquina es dejada a la estadística y la computación inteligente.

Los métodos de aprendizaje supervisado son los más empleados en el aprendizaje de la máquina, entre ellos, aparte de las redes neuronales, se encuentran: los árboles de decisiones, la predicción de decisiones, los regresores logísticos, las máquinas de soporte vectorial, las máquinas de kernel y los clasificadores Bayesianos. En años resientes, se han retomado a las redes neuronales, redefiniendolas en estructuras “profundas” (redes neuronales profundas), pertenecientes al aprendizaje profundo (del término en inglés “deep learning”).

El aprendizaje profundo, según LeCun, Bengio e Hinton [16], permite a los modelos informáticos compuestos de múltiples capas de procesamiento aprender representaciones de datos con múltiples niveles de abstracción. En el aprendizaje profundo se descubren estructuras intrincadas en grandes conjuntos de datos, comúnmente por medio del algoritmo de propagación hacia atrás. La finalidad del aprendizaje profundo es indicar cómo una máquina o algoritmo debería cambiar sus parámetros internos, los cuales son usados para calcular la representación de un sistema, en cada capa interna de la máquina a partir de la representación en la capa anterior dentro de ésta.

Se entiende por red neuronal profunda a una pila de secciones (diferente a las capas en una red común) sujetas a entrenamiento, y las cuales desempeñan una determinada labor. Cada sección en la pila transforma su entrada para incrementar tanto la selectividad, así como la invariancia de la representación. De esta forma, se han resuelto muchos de los problemas que se presentaban

al aplicar redes neuronales convencionales, especialmente los problemas al usar algoritmos de entrenamiento basados en gradientes. Entre estas redes destacan, según Deng [17], Lipton et al. [18], Liu et al. [19], Shafique et al. [20], Islam et al. [21], Alom et al. [22] y Zhang et al. [23], las redes: modulares, autoasociativas, convulocionales y de término de memoria grande-corta.

Las redes modulares son una colección de diferentes tipos de redes neuronales, trabajando de forma independiente, pero contribuyendo en la salida general de la red. Cada red tiene un conjunto de datos de entrada, el cual es diferente para cada una, con lo que se realizarán diferentes subtareas no relacionadas entre sí. La ventaja de este tipo de red es que descompone un proceso complejo, el entrenamiento de la red, en pequeños procesos con menor complejidad. Aunque se pueden tener demasiadas redes trabajando al mismo tiempo, incrementado la exigencia de cálculo y memoria, como desventaja.

Por otro lado, las redes autoasociativas (“autoencoders” en inglés) se encargan del tratamiento de los datos de entrada con el fin de disminuir su tamaño. Este tipo de red es similar al perceptron multicapa, pero en vez de realizar predicciones basándose en determinados datos, la red autoasositiva trata de reconstruir estos datos de otra forma más conveniente. En general, estas redes tienen tres etapas: entrada, codificación y reconstrucción; en la entrada se reciben los datos que van a ser transformados en la etapa de codificación, posteriormente en la reconstrucción a la transformación anterior se le aplica un mapeo para obtener un estimado de los datos originales.

El concepto de las redes convolucionales es semejante al de las redes de conexiones hacia adelante, pero estas redes usan la operación de convolución en lugar de la típica multiplicación de matrices; en este sentido, el número de pesos disminuye y por ende la complejidad de la red también baja. Esta red está formada comúnmente por tres capas: convolucional, submuestreo y totalmente conectada. La capa convolucional se encarga de estimar los pesos de la red, la capa de submuestreo disminuye el tamaño de estos pesos, y la capa totalmente conectada da el resultado final conforme a las anteriores.

Similar a las redes recurrentes tradicionales, las redes de término de memoria grande-corta (o LSTM por las siglas en inglés de “long-short term memory”) trabajan con recurrencias en su estructura interna: se introduce el concepto de compuertas, las cuales analizan los datos de entrada, salida y términos de memoria de la red por separado, para que después, los resultados de cada

compuerta se interconecten entre sí obteniendo una respuesta global. Una versión corta de estas redes son las unidades recurrentes con compuertas (o GRU por las siglas en inglés de “gated recurrent units”), las cuales tienen un número menor de compuertas en comparación a las LSTM, ya que combinan la operación de dos o más compuertas, de una LSTM normal, en una sola compuerta.

A pesar de que las redes profundas se han desarrollado durante los últimos años, son las redes convolusionales y LSTM las estructuras con más atención, ya que se consideran mejoras directas de sus contrapartes convencionales. Motivo por el cual se han implementado estas redes, relativamente nuevas, en tareas que se han resuelto aplicando redes neuronales convencionales, obteniendo mejores resultados, además de implementarse en nuevas problemáticas sin mayor dificultad. La mejora viene en el aprendizaje y distribución de información en toda la estructura de la red neuronal.

2.1.4. Sistemas difusos

Los sistemas difusos tratan de emular una parte específica del proceso cognitivo del ser humano, el razonamiento. Para ello, los sistemas difusos toman como base lo establecido por la lógica difusa fundada por Zadeh en los años 60 y que actualmente continúa en desarrollo. En Wang [24], se puede encontrar el desarrollo histórico de la lógica difusa y los sistemas difusos, teoría de los conjuntos difusos y ejemplos de su uso en varias aplicaciones. A continuación, se habla, de manera general, de los elementos utilizados en sistemas difusos y de los que atañen a este trabajo.

La lógica difusa, por medio de los conjuntos difusos, ofrece otra manera de representar elementos dentro de un conjunto diferente a la manera convencional. Dado un conjunto universal, universo de discurso U, que contenga todos los elementos implicados en un contexto particular o aplicación. Un conjunto clásico A, en U, puede definirse ya sea listando todos sus elementos o definiendo las propiedades que deben ser cumplidas por los miembros del conjunto como en (2.16).

$$A=\{x\in U \mid x \text{ cumple ciertas propiedades}\} \tag{2.16}$$

$$\mu_A(x)=\begin{cases}1, & \text{si } x\in A\\ 0, & \text{si } x\notin A\end{cases} \tag{2.17}$$

Otro método para representar a los elementos de A es a través de una función de pertenencia o membrecía, denotada por $\mu_A(x)$. En (2.17), la función de pertenencia discrimina a todos los

elementos que no pertenecen a A asignándoles un valor de 0, mientras que al contrario, a los que pertenecen les asigna un valor de 1. Por tanto, A es equivalente a $\mu_A(x)$, en el sentido de que, si se conoce la función de pertenencia de un conjunto se conoce al conjunto mismo.

Pero sucede que hay conjuntos que no tienen fronteras bien definidas, la teoría clásica de conjuntos requiere que las propiedades de un conjunto se definan apropiadamente, lo que dificulta el manejo de estos conjuntos. Es aquí donde se introduce el concepto de conjunto difuso: un conjunto difuso, en U, es caracterizado por una función de pertenencia que toma valores en el intervalo $[0,1]$. Un conjunto difuso es una generalización de un conjunto clásico porque permite a su función de pertenencia tomar cualquier valor entre 0 y 1 (función continua con rango $[0,1]$); un conjunto clásico da valor a los elementos que cumplen todos sus requerimientos y un conjunto difuso da un valor a los elementos en la medida que cumplen sus requerimientos.

Un conjunto difuso puede representarse como un conjunto de pares ordenados, (2.18), correspondientes a algún elemento genérico x y su valor de pertenencia. Cuando U es continuo, A se escribe como en (2.19) donde el símbolo de integración no denota una integral sino la colección de $x \in U$ asociados a $\mu_A(x)$. De forma similar, si U es discreto, A se escribe como en (2.20) donde el símbolo de sumatoria no denota una suma sino la colección de $x \in U$ asociados a $\mu_A(x)$.

$$A = \{(x, \mu_A(x)) \mid x \in U\} \tag{2.18}$$

$$A = \int_U \frac{\mu_A(x)}{x} \tag{2.19}$$

$$A = \sum_U \frac{\mu_A(x)}{x} \tag{2.20}$$

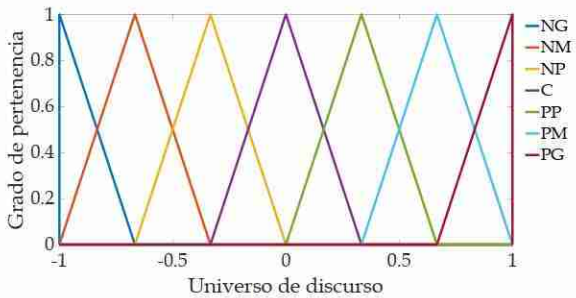


Figura 2.3. Ejemplos de conjuntos difusos: siete conjuntos difusos (NG: negativo grande, NM: negativo medio, NP: negativo pequeño, C: cero, PP: positivo pequeño, PM: positivo medio y PG: positivo grande), con sus respectivas funciones de pertenencia tipo triangular, caracterizan al universo de discurso U.

Las propiedades que un conjunto difuso trata de caracterizar usualmente son difusas, por ejemplo: “los números cercanos a cero”, la cual no es una descripción precisa. Siguiendo esta línea, diferentes funciones de pertenencia pueden caracterizar a la misma descripción. Pero, a pesar de que dos funciones de pertenencia caractericen a la misma descripción, éstas corresponden a dos conjuntos difusos diferentes, ya que un conjunto difuso tiene una correspondencia uno a uno con su función de pertenencia (los conjuntos difusos equivalen a su función de pertenencia). La elección de la función de pertenencia se basa en varios factores, aunque generalmente se construyen a partir del conocimiento de la descripción o del ajuste con los datos de dicha descripción.

Es evidente que varios conjuntos difusos pueden describir a U, como se observa en la Figura 2.3, entre más conjuntos difusos haya mejor sera la descripción de U. Asimismo, los conceptos y operaciones entre estos conjuntos son variados, como ejemplos se tienen: el soporte, el centro, la altura, el punto de cruce, el corte alfa, el inverso de un conjunto difuso, conjunto difuso convexo, conjunto difuso normal, normas-T (intersección de conjuntos) y normas-S (unión de conjuntos). Estos conceptos se describirán, sin entrar en gran detalle, brevemente en seguida.

A grandes rasgos, para un conjunto difuso A en U con x elementos, los conceptos anteriores se pueden definir de la siguiente forma: el inverso, la función de pertenencia del conjunto inverso $\bar{A}$, es $\mu_{\bar{A}}(x)=1-\mu_A(x)$; el soporte se define como $\mu_A(x)\geq 0$, si el soporte es igual a 0 se trata de un conjunto vacío y si solo se trata de un elemento se tiene un conjunto “singleton”; el punto de cruce se define como $\mu_A(x)=0.5$; el centro es el valor medio de todos los puntos en los que la función de pertenencia alcanza su máximo, en el caso de que este valor no sea finito, es decir, sea igual a infinito positivo (negativo) entonces el centro será definido como el menor (mayor) punto que alcanza dicho valor máximo; la altura es el valor de pertenencia máximo (entre 0 y 1), si un conjunto difuso tiene altura igual a 1 se dice que este conjunto es normal.

$$A_\alpha=\left\{x\in U\,|\,\mu_{A_\alpha}(x)\geq\alpha\right\} \tag{2.21}$$

$$\mu_A[\lambda x_1+(1-\lambda)x_2]\geq\min[\mu_A(x_1),\mu_A(x_2)],\quad \forall x_1,x_2\in\mathbb{R}^n\ \&\ \forall\lambda\in[0,1] \tag{2.22}$$

Para la definición de corte alfa, A_α, se sigue a (2.21), mientras que para la definición de convexidad se usa a (2.22). La definición de norma-T y norma-S hace uso de un segundo conjunto

difuso B: la norma-S de dos conjuntos difusos, correspondiente a su unión, se establece como $\mu_{A\cup B} = \max(\mu_A(x), \mu_B(x))$; la norma-T de dos conjuntos difusos, correspondiente a su intersección, se establece como $\mu_{A\cap B} = \min(\mu_A(x), \mu_B(x))$. Ya que se habla de operaciones entre conjuntos, los conjuntos difusos cumplen también con las leyes de Morgan. Cabe destacar que estos son conceptos básicos, en la literatura existen más operaciones, relaciones y conceptos para conjuntos difusos, incluso otras formas de definir la norma-S y la norma-T.

Si una variable bien definida puede ser descrita con palabras en "lenguaje natural" (por ejemplo: "la temperatura de hoy es alta", la variable es la temperatura de hoy y se describe con la palabra alta), esta variable se denomina "variable lingüística". En una variable lingüística, las palabras que la describen son caracterizadas por conjuntos difusos definidos en el universo de discurso, U, en el que la variable está definida. La Figura 2.3 puede representar cualquier variable lingüística, por ejemplo temperatura en un rango acotado de $[-1,1]$; U serán todos los valores que la temperatura puede tomar en tal rango, los valores lingüísticos que la temperatura puede tomar son los siete conjuntos difusos con una semántica que los relaciona con U.

El conocimiento humano, en su representación más simple, tiene como concepto fundamental a las variables lingüísticas; cuando se usan sensores para tomar mediciones de una variable estos arrojan números, cuando se le pregunta a una persona evaluar alguna variable ésta lo hace con palabras. Por tanto, al introducir el concepto de variable lingüística es posible hacer descripciones vagas en lenguaje natural con términos matemáticos precisos. Esto es el primer paso para incorporar el conocimiento humano, el razonamiento, dentro de los sistemas de ingeniería.

SI (proposición difusa), ENTONCES (proposición difusa) (2.23)

SI (proposición difusa), ENTONCES $y = ax_1 + b, \quad x_1 \in U_1 \ \& \ a,b,y \in \mathbb{R}$ (2.24)

x_1 es $A_1, \quad x_1 \in U_1 \ \& A_1$ es un conjunto difuso en U_1 (2.25)

x_1 es A_1 y x_2 es $A_2, \quad x_1, x_2 \in U_1 \ \& A_1, A_2$ son conjuntos difusos en U_1 (2.26)

x_1 es A_1 o x_2 es A_2 (2.27)

x_1 no es A_1 o x_1 es A_2 (2.28)

x_1 es A_1 y x_1 no es A_2 (2.29)

Los sistemas difusos, encargados de la incorporación del razonamiento con sistemas de ingeniería, son construidos a partir de reglas del tipo SI-ENTONCES. Las reglas SI-ENTONCES son, estados condicionados, descritos por (2.23) o (2.24), formados por proposiciones difusas que relacionan una variable lingüística con un conjunto difuso, (2.25)-(2.29). Las conjunciones usadas en las proposiciones “y”, “o” y “no” son representadas por medio de la aplicación de la norma-T, la norma-S y el complemento (inverso) al valor de la función de pertenencia de las variables lingüísticas con su conjunto difuso, respectivamente.

Se tienen dos tipos de proposiciones: atómicas (2.25) y compuestas (2.26)-(2.29). Las proposiciones atómicas son estados aislados, mientras que las proposiciones compuestas (varias proposiciones atómicas) relacionan dos o más variables lingüísticas que pueden estar en el mismo o en diferentes universos de discurso; las proposiciones compuestas deben entenderse como relaciones difusas. La interpretación de las reglas difusas depende en gran medida de como se definen las conjunciones (normas-T, normas-S y complementos). Y en general, aunque no se limita, se usan a (2.25) y (2.26) para la construcción de sistemas difusos.

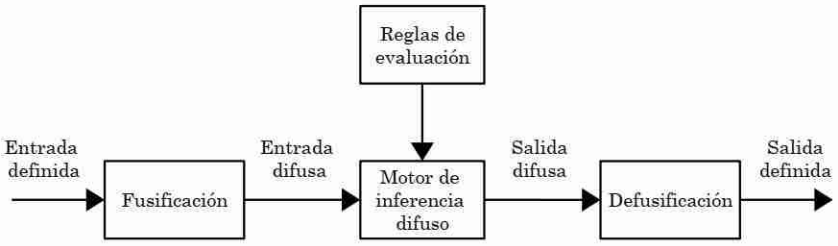


Figura 2.4. Diagrama representativo de un sistema difuso.

Comúnmente, el funcionamiento de un sistema difuso es descrito por la Figura 2.4, en donde se aprecian cuatro elementos principales:

- Fusificación: se establece una relación entre los valores de las variables lingüísticas, que se usan como entrada del sistema difuso, y los conjuntos difusos de entrada en el universo de discurso en el que están definidos (grado de pertenencia de cada valor con respecto a cada conjunto), ver Figura 2.3. Esto para convertir las entradas bien definidas en entradas difusas.
- Reglas de evaluación: conjunto de reglas SI-ENTONCES, que se construyen a partir de la experiencia y el conocimiento del fenómeno que el sistema difuso va a describir. Estas reglas resumen el comportamiento causa-efecto del sistema en cuestión.

- Motor de inferencia difuso: interpreta las reglas de evaluación para generar una respuesta (salida difusa). Se aplica el operador difuso correspondiente a las entradas difusas (acorde a la conjunción usada en las reglas), luego se aplica una función de inferencia para determinar la salida difusa de cada regla y éstas se combinan en un solo conjunto difuso (agregación) que es la representación del sistema ante esas entradas. Gupta y Qi [25] hablan de esto aplicado a las normas-T y Tserkovny [26] propone nuevas formas de hacer dicha inferencia.
- Defusificación: convierte a la salida difusa generada por el motor de inferencia en un valor bien definido, casi siempre considerando la geometría del conjunto difuso resultante. Algunos métodos para ello son: métodos del máximo, media ponderada, singleton y centroide.

No hay un proceso formal para la elección de funciones de pertenencia en un sistema difuso, aunque se han desarrollado formas estándar como las triangulares, trapezoidales, sigmoideas y gaussianas. Pueden usarse diferentes formas y, dependiendo de cuál se use, se pueden obtener diferentes resultados (de esto hablan en detalle Ali et al. [27]).

Cuando se construye un sistema difuso con reglas de la forma (2.23), se dice que se trata de un sistema difuso puro. Si un sistema difuso se construye con reglas de la forma (2.24), se trata de un sistema difuso Takagi-Sugeno-Kang (apellidos de las personas que desarrollaron ese algoritmo, también se le puede referir como TSK o solo TS). La diferencia fundamental entre estos sistemas difusos está en la forma en que realizan la inferencia; un sistema difuso puro usa procesos de inferencia de tipo Mamdani y un sistema difuso TSK utiliza procesos de tipo Sugeno.

La idea básica de la inferencia de Mamdani nace al querer crear un método de control de sistemas, por medio de la sinterización de un conjunto de reglas lingüísticas obtenidas de la experiencia de operadores humanos, en donde la salida de cada regla es un conjunto difuso. En este tipo de inferencia, se tienen bases de reglas más intuitivas y fáciles de entender. El resultado de cada regla es un conjunto difuso derivado de la función de pertenencia de salida y el método de implicación que se usa en la inferencia, estos conjuntos difusos de salida se combinan en un único conjunto difuso utilizando un método de agregación. Luego, para calcular un valor de salida, el conjunto difuso de salida combinada se desfusifica utilizando un método de desfusificación.

Por su parte, la inferencia tipo Sugeno utiliza funciones de pertenencia de salida singleton, que son constantes o una función lineal de los valores de entrada. El proceso de defusificación

para este tipo de inferencia es más eficiente en comparación con el de Mamdani, ya que usa un promedio ponderado de algunos puntos de datos en lugar de calcular un centroide de una área. Dada la dependencia lineal de cada regla a las variables de entrada, este método es ideal para actuar como supervisor de múltiples controladores lineales que se aplican, respectivamente, a diferentes condiciones de operación en un sistema dinámico no lineal. Del mismo modo, esto es adecuado para modelar sistemas no lineales mediante la interpolación entre múltiples modelos lineales.

Resumiendo, la inferencia de tipo Mamdani es intuitiva, adecuada para trabajar con el conocimiento humano y sus reglas son fáciles de interpretar en ese aspecto, pero es complicada de implementar computacionalmente hablando. La inferencia de tipo Sugeno es adecuada para realizar análisis matemáticos, es computacionalmente eficiente y por ende fácil de aplicar, garantiza una superficie de salida continua y trabaja bien con algoritmos de optimización y adaptables, así como algoritmos lineales como el control PID. Esto, acorde a Kaur y Kaur [28] y Blej y Azizi [29].

Los sistemas difusos, para su análisis, se tratan como sistemas de tipo MISO (siglas en inglés de “multi-input single-output”, multi-entrada una-salida) por simplicidad; un sistema MIMO (siglas en inglés de “multi-input multi-output”, multi-entrada multi-salida) siempre puede descomponerse en una colección de sistemas MISO. Las reglas de evaluación de un sistema difuso son: completas, si ningún punto en el espacio de entrada tiene un valor de pertenencia cero en la parte de los antecedentes; consistentes, si no hay reglas con los mismos antecedentes, pero diferentes consecuencias; continuas, si no existen reglas cercanas entre sí cuyos conjuntos difusos en la parte de consecuencias tengan intersecciones vacías.

La aportación principal de los sistemas difusos es el dotar de un procedimiento sistemático para la transformación de cualquier conjunto de variables lingüísticas en un mapeo de funciones no lineales. Asimismo, tomando en cuenta lo anterior, un sistema difuso es capaz de aproximar funciones en tiempo continuo, al igual que funciones en tiempo discreto, con un cierto grado de error, lo que se aprovecha para modelar sistemas en ingeniería (modelado tipo caja blanca). Existen diversas ventajas y desventajas del uso de estos sistemas, en algunos casos serán más atractivos de usar que otras técnicas de estimación y viceversa.

Dentro de las ventajas del uso de sistemas difusos se tienen: fáciles de entender (conceptos matemáticos sencillos), flexibles (pueden ser complementarios a otros algoritmos), tolerantes a

datos imprecisos y se construyen con base en la experiencia humana en vez de seguir en rigor leyes físicas (ecuaciones). Y dentro de las desventajas están: requieren gran número de reglas para realizar una buena aproximación, no hay un criterio para determinar la elección y número exacto de reglas, así como también la construcción de sistemas difusos puede resultar en varias combinaciones entre funciones de pertenencia, métodos de fusificación, métodos de inferencia y métodos de defusificación (es decir, no hay una forma óptima de determinar el mejor sistema difuso).

El número de reglas en un sistema difuso incrementa exponencialmente con el número de variables que se usan en su entrada, además, los paramentos matemáticos de las fórmulas que describen a tales sistemas también incrementan de la misma forma. Se entiende entonces que el conocimiento requerido para crear el sistema difuso aumentará de forma exponencial, lo que puede afectar la interpretación y seguimiento de las reglas. Para lidiar con este problema se han desarrollado los llamados sistemas difusos jerárquicos (Lee et al. [30] y Wang et al. [31]).

Un sistema difuso jerárquico es aquel que se compone de varios sistemas difusos, a los que se les denomina subsistemas difusos o unidades de lógica difusa, interconectados de forma jerárquica. Así, se evita el uso de sistemas difusos con grandes dimensiones, sustituyéndolos por sistemas difusos de menor dimensión interactuando entre sí, obteniendo resultados muy similares. Aunque se mejora el diseño, también se tienen algunos inconvenientes, como que se pierde parte de la transparencia e interpretación de las reglas difusas en comparación con los sistemas difusos convencionales.

Para lidiar con el problema de la creación de reglas difusas se han realizado varias propuestas, las cuales van desde usar técnicas externas para hacer estimaciones de las reglas hasta procesos de aprendizaje las mismas. De esto último han surgido los ANFIS, los cuales tienen una gran aceptación por su versatilidad y fácil aplicación. Los ANFIS son una propuesta que no requiere del conocimiento del sistema a estudiar, sino que este conocimiento puede ser adquirido durante su funcionamiento, lo que significa una gran mejora a los sistemas difusos tradicionales.

2.1.5. ANFIS

Derivado de la dificultad para encontrar la manera adecuada de representar el conocimiento humano en reglas de evaluación, además de la necesidad de ajustar las funciones de pertenencia para hacer eficiente el funcionamiento de un sistema difuso, en general, nacen los ANFIS. Un ANFIS

(siglas en inglés de “adaptive network based fuzzy inference system”, sistema de inferencia difusa basado en red(es) adaptable(s)) puede generar un sistema difuso adecuado, por medio de algún algoritmo de aprendizaje.

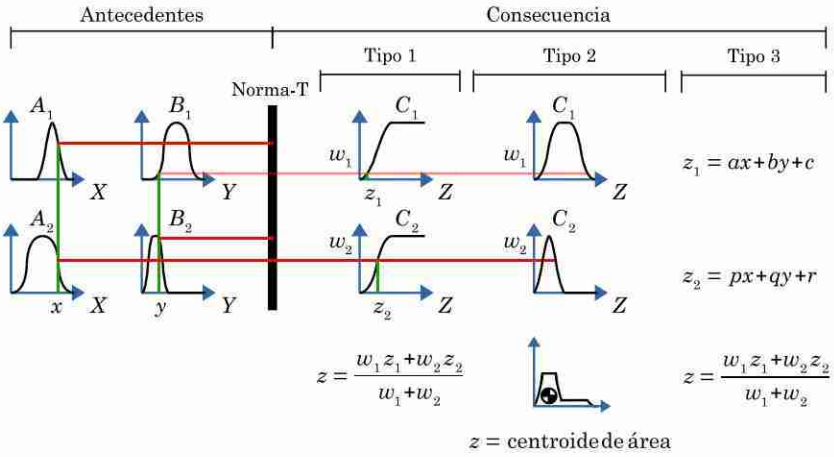


Figura 2.5. Esquema de razonamiento para los tres tipos principales de ANFIS: cada uno a través de un sistema difuso de dos entradas (x, y), una salida (z), con dos reglas, con una norma-T para la obtención de sus antecedentes (w_1, w_2) y su respectivo razonamiento que involucra a sus consecuencias (z_1, z_2).

En trabajos como los de Jang [32] y Walia et al. [33] se muestra a detalle la construcción de los ANFIS. Estos sistemas pueden diseñarse de la misma forma que un sistema difuso normal, hasta la parte del motor de inferencia y las reglas de evaluación; se busca interpretar los antecedentes de las reglas SI-ENTONCES de tal manera que se pueda generar una consecuencia adecuada, por medio de un aprendizaje (a esto se le refiere como razonamiento). Para los ANFIS, ver Figura 2.5, existen tres tipos principales de razonamiento y basándose en ellos se puede hacer una clasificación de estos sistemas:

- Tipo 1: la salida del sistema es el promedio ponderado de la salida de cada regla, evaluadas en el universo de discurso de salida, por su correspondiente antecedente normalizado.
- Tipo 2: la salida del sistema se obtiene aplicando un operador de máximo a las salidas difusas calificadas (obtenidas de los antecedentes de las reglas de evaluación y sus funciones de pertenencia de salida correspondientes). Esta salida es difusa y se debe de usar algún método de defusificación, como los de tipo centroide, para obtener el valor definido de la salida.
- Tipo 3: la salida de cada regla de evaluación es una combinación lineal de las variables de entrada más un término constante. La salida del sistema es el promedio ponderado de la

salida de cada regla por su respectivo antecedente normalizado. En este tipo de sistemas solamente se usan reglas difusas del tipo TSK.

La diferencia fundamental, entre los tres tipos de ANFIS, se nota en la obtención del valor definido de la salida del sistema difuso.

Salta a la vista que un ANFIS puede operar de la misma forma que una red neuronal, en el sentido de que éste puede cambiar sus parámetros por medio de cualquier algoritmo de aprendizaje. En la Figura 2.6, se pueden apreciar dos tipos de topologías de ANFIS que se asemejan a las redes neuronales de conexiones hacia adelante, pero en este caso cada nodo en las redes realiza una función en particular dependiendo de las señales entrantes y de los parámetros de cada nodo.

Para la construcción de ANFIS se usan dos tipos de nodos: cuadrados y circulares. Los nodos cuadrados (nodos adaptables) tienen parámetros, mientras que los circulares (nodos fijos) no tienen. Cuando los ANFIS se someten a entrenamiento solo los nodos adaptables se ven afectados por esto, ya que los nodos fijos solo representan operaciones. Luego, las consideraciones que se hacen para aplicar un algoritmo de entrenamiento en redes neuronales pueden aplicarse a los ANFIS, si se respeta la configuración de nodos anterior.

Estructuralmente, observando la Figura 2.5, un ANFIS de cualquier tipo (Figura 2.6) es visto como una red neuronal de cinco capas: la primera capa se encarga de fusificar a las entradas que se usan en el sistema, la segunda capa combina los resultados de la capa anterior por medio de una norma-T, en la tercera capa se normalizan los resultados de la anterior capa obteniendo los pesos de cada regla, la cuarta capa calcula la salida de cada regla y la multiplica por su correspondiente peso generado en la capa anterior, y finalmente en la quinta capa se suman las salidas anteriores para obtener el valor de salida general del sistema lo que corresponde a la defusificación.

$$O_i^1 = \mu_{A_i}(x) \quad \& \quad O_i^2 = \mu_{B_i}(y), \quad \text{con } i = 1,2 \tag{2.30}$$

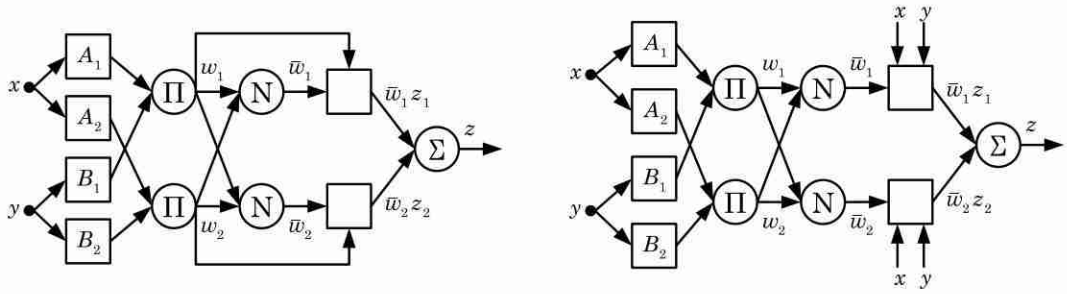
$$w_i = \prod\left(O_i^1 O_i^2\right), \quad \Pi \text{ representa una norma-T} \tag{2.31}$$

$$\bar{w}_i = \frac{w_i}{w_1 + w_2} \tag{2.32}$$

$$O_i^3 = \bar{w}_i z_i \tag{2.33}$$

$$z = \sum_{j=1}^{i} O_j^3 \tag{2.34}$$

Por ejemplo, para un ANFIS tipo 3 (Figura 2.6 (b)), cada una de sus capas se puede caracterizar con cada ecuación del conjunto (2.30)-(2.34), respectivamente. Para los ANFIS tipo 1 (Figura 2.6 (a)), la descripción anterior también se aplica; pero como se aprecia en la Figura 2.5, la parte de z_i cambia, dado que se consideran las funciones de pertenencia en la salida. Los ANFIS tipo 2 son más complicados de caracterizar, aunque con una modificación en las últimas capas de la red en la Figura 2.6 (a) puede conseguirse una aproximación de estos (la modificación consiste en cambiar los bloques por una versión discreta del método de defusificación usado).



(a) (b)

Figura 2.6. ANFIS en forma de red neuronal: (a) ANFIS tipo 1, (b) ANFIS tipo 3.

Entonces, (2.30) y (2.33) son las únicas bajo modificación y por tanto deben de tener condiciones iniciales en función de su estructura interna (revisando las Figuras 2.5 y 2.6). Cabe mencionar que los ANFIS tipo 3 son los más fáciles de usar, por su mayor parecido a una red neuronal, lo que significa que son los sistemas más empleados. Dadas sus características, los ANFIS tipo 1 son usados en menor medida y los ANFIS tipo 2 son raramente empleados. Cualquier tipo de entrenamiento usado en redes neuronales, de conexiones hacia adelante, puede usarse en los ANFIS según Horikawa et al. [34] y Jang [35].

Al igual que los sistemas difusos normales, estos sistemas tienen la desventaja de que para un mejor funcionamiento requieren un número mayor de reglas; para abordar esta desventaja se puede hacer uso de sistemas jerárquicos o métodos más sofisticados como el propuesto por Panella [36]. Desventajas propias de los ANFIS son: no hay una forma adecuada de seleccionar sus condiciones iniciales, además de que su entrenamiento puede ser lento debido a que se trata con redes de conexiones hacia adelante, así como que su estructura no pueda tener un sentido lógico.

2.2. Estado del arte

Ya establecidos los fundamentos de la teoría, en esta sección se habla de trabajos recientes que involucran el desarrollo de las redes neuronales y de los sistemas difusos. Con ello, se determinó la composición del algoritmo de red difuso-neuronal que se requiere para el cumplimiento de los objetivos de esta tesis.

2.2.1. Aplicaciones recientes que involucran redes neuronales

Como se explicó en la sección anterior, el desarrollo de redes neuronales tiene la tendencia a la creación o establecimiento de redes profundas que hacen mejor uso de la información, ejemplo de esto es lo mostrado por Prieto et al. [37] y Salehinejad et al. [38]. En seguida, se hablará de diversos trabajos que hacen uso de redes neuronales, destacando ciertas características como lo son: mejoras, nuevas topologías, y combinaciones entre topologías existentes. También se hace mención de las áreas en las que se están aplicando dichas redes neuronales, para tener una idea de su impacto en los años recientes y hacia donde se dirige su desarrollo.

Dentro del área del procesamiento de imágenes se tienen varios trabajos, destacando el de Howard et al. [39], en donde se desarrolló una nueva estructura de red neuronal la cual se denominó "MobileNet", que toma como base a las redes convolucionales, pero se busca la simplificación y reducción del tamaño en su estructura interna. Esta red fue pensada para aplicaciones de visión móviles e integradas, como lo son la identificación y clasificación de objetos, identificación de atributos faciales y geo-localización a gran escala dando mejores resultados (en cuanto a tamaño y eficacia) que las redes convolusionales previas a ésta.

En el mismo sentido, pero enfocándose en la detección facial, Wang y He [40] propone un algoritmo que combina redes convolucionales y a la técnica de "comparación de plantillas". Comparación de plantillas, es una técnica dentro del procesamiento de imágenes para encontrar pequeñas partes de una imagen que coincidan con una imagen plantilla. Los resultados que se obtuvieron haciendo uso de este nuevo algoritmo fueron generados a una velocidad de convergencia aceptable, sin sacrificar la exactitud de los mismos, demostrando la compatibilidad de las redes neuronales con otras técnicas para mejorar los resultados que se tenían con anterioridad.

Otros trabajos que involucran redes convolucionales en el área del procesamiento de imágenes son los de Xiao et al. [41], Donahue et al. [42] y Shi et al. [43], teniendo en común la clasificación de objetos como problema a tratar. En el primer trabajo, se definen varios niveles de atención y se usan redes convolucionales de dos formas: para filtrar la información y para procesar la información filtrada, obteniendo un mejor desempeño en comparación de que si se hubiera usado una sola red. En el segundo y tercer trabajo, se combinan las redes convolucionales y una parte de las redes LSTM para hacer predicciones de manera eficiente a partir de la memoria de las redes.

Algunas aplicaciones en el área de control automático: en estimaciones de sistemas el trabajo de Khristoforov y Bochkarev [44] y Anderson et al. [45]; en el diagnóstico de fallas el trabajo de Ali et al. [46] y Lin et al. [47]; y en el control de sistemas las aportaciones de Le et al. [48] y Bao et al. [49]. En las tareas de estimación, se ordenan los datos que las redes van a usar de forma que sea fácil obtener un buen resultado. Mientras que en el control, las redes actúan como buscadores de parámetros para los algoritmos de control PID y modos deslizantes.

Siguiendo en la identificación de sistemas, trabajos como el de Faris et al. [50] y Bektas et al. [51], muestran que las redes de conexiones hacia adelante simples (estructuras convencionales) pueden tener un nivel de operación similar al de las nuevas propuestas de redes, pero se deben de desarrollar y aplicar entrenamientos más sofisticados en ellas.

Por parte de las redes recurrentes, Ogunmolu et al. [52] comparan redes de conexiones hacia adelante, recurrentes, GRU y dos versiones de LSTM (rápida y normal) para identificación de sistemas, resultando que las redes GRU y LSTM dieron los mejores resultados. Chung et al. [53] hicieron algo parecido pero solo con redes recurrentes, GRU y LSTM, concluyendo que las GRU y LSTM son superiores. Para modelos que usan gran cantidad de datos, Zaheer et al. [54] implementan redes LSTM y técnicas de estimación Bayesianas en conjunto con buenos resultados.

Más áreas en donde se han aplicado redes LSTM son la robótica (planeación de trayectorias y ubicación de dispositivos, Nicola et al. [55] y Liu et al. [56]), análisis de oraciones (Li y Jiang [57] y Coto-Jimenez et al. [58]), medicina (análisis de datos en robots médicos, Aviles et al. [59]) y control (regulación de temperatura, Wang et al. [60]). Intentos por mejorar la estructura inicial de la red LSTM pueden observarse en los trabajos de Wang [61] y Li et al. [62], donde se usa el concepto de convexidad y redes modulares, respectivamente.

Tabla 2.1. Comparación de características entre redes neuronales: se mencionan las ventajas y desventaja, a grandes rasgos, de las redes usadas en los trabajos citados en la Subsección 2.2.1.

Red	Ventajas	Desventajas
Conexiones hacia adelante	▪ Permiten asilar elementos de un sistema para su análisis. ▪ Buena para la selección y la clasificación de objetos.	▪ Dimensiones grandes. ▪ Tiempo de entrenamiento largo.
Recurrente	▪ Buena para la predicción de series de tiempo. ▪ El uso de memoria interna. ▪ Se adecua a aplicaciones de identificación de sistemas y control de sistemas. ▪ Tiempo de entrenamiento menor al de las redes de conexiones hacia adelante.	▪ Problemas con la transmisión de información en el entrenamiento. ▪ Debido a las recurrencias, se tienen algoritmos de entrenamiento más complejos en comparación a los de las redes de conexiones hacia adelante. ▪ Dimensiones grandes. ▪ Solo maneja memoria a largo plazo de lo que aprende.
Convolucional	▪ Una versión mejorada de las redes de conexiones hacia adelante. ▪ Entrenamiento rápido. ▪ Menores dimensiones.	▪ Es conveniente usar varias redes en vez de solamente una, es decir, es mejor usar varias estructuras de la misma red conectadas entre sí. ▪ El entrenamiento se complica por la operación de convolución.
LSTM	▪ Una versión mejorada de las redes recurrentes. ▪ Mejor manejo de los términos de memoria (memoria a corto y mediano plazo). ▪ Mejores relaciones de datos.	▪ Estructura más compleja. ▪ No se elimina del todo el problema de transmisión de información. ▪ Tiempo de entrenamiento similar a su versión anterior.
GRU	▪ Versión corta de una LSTM, menos complicada. ▪ Rendimiento superior a una red recurrente. ▪ Entrenamiento corto.	▪ Su desempeño no es tan bueno como el de una red LSTM. ▪ Su estructura puede variar mucho dependiendo de la tarea.
Híbrida	▪ Aprovecha las ventajas de las diferentes redes que se combinan en su estructura. ▪ La mayoría de las veces da resultados superiores a los que se obtienen con un solo tipo de red.	▪ En general, el entrenamiento se complica al incrementar la complejidad de la red. ▪ También se pueden heredar las desventajas (algunas o todas) de las redes usadas en su diseño.

A manera de resumen, en la Tabla 2.1 se listan las ventajas y desventajas (sin considerar alguna aplicación o tipo de entrenamiento en particular) de los tipos de redes neuronales citadas anteriormente. Con esto, se pueden emitir juicios rápidos y certeros sobre los diferentes tipos de redes neuronales, según se requiera.

2.2.2. Aplicaciones recientes que involucran sistemas difusos

Las aplicaciones de los sistemas difusos son numerosas, abundando principalmente en el área de control automático. Actualmente, este tipo de sistemas son muy utilizados en diversos campos, razón por la cual en esta subsección se hará mención de algunos trabajos recientes, y destacados, que emplean a los sistemas difusos para resolver diferentes problemáticas. De antemano, se puede decir que los sistemas difusos ejercen mayormente la función de analizar información para realizar una toma de decisiones.

Resaltan áreas como la medicina, con los trabajos de Sang et al. [63] y Aviles et al. [64] que usan sistemas difusos junto con otros algoritmos como redes LSTM y modos deslizantes (en adición con redes de conexiones hacia adelante), respectivamente, para controlar robots cirujanos, consiguiendo movimientos muy precisos y suaves. También se tienen trabajos con ANFIS, como la gestión de los recursos de emergencias (Yousefi et al. [65], con el apoyo algoritmos genéticos), el control de los parámetros de caminadoras en función del ritmo cardíaco (Lu et al. [66]) y el control de robots de rehabilitación (Huang et al. [67]).

Concerniente a las estimaciones basadas en análisis de datos, Li et al. [68] y Sri et al. [69] realizaron predicciones para el consumo de energía eléctrica, empleando redes LSTM con ANFIS jerárquicos y redes LSTM con sistemas difusos convencionales, respectivamente. Similar a lo anterior, Yao et al. [70] y Cheng et al. [71] proponen el uso de redes LSTM con sistemas difusos y sistemas ANFIS en conjunto con redes profundas, respectivamente, aplicadas al pronóstico de los parámetros de las corrientes de viento para aprovecharlas en la generación de energía eólica. Hosseinzadeh y Wolfs [72] hacen uso de las redes recurrentes y sistemas difusos, para estimar el tiempo de descarga en redes de distribución de energía.

En el mismo ánimo, Rashidy et al. [73] utilizan una estructura ANFIS jerárquica para la detección de fallas en servoválvulas hidráulicas. Entre los algoritmos para la detección de fallas,

pero no ideados para aplicaciones en específico, se encuentran las propuestas de: Zhang et al. [74] en donde se diseña un observador discreto de fallas con base en sistemas difusos; Wei et al. [75] que involucran en el diseño de controladores difusos la estimación de fallas en el sistema que se busca controlar.

Formas de usar sistemas difusos para tomar decisiones son discutidas por Yera y Martinez [76], mientras Fernandez et al. [77] hablan del desarrollo de algoritmos genéticos y sistemas difusos en ese sentido. Como ejemplos de esto: el trabajo de Kamal et al. [78] administra el acceso de seguridad a una planta de energía empleando ANFIS (apoyándose en redes LSTM), el trabajo de Huang et al. [79] predice eventos en el mercado financiero a través de sistemas difusos jerárquicos, y Boroushaki et al. [80] presentan el control de un reactor nuclear (estimando sus parámetros con redes neuronales) por medio de un sistema difuso. Ahmadian et al. [81] y Kostikova et al. [82] proponen algoritmos difusos para el análisis de las propiedades de sistemas.

Entre los trabajos enfocados al control automático, sin considerar aplicaciones específicas, se encuentran: el de Tang et al. [83] con el ajuste óptimo de controladores PID; para sistemas no lineales, Azimi et al. [84], Yu y Hu [85] y Ji et al. [86] muestran controladores difusos con condiciones suficientes de estabilidad basadas en la teoría de Lyapunov. Similarmente, Qiu et al. [87] hablan de la aplicación de sistemas difusos para sistemas conectados a redes de comunicaciones. Castillo et al. [88] comparan los tres tipos de ANFIS enfocándose en el tipo 2, y Heo et al. [89] establecen que los ANFIS pueden considerarse dentro de las técnicas de aprendizaje profundo.

Cabe destacar que aunque se mencionaran otros algoritmos aparte de los sistemas difusos, cada algoritmo ejerce una función distinta y son combinados para obtener la respuesta final en cada trabajo. Pero, dada la naturaleza de los sistemas difusos y otras técnicas de computación flexible descritas al inicio, estos algoritmos pueden aplicarse de tal manera que los sistemas difusos modifiquen su estructura interna conforme a estos. Se hablará de ello en la siguiente sección, particularmente de las redes neuronales que son las más compatibles con los sistemas difusos.

2.2.3. Sistemas híbridos de redes neuronales y sistemas difusos

Con los ANFIS se nota que, de manera relativamente sencilla, se pueden combinar las redes neuronales con los sistemas difusos, con la intensión de crear sistemas híbridos que tomen las

características principales de cada uno (Mitra y Hayashi [90] y Ding et al. [91]). Reflexionando al respecto, surge una analogía con las computadoras: las redes neuronales se asemejan al hardware, encargado de recolectar y exponer información, mientras que los sistemas difusos se asemejan al software, encargado de procesar y analizar la información para ofrecer resultados.

Tabla 2.2. Comparación de características entre sistemas difusos: se mencionan las ventajas y desventajas de los sistemas difusos de los trabajos citados en las Subsecciones 2.2.2 y 2.2.3.

Sistema	Ventajas	Desventajas
Difuso convencional	▪ Robustez a datos imprecisos e incompletos. ▪ Representa el conocimiento de los expertos en el tema o sistema con el que se trata.	▪ Dimensiones grandes, ya que no se sabe con cuáles y cuántas reglas se caracteriza correctamente a un sistema. ▪ Diseño complicado.
Difuso y redes neuronales	▪ Menores dimensiones que un simple sistema difuso. ▪ Se analizan diferentes partes de un sistema de diferentes formas, usando la estructura más conveniente en cada caso.	▪ Las redes están desacopladas del sistema difuso. ▪ Se tienen que entrenar a las redes neuronales independientemente de los sistemas difusos.
ANFIS	▪ Se simplifica la definición de las reglas difusas. ▪ Los parámetros del sistema difuso se ajustan por medio de un algoritmo de entrenamiento, similar a las redes neuronales.	▪ El entrenamiento puede ser largo, ya que se tratan de redes similares a las redes de conexiones hacia adelante convencionales. ▪ Aún esta presente el problema del número de reglas.
Difuso-neuronal o neuro-difuso	▪ Aprovecha las ventajas de las diferentes redes neuronales en la estructura interna del sistema difuso. ▪ Versatilidad, se puede expresar el conocimiento de forma similar a un experto sin ser un experto. ▪ Entrenamiento corto.	▪ Infinidad de combinaciones entre los parámetros difusos y las redes neuronales. ▪ La interpretación del conocimiento representado puede perderse parcialmente a causa del uso de las redes neuronales. ▪ Se deben ajustar varios parámetros al mismo tiempo.

La combinación de redes neuronales y sistemas difusos suele denominarse de varias formas: sistemas difusos dinámicos, sistemas neuro-difusos, redes (o sistemas) difuso-neuronales, sistemas FIS (siglas en inglés de “fuzzy inference system”, sistema de inferencia difuso). La denominación

es indistinta, en la literatura todas ellas son tratadas como sinónimos, pero puede ser influenciada por la parte dominante en la combinación. Ventajas y desventajas de los sistemas difusos citados en este trabajo se muestran en la Tabla 2.2, resaltando el hecho que los sistemas difuso-neuronales tienen el mejor desempeño, en general, de entre todos los sistemas mencionados en la tabla.

Siendo el caso de la identificación de sistemas, Babuška y Verbruggen [92] reportan que las redes neuronales y los sistemas difusos, incluidos los ANFIS, pueden aproximar funciones correspondientes a sistemas no lineales mejor que las técnicas de estimación convencionales. La diferencia entre los algoritmos mencionados es que las redes neuronales no necesitan conocimiento explícito para su funcionamiento, porque no manejan información afín al sistema, sino información que aprenden codificada en sus parámetros. Por el contrario, los sistemas difusos y ANFIS muestran la información de un sistema de forma clara por medio de sus reglas SI-ENTONCES.

Luego, un sistema difuso-neuronal combina la semántica de un sistema difuso con la capacidad de aprender de las redes neuronales. Los sistemas difusos TSK y los ANFIS tipo 3 son los que tienen la mayor compatibilidad con las redes neuronales, esto porque la parte de consecuencia en sus reglas de evaluación corresponde a combinaciones lineales de las variables de entrada (sistemas TSK de primer orden); estas ecuaciones lineales son interpretadas como una aproximación, conformada de varias partes, a una función no lineal. Las estructuras de la Figura 2.6 pueden fácilmente ser ajustadas para que en alguna de sus capas, ya sea en la primera o en las últimas, se adapte una red neuronal para mejorar el entrenamiento de los parámetros de los ANFIS.

Primeros trabajos destacados sobre la idea anterior, está lo realizado por Takagi y Hayashi [93], Buckley y Hayashi [94], Kasabov [95] y Benítez et al. [96], en donde se observan estructuras similares a los ANFIS, pero construidas a partir de redes de conexiones hacia adelante tradicionales. Trabajando a partir del modelo ANFIS tipo 3 más directamente, Wu y Er [97] hacen modificaciones para implementar, semejante a los trabajos antes mencionados, redes de conexiones hacia adelante en las capas internas de un ANFIS.

Con la introducción de funciones no lineales para la parte de consecuencia de los sistemas difusos TSK (como lo realizaron Dong et al. [98]), en vez de funciones lineales, es posible utilizar redes neuronales para estimar las consecuencias en las reglas de evaluación con mayor facilidad. Bajo esta premisa, Ganjefar y Tofighi [99] usaron redes neuronales de onda corta (wavelet network)

para estimar las consecuencias de las reglas de evaluación de un sistema TSK, el cual se usó en tareas de identificación de sistemas. En trabajos recopilatorios, como los de Viharos y Kis [100] y Shihabudheen y Pillai [101], se habla de estructuras de sistemas difuso-neuronales, con las características que se han mencionado, a manera de resumen.

Aplicaciones, en diversos campos, de varios tipos de sistemas difuso-neuronales se tienen a los trabajos de: Bhuyan et al. [102] con pruebas de confiabilidad a programas de cómputo recién creados, Pratama et al. [103] y Pratama et al. [104] con el procesamiento de grandes flujos de datos, Mastorocostas e Hilas [105] con predicciones de datos en telecomunicaciones, Schatz et al. [106] usan varios sistemas para el diagnóstico de pacientes usando sus signos vitales, Chang [107] con la identificación de caracteres a mano alzada, Li y Lin [108] con la predicción de series de tiempo caóticas, Krleža y Fertalj [109] con estructuras jerárquicas para analizar información en forma de grafos.

Otros trabajos destacados son: Obo et al. [110] en el área de la robótica médica para reconocimiento de posturas humanas; Tang et al. [111] con la predicción del flujo de tráfico vehicular; Zhao et al. [112] con el modelado y construcción de un nuevo sensor para un determinado proceso químico. Enfocados en el control automático, se han desarrollado varios trabajos explicando cómo se aplican o se pueden llegar a aplicar estructuras particulares de sistemas difuso-neuronales, siendo el caso en años recientes de los trabajos de Lee y Teng [113], Abiyev [114], Baruch et al. [115], Lee et al. [116], Zhang [117] y Zahedi y Zahedi [118].

2.2.4. Conformación de la solución propuesta a la problemática de la tesis

Considerando a las referencias citadas a lo largo del presente capítulo, así como las Tablas 2.1 y 2.2, es posible ver la evolución de las redes neuronales, los sistemas difusos y las redes difuso-neuronales. Esta evolución va desde el diseño de sistemas similares a los ANFIS a partir de redes neuronales, mejorar la estructura de los ANFIS introduciendo ciertos tipos de redes neuronales en su estructura interna, hasta diseñar redes difuso-neuronales completamente diferentes a un ANFIS. Lo que ha dado como resultado nuevas estructuras de redes neuronales apoyadas en sistemas difusos, modelos más sofisticados para identificación y control de sistemas, lo que es parte de las motivaciones de este trabajo, explicadas previamente en el Capítulo 1.

A partir de esto, surgió la idea de que una red difuso-neuronal, con los nuevos desarrollos en este campo, podría implementarse en el área de control automático, en conformidad a los requerimientos establecidos en el capítulo anterior. En el entendido que una red neuronal puede modificar sus parámetros en función de procesos de aprendizaje, mientras que un sistema difuso puede ofrecer una interpretación estructurada, con reglas causa-efecto, a lo aprendido por la red. De esta forma, se pueden obtener modelos tipo caja gris, ya sea para identificar o controlar sistemas.

Trabajos previos, a esta idea, son los ANFIS. Un ANFIS, estructuralmente hablando, puede ser visto en dos partes: la obtención de antecedentes y la obtención de consecuencias de las reglas causa-efecto que describen a un sistema. Para calcular cada una de estas partes, se emplean algoritmos que son bastante similares a las redes neuronales, siendo que la parte de los antecedentes de las reglas es más fácil de obtener en comparación a la parte de las consecuencias.

La dificultad para hallar las consecuencias de las reglas causa-efecto en un ANFIS puede resolverse al aplicar redes neuronales profundas, ya que como se menciona en las Subsecciones 2.1.3 y 2.2.1, este tipo de sistemas tienen un mayor rendimiento en comparación a sus contrapartes convencionales. Además, las redes profundas exhiben un mejor manejo de la información que utilizan, ya que están compuestas de varias secciones que facilitan la transmisión de información en la estructura interna de las redes.

Entonces, se propone la integración de una nueva estructura de red difuso-neuronal, la cual se nombró como Red Recurrente LSTM Difusa. A gran escala, la Red Recurrente LSTM Difusa es el resultado de la implementación de un sistema difuso y una red neuronal profunda LSTM; en el sentido práctico, la inferencia (obtención de la parte de las consecuencias en las reglas difuso) del sistema difuso usa una red LSTM para mejorar dicho proceso.

Se seleccionó como base de la propuesta de solución, de la problemática planteada al principio del documento, a la estructura de las redes LSTM porque, históricamente, las redes neuronales recurrentes ofrecen mejores resultados al trabajar con modelos secuenciales, que son el tipo los modelos que se usan en el área del control automático. La red LSTM, al resolver varias problemáticas presentes en las redes neuronales recurrentes convencionales, fue el algoritmo más llamativo para trabajar, por lo que el desarrollo de la nueva estructura giró en torno a las características de esta red.

Entonces, en la propuesta de solución se aprovechan las ventajas del manejo de información de la red LSTM en la producción de reglas difusas. Los resultados obtenidos con la red propuesta se representan como relaciones causa-efecto, lo que es más significativo que el simple uso de los parámetros de una red LSTM en solitario, los cuales no tienen una interpretación clara que se relacione con el sistema que están caracterizando. La nueva red genera una mejor aproximación al modelo o función que caracteriza a un sistema, según sea requerido por la tarea de identificación y control de sistemas en la que se llegue a emplear. El desarrollo de la estructura de la Red Recurrente LSTM Difusa se muestra en el siguiente capítulo.

2.3. Conclusiones del capítulo

Se dio una visión de las redes neuronales y los sistemas difusos útil para el presente trabajo, además de comparar las características de varios tipos de redes neuronales y sistemas difusos. A partir de ello, se determinó que, para resolver la problemática planteada en el capítulo anterior, se tiene que desarrollar una estructura de red difuso-neuronal que involucre el uso de redes LSTM.

Capítulo 3

Red Recurrente LSTM Difusa para la identificación de sistemas no lineales

Con un panorama de las redes neuronales, sistemas difusos y la combinación de ambos algoritmos, se puede entender mejor el desarrollo de este trabajo. Como se mencionó en el capítulo anterior, se requiere establecer una red difusa-neuronal que aproveche las características de una red LSTM, por lo que en este capítulo, se explica la conformación de una red LSTM y a partir de ello se define la estructura deseada. Además, se comparó el nuevo algoritmo con otros similares en la identificación de sistemas no lineales.

3.1. Estructura de una red LSTM

Como se dijo en la Subsección 2.1.3, la red LSTM, desarrollada por Hochreiter y Schmidhuber [119] y Gers et al. [120], es considerada como una estructura mejorada de las redes recurrentes tradicionales. Cuando se introdujo por primera vez, en los 90, esta red no llamó mucho la atención. Fue descartada debido a su estructura compleja y a que necesita de más recursos para trabajar, en comparación con una red recurrente normal. Por lo que hasta hace algunos años, a inicios de la década del 2010, se retomó su desarrollo ya la tecnología de la época lo permite.

Gracias al continuo desarrollo de las tecnologías de la informática, que permiten a los procesadores de las computadoras realizar operaciones matemáticas con mayor eficacia, fue posible

el uso factible de las redes LSTM. Como es notorio en la Subsección 2.2.1, las redes LSTM han logrado una gran aceptación y un desempeño mejor en el manejo de datos, en comparación a las redes recurrentes convencionales. La estructura básica de una red LSTM se muestra en la Figura 3.1, y en (3.1)-(3.6), donde saltan a la vista varias etapas en las que los datos usados por la red son tratados de manera distinta.

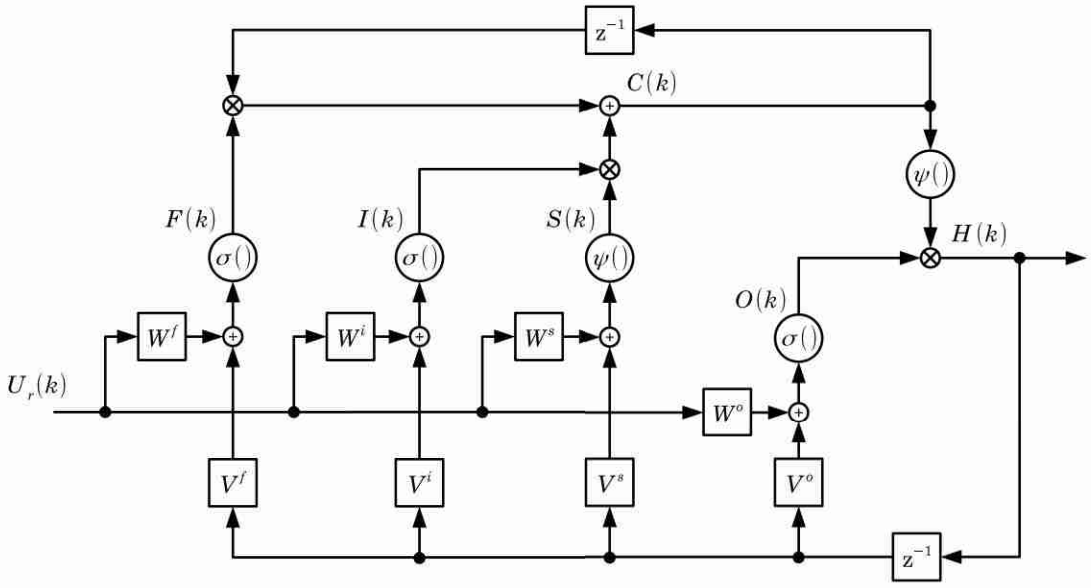


Figura 3.1. Estructura básica de una red LSTM.

Ecuaciones representativas de una red LSTM:

$$f_t = \sigma(u_t, h_{t-1}) \tag{3.1}$$

$$i_t = \sigma(u_t, h_{t-1}) \tag{3.2}$$

$$s_t = \psi(u_t, h_{t-1}) \tag{3.3}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes s_t \tag{3.4}$$

$$o_t = \sigma(u_t, h_{t-1}) \tag{3.5}$$

$$h_t = o_t \otimes \psi(c_t) \tag{3.6}$$

siendo: u_t los datos externos de entrada, f_t la aptitud del estado interno de la red, i_t la aptitud de la entrada interna de la red, s_t la entrada interna generada en la red, c_t el estado interno de la red, o_t la aptitud de la salida generada por la red, y h_t la salida general de la red. El operador $\otimes$

alude al producto elemento a elemento, mientas que $\sigma(\cdot)$ y $\psi(\cdot)$ representan a la función sigmoidea (valuada en $[0,1]$) y a la función tangente hiperbólica (valuada en $[-1,1]$), respectivamente.

El funcionamiento de la red LSTM se describe a continuación, pero primero cabe aclarar que tanto los datos de entrada externos, así como la salida general de la red (retardada), están presentes en la mayoría de las secciones de la red (en específico, en las secciones que evalúan las aptitudes y la entrada generada por la red). La parte principal de una red LSTM es su estado interno, o memoria (3.4), que determina cuáles datos son útiles para la red; en principio se evalúa la aptitud de lo que se ha aprendido por medio de (3.1), después se suma a esto la entrada interna generada por la red (3.3) evaluada por (3.2), la salida de la red LSTM (3.6) resulta de la memoria generada evaluada por (3.5). Las evaluaciones, que se hacen en la red, son multiplicaciones por valores entre 0 y 1, con 0 como inútil y 1 como útil.

Al igual que las redes convencionales, para las redes LSTM se han desarrollado varios algoritmos de entrenamiento, tal y como se muestra en los trabajos de Ergen y Kozat [121] y Gal y Ghahramani [122]. Estos algoritmos van desde adaptaciones del algoritmo de propagación hacia atrás, métodos que involucran al gradiente del error descendiente, y hasta métodos estocásticos. Asimismo, el tratamiento de la información descrito anteriormente es aprendido y aplicado durante el entrenamiento de la red, sin importar el método que se use, lo cual resulta muy práctico.

Este modelo de red neuronal fue elaborado para evitar el problema de "desaparición del gradiente" (mencionado en la Subsección 2.1.2), el cual está presente en las redes neuronales recurrentes. Los problemas con la transmisión del gradiente del error, a través de la red durante el entrenamiento, se ven solucionados con la incorporación de las compuertas y en especial del estado interno de la red; debido a que la información aprendida es representada por una ecuación lineal, en vez de una red con una función de activación que escala el valor del error en el entrenamiento, por lo que la información puede pasar sin dificultad (o al menos con mayor facilidad) por toda la red.

Por otra parte, el nombre "término de memoria grande-corta" viene de esta explicación: las redes neuronales recurrentes tienen dependencias grandes de memoria representadas por sus pesos sinápticos, los cuales cambian lentamente durante el entrenamiento, codificando conocimiento general relacionado con los datos que maneja; también se tienen dependencias pequeñas, en forma de activaciones efímeras, las cuales pasan de nodo en nodo en la red. Las redes LSTM, por medio

de su estado interno, almacenan la información de manera selectiva usando compuertas, lo que representa una situación intermedia y mejor a lo hecho una red recurrente común.

Una estructura de red LSTM notable en tiempo discreto, aparte de las muchas que se han propuesto en la literatura, es la descrita por las siguientes ecuaciones (definidas en el conjunto de los números reales $\mathbb{R}$):

$$F(k)=\sigma\left(W^f U(k)+V^f H(k-1)\right) \tag{3.7}$$

$$I(k)=\sigma\left(W^i U(k)+V^i H(k-1)\right) \tag{3.8}$$

$$S(k)=\psi(W^s U(k)+V^s H(k-1)) \tag{3.9}$$

$$C(k)=F(k)\otimes C(k-1)+I(k)\otimes S(k) \tag{3.10}$$

$$O(k)=\sigma(W^o U(k)+V^o H(k-1)) \tag{3.11}$$

$$H(k)=O(k)\otimes\psi(C(k)) \tag{3.12}$$

$$Y_r(k)=\psi(W^r H(k)) \tag{3.13}$$

con: $F(k), I(k), S(k), C(k), O(k), H(k)\in\mathbb{R}^n$ como secciones de la red LSTM (en orden: aptitud del estado interno, aptitud de la entrada interna, entrada interna, estado interno, aptitud de la salida y salida de la red LSTM), $U(k)\in\mathbb{R}^m$ como el vector de entradas externas, y $Y_r(k)\in\mathbb{R}^l$ como el vector de salidas estimadas formado a partir de la salida de la red LSTM. Los pesos sinápticos de la red en general: $W^f, W^i, W^s, W^o\in\mathbb{R}^{n\times m}$; $W^r\in\mathbb{R}^{l\times n}$; $V^f, V^i, V^s, V^o\in\mathbb{R}^{n\times n}$ como matrices diagonales o $V^f, V^i, V^s, V^o\in\mathbb{R}^n$ como vectores. Además, el operador $\otimes$ es la representación del producto elemento a elemento entre vectores, mientras que $\sigma(\cdot)$ y $\psi(\cdot)$ representan a las funciones sigmoidea y tangente hiperbólico, respectivamente.

Para la estructura que se acaba de mostrar, (3.7)-(3.12) son homólogas a (3.1)-(3.6), en el sentido que están organizadas de la misma forma y realizan las mismas funciones. Así, (3.13) se interpreta como las salidas estimadas, correspondientes a una determinada tarea, que se conforman por la suma ponderada de elementos en (3.12) aplicada como argumento de la función $\psi(\cdot)$. Luego, los superíndices m, n y l representan al número de entradas externas, el número de salidas de la red LSTM y el número de salidas estimadas, respectivamente. Por último, las matrices de pesos sinápticos W ponderan a la entrada externa de la red (a excepción de W^r, que pondera la salida sin retardo

de la red LSTM), y las matrices de pesos sinápticos V ponderan a la salida retardada de la red LSTM.

Estructuras de redes neuronales como la que se acaba de exponer son muy utilizadas en el control automático (no necesariamente utilizando redes LSTM); ya que al incrementar el tamaño de la red LSTM (o la red que se utilice) se mejoran las estimaciones realizadas por ésta, a partir de (3.13), porque estas estimaciones tendrán más elementos que describen mejor su comportamiento. Tomando en cuenta la red LSTM (3.7)-(3.12), en la siguiente sección se muestra una propuesta de red, la cual incluirá fundamentos de los sistemas difusos.

3.2. Red Recurrente LSTM Difusa propuesta

La estructura de la red neuronal propuesta, resultado de combinar la lógica difusa con una red LSTM, toma como base a la estructura de un ANFIS (Figura 2.6 (b)) con algunas modificaciones. La nueva red, nombrada "Red recurrente LSTM difusa" y que se exhibe en la Figura 3.2, aún conserva varios procesos característicos de un sistema difuso: fusificación de las entradas del sistema, obtención de los antecedentes de las reglas difusas, estimación de las consecuencias para cada antecedente y defusificación para obtener la respuesta del sistema difuso. El cómo se realizan estos procesos en la estructura propuesta se describe a continuación.

La siguiente descripción se realizó tomando como apoyo lo establecido para (3.7)-(3.13) definidas en el dominio de los números reales; se reutilizaron los elementos de dicha estructura ($U(k)$, $F(k)$, $I(k)$, $S(k)$, $C(k)$, $O(k)$, $H(k)$, $F(k)$ y $Y_r(k)$, así como los respectivos pesos sinápticos W y V), incluyendo a los superíndices m, n y l para indicar el número de salidas externas a la red, el número de salidas de la red LSTM empleada y el número de salidas estimadas por la red en general, respectivamente.

Iniciando con el proceso de fusificación de las entradas externas a la red, $U(k)\in\mathbb{R}^m$, éste se hizo por medio de tres funciones de pertenencia de tipo gaussiana (Figura 3.3). Las funciones gaussianas que se emplean corresponden a la siguiente ecuación:

$$\mu(k)=\exp\left(-\frac{(u_m(k)-\varsigma)^2}{2\nu}\right), \quad \nu=\delta^2>0 \tag{3.14}$$

donde: $u_m(k)\in\mathbb{R}$ es la variable que se fusifica (elemento de $U(k)$). En la función Gaussiana: $\varsigma\in\mathbb{R}$

es una constante que indica su centro y $v=\delta^2 \in \mathbb{R}$ es otra constante que indica su ancho.

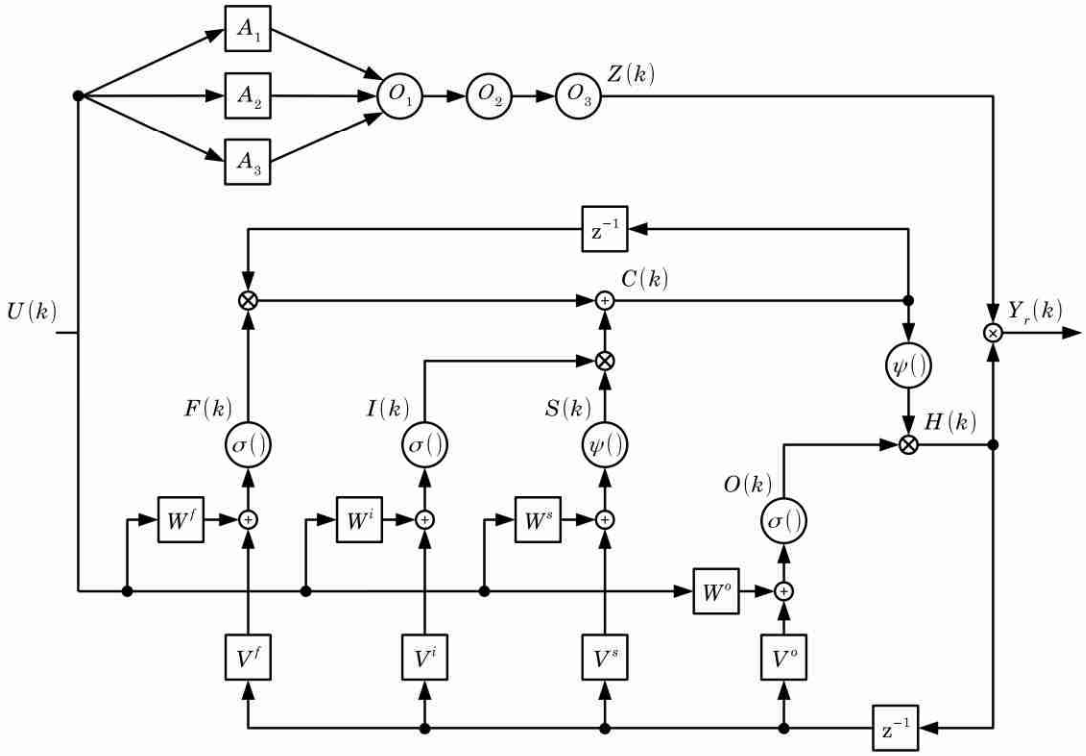


Figura 3.2. Red Recurrente LSTM Difusa.

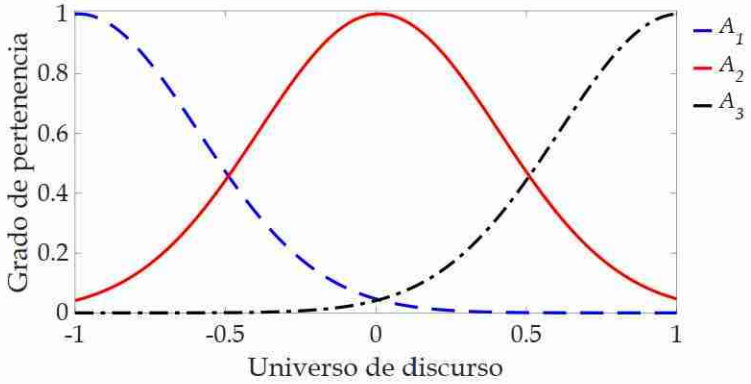


Figura 3.3. Funciones de pertenencia de la Red Recurrente LSTM Difusa: se distinguen tres conjuntos difusos, con funciones de pertenencia gaussianas, nombrados como A_1, A_2 y A_3, respectivamente.

Entonces, cada elemento del vector $U(k)$ es fusificado, usando las tres funciones de pertenencia A_1, A_2 y A_3 mostradas en la Figura 3.3, a través de (3.14). Esto por demás se puede expresar en forma vectorial: un vector que obtiene el grado de pertenencia de las entradas externas

en el conjunto difuso A_1 (usando sus respectivas funciones de pertenencia), y de igual forma otros vectores haciendo lo mismo para los conjuntos A_2 y A_3, es decir:

$$Z_1=\mu(k)_{U(k),A_1},\quad Z_2=\mu(k)_{U(k),A_2},\quad \&\quad Z_3=\mu(k)_{U(k),A_3},\quad Z_1,Z_2,Z_3\in\mathbb{R}^m \tag{3.15}$$

Ya que se va a aplicar un entrenamiento a la propuesta, se requiere entonces que todos los parámetros de la red se adecúen conforme al entrenamiento de la misma. Por lo que en el caso de las funciones de pertenencia, se debe de ajustar el valor de sus centros y anchos. Para ello, (3.15) se replantea de la siguiente forma:

$$Z_1=\exp\left((U(k)-\varsigma_1)^2\otimes\left(-\frac{1}{2}\Upsilon_1\right)\right),\quad \varsigma_1,\Upsilon_1\in\mathbb{R}^m \tag{3.16}$$

$$Z_2=\exp\left((U(k)-\varsigma_2)^2\otimes\left(-\frac{1}{2}\Upsilon_2\right)\right),\quad \varsigma_2,\Upsilon_2\in\mathbb{R}^m \tag{3.17}$$

$$Z_3=\exp\left((U(k)-\varsigma_3)^2\otimes\left(-\frac{1}{2}\Upsilon_3\right)\right),\quad \varsigma_3,\Upsilon_3\in\mathbb{R}^m \tag{3.18}$$

teniendo: ς_1, ς_2 y ς_3 como los vectores de los centros de sus respectivas funciones de pertenencia; Υ_1, Υ_2 y Υ_3 como los vectores de los anchos de sus respectivas funciones de pertenencia, en donde se tiene la misma situación en todos los casos, pero se toma a Υ_1 para mostrar que $\Upsilon_1=\left[\frac{1}{\upsilon_{1,1}},\cdots,\frac{1}{\upsilon_{1,m}}\right]'$ con $\upsilon_{1,m}=\delta_{1,m}^2$ definido en cualquier m. De aquí en adelante, los operadores $'$ y $\otimes$ se refieren a la operación transpuesta y al producto elemento a elemento en vectores y matrices, respectivamente.

Las reglas difusas que se obtienen con la red propuesta, (3.19), son similares a las que se obtienen con los sistemas difusos TSK, nótese:

$$R_p:\ \text{SI}\left(u_{q_4}(k)\ \text{ES}\ A_{q_1}\right)\ \text{Y}\left(u_{q_5}(k)\ \text{ES}\ A_{q_2}\right),\ \text{ENTONCES}\left(y_{r_{l,q_3}}(k)=h_n(k)\right) \tag{3.19}$$

donde: R_p indica el número de regla, y el total de reglas se calcula como $p=l(3^m)$; en cualquier regla, q_4 y q_5 toman valores de m pero nunca el mismo valor entre ellos ($u_{q_4}, u_{q_5}\in U$); q_1 y q_2 pueden tomar el mismo valor en cualquier regla, restringiéndose a 1, 2 y 3, ya que estos son los conjuntos difusos utilizados en la fusificación (A_1, A_2 y A_3); q_3 toma valores de 1 a p/l, va acompañado de l para el caso de múltiples salidas, $y_{r_{l,q_3}}$ es único de cada regla; se tiene $n=p$, indicando el mismo número de salidas de LSTM y de reglas, h_n es único de cada regla. Lo anterior, considerando la creación de reglas completas, consistentes y continuas (explicado en la Sección 2.1.4).

Continuando con la descripción del funcionamiento de la red propuesta, para la obtención de los antecedentes de la red se tienen que seguir una serie de pasos para modificar la organización de la información que se ha obtenido hasta el momento. Primero, se tiene que ordenar a (3.16)-(3.18) en un solo vector, como se hizo para (3.20):

$$Z_4 = \mathcal{O}_1 = T_1 Z_1 + T_2 Z_2 + T_3 Z_3 = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_2 \end{bmatrix}, \quad Z_4 \in \mathbb{R}^{3m} \tag{3.20}$$

se hace uso de matrices auxiliares T_1, T_2 y T_3, de dimensiones adecuadas, compuestas de elementos con valores de 0 y 1 que sirven solamente para acomodar los datos precedentes, por lo que sus valores no están sujetos a modificación en el entrenamiento de la red.

Los antecedentes de la red son el resultado de una multiplicación (operación que califica como norma-T, Jin y Sendhoff [123]), la cual se da entre la combinación de los elementos del vector resultante de (3.20) (ningún elemento se multiplica por si mismo):

$$Z_5 = \mathcal{O}_2 = T_4 Z_4 \otimes T_5 Z_4, \quad Z_5 \in \mathbb{R}^q, \quad q = \frac{p}{l} \tag{3.21}$$

con T_4 y T_5 nuevamente como matrices auxiliares para el acomodo de los datos.

Si (3.21) se normaliza, como se observa en (3.22), se consiguen los antecedentes, $Z(k)$, para las reglas difusas de la red propuesta en el caso de sistemas con una sola salida. En el caso de un sistema con varias salidas, este vector $Z(k)$ se obtiene con un arreglo, (3.23), en el que el resultado de (3.22) se repite para cada salida que se estima. Es decir, cada salida estimada, de un sistema de múltiples salidas, es obtenida usando los mismos antecedentes que las otras salidas.

$$Z_6 = \mathcal{O}_3 = \frac{1}{\varrho} Z_5^{'} = Z(k), \quad \varrho = \sum_{j=1}^{q} z_{5_{1,j}}, \quad Z_6 \in \mathbb{R}^q, \quad z_{5_{1,j}} \in \mathbb{R} \in Z_5 \tag{3.22}$$

$$Z(k) = T_8 Z_6 T_6 + T_9 Z_6 T_7 = \begin{bmatrix} Z_6 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & Z_6 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & Z_6 \end{bmatrix}, \quad Z(k) \in \mathbb{R}^{n \times l} \tag{3.23}$$

Las matrices T_6, T_7, T_8 y T_9, en (3.23), ayudan a organizar los datos y los elementos **0** son vectores de dimensiones iguales a Z_6 con la misma finalidad. Las consecuencias de las reglas difusas, para la red propuesta, corresponden a la salida de la red LSTM presente en la nueva estructura; se usa a la red LSTM formada con (3.7)-(3.12).

Por último, las salidas estimadas por la red son calculadas por (3.24), la cual es una ecuación que efectúa la misma operación de defusificación que se mencionó para (2.34), que se refiere a la media ponderada de cualquier sistema TSK pero en forma vectorial.

$$Y_r(k) = Z^{'}(k)H(k), \quad Y_r(k) \in \mathbb{R}^l \tag{3.24}$$

Con la nueva estructura de red difuso-neuronal definida, lo siguiente es establecer un algoritmo de entrenamiento con el que dicha red modifique sus parámetros. Además, es importante encontrar condiciones de estabilidad para la red recurrente LSTM difusa, tomando en cuenta a la estructura de la red y a su algoritmo de entrenamiento, las cuales garanticen en la medida de lo posible su buen funcionamiento. De estos temas se habla en las siguientes subsecciones.

Una vez aclarada la construcción y operación de la Red Recurrente LSTM Difusa, en la siguiente sección se compara su desempeño en diferentes tareas, relacionadas con la identificación de sistemas no lineales, con respecto al de otras redes neuronales y sistemas difusos. Esto, para comprobar si la nueva red cumple con lo establecido hasta el momento.

3.2.1. Entrenamiento de la Red Recurrente LSTM Difusa

Para el entrenamiento de la Red Recurrente LSTM Difusa, se diseñó un algoritmo de entrenamiento que funciona basándose en iteraciones. Este algoritmo toma datos generados por la red, y del sistema en el que es aplicada, en instantes de tiempo determinados por un periodo de muestreo fijo, esto es a lo que se refiere con iteraciones. Los datos son procesados, utilizando los resultados para ajustar los parámetros de la red. Estos procesos, el procesamiento de datos y el ajuste de parámetros, se realizan entre cada iteración.

De esta forma, se puede ejecutar este algoritmo entrenamiento para la red propuesta en aplicaciones que acontecen en línea, cada vez que se muestrea a la red propuesta y al sistema de interés en tiempo real. O también, este algoritmo se puede ejecutar para la red propuesta en

aplicaciones fuera de línea, haciendo uso del historial de datos perteneciente a la encomienda en que es usada la red.

El entrenamiento se basa en el ya establecido algoritmo de propagación hacia atrás, según lo mencionado en la Subsección 2.1.2; se trata de una variación conocida como "propagación hacia atrás a través del tiempo" (más conocido por las siglas BPTT que corresponden a su nombre en inglés, "back propagation through time"). Este algoritmo se usa para ajustar parámetros de redes recurrentes, generalmente fuera de línea, en donde se manipulan los datos de entrada y los generados por la red en que se aplica de tal forma que la red es modificada para tratarse como una red de conexiones hacia adelante (se dice que se "desenrolla" a la estructura de la red recurrente, porque se toman datos generados en diferentes muestreos, por ejemplo en $k-1$, k y $k+1$).

Un sistema discreto es definido en iteraciones, que son mediciones a partir de un periodo de muestreo sobre el funcionamiento de dicho sistema. La red propuesta en este trabajo está definida en tiempo discreto, de igual forma las aplicaciones en las que se puede usar la propuesta deben establecerse en tiempo discreto, hechos que se consideraron en el entretenimiento de la red.

Al tratarse de sistemas discretos, se pueden guardar los datos de las iteraciones de los sistemas en registros o historiales de datos. En el entrenamiento se emplean los datos generados por la red en su iteración actual e inmediata anterior, los elementos de la entrada externa a la red pueden usar el valor en su iteración actual o cualquier anterior dependiendo de como se hayan definido, y del sistema en el que se está trabajando solo se requiere de sus iteraciones actuales.

Para aplicar el algoritmo de entrenamiento del que se habla, se usó a (3.25), que se asemeja a (2.15) pero con algunas diferencias: (3.25) es aplicada de forma matricial o vectorial (teniendo cuidado al momento de manejar los datos involucrados en el entrenamiento), el término de velocidad de entrenamiento es sacado de la modificación del peso por lo que ésta solo dependerá del paso del gradiente del error a través de la red.

$$W(k+1)=W(k)+\eta_W \Delta W(k)+\alpha_W \Delta W(k-1), \quad \eta_W > \alpha_W \in \mathbb{R} \tag{3.25}$$

donde: $W(k+1)$ es la matriz (o vector) de pesos sinápticos modificados para la siguiente iteración, $\Delta W(k)$ es el cambio que se aplica a los pesos con las mismas dimensiones que su matriz de pesos correspondiente en la iteración actual, $\Delta W(k-1)$ es el cambio en la iteración anterior que se toma

en cuanta para dar rapidez y estabilidad al algoritmo de entrenamiento, η_W es una variable que regula el impacto de la modificación de los pesos (velocidad de aprendizaje), y α_W es una variable que regula el impacto de la modificación anterior de los pesos (amortiguamiento).

Como este algoritmo de entrenamiento es del tipo supervisado, se tiene que generar un error, de la forma de (3.26), que sirva de guía para determinar el ajuste de parámetros de la red. A partir del error se formará la función que se quiere optimizar (minimizar), en este caso correspondiente a (3.27), requerida por el algoritmo de entrenamiento.

El índice de desempeño del algoritmo en determinado número de iteraciones corresponde a (3.28), este índice mide la efectividad del entrenamiento y generalización de la red recurrente LSTM difusa; los valores de esta ecuación tienen que tener una tendencia hacia cero durante el entrenamiento, lo mismo que en la generalización, ya que esto indica un buen desempeño del algoritmo de entrenamiento (entre más cercano a cero se mantenga es mejor). Esto se ideó y realizó conforme a lo mencionado para (2.2)-(2.15) del capítulo anterior.

$$E(k)=Y_d(k)-Y_r(k), \quad Y_d(k), \quad E(k)\in\mathbb{R}^l \tag{3.26}$$

$$\xi(k)=\frac{1}{2}E^{'}(k)E(k), \quad \xi(k)\in\mathbb{R} \tag{3.27}$$

$$\Xi(k)=\frac{1}{N}\sum_{k=1}^{N}\xi(k), \quad \Xi(k)\in\mathbb{R} \tag{3.28}$$

aquí: $E(k)$ es el vector de error formado por la salida a estimar, $Y_d(k)$, y la salida estimada por la red, $Y_r(k)$, además N indica el número total de iteraciones del proceso que se está efectuando o evaluando (ya sea entrenamiento o generalización).

El error (3.26) tiene que hacerse pasar por toda la red, con el fin de dar valor a la modificación de los pesos sinápticos. Este proceso se muestra en (3.29)-(3.59) y para ello se debe de considerar el flujo de datos de la red, así como las dimensiones de sus elementos, en conformidad a lo descrito en esta subsección y la Subsección 2.1.2.

$$D_1=E(k)H^{'}(k) \tag{3.29}$$

$$D_2=T_6D_1T_8+T_7D_1T_9 \tag{3.30}$$

$$D_3=\frac{1}{\varrho}D_2 \tag{3.31}$$

$$D_4 = T_5^{'} \otimes D_3^{'} + T_4^{'} \otimes D_3^{'} \tag{3.32}$$

$$D_5 = T_1^{'} \otimes D_4^{'} \tag{3.33}$$

$$D_6 = T_2^{'} \otimes D_4^{'} \tag{3.34}$$

$$D_7 = T_3^{'} \otimes D_4^{'} \tag{3.35}$$

$$D_8 = \exp\left((U(k) - \varsigma_1)^2 \otimes \left(-\frac{1}{2} \Upsilon_1 \right) \right) \otimes D_5 \tag{3.36}$$

$$D_9 = -\frac{1}{2} \Upsilon_1 \otimes D_8 \tag{3.37}$$

$$D_{10} = (U(k) - \varsigma_1)^2 \otimes D_8 \tag{3.38}$$

$$D_{11} = 2(U(k) - \varsigma_1) \otimes D_9 \tag{3.39}$$

$$D_{12} = \exp\left((U(k) - \varsigma_2)^2 \otimes \left(-\frac{1}{2} \Upsilon_2 \right) \right) \otimes D_6 \tag{3.40}$$

$$D_{13} = -\frac{1}{2} \Upsilon_2 \otimes D_{12} \tag{3.41}$$

$$D_{14} = (U(k) - \varsigma_2)^2 \otimes D_{12} \tag{3.42}$$

$$D_{15} = 2(U(k) - \varsigma_2) \otimes D_{13} \tag{3.43}$$

$$D_{16} = \exp\left((U(k) - \varsigma_3)^2 \otimes \left(-\frac{1}{2} \Upsilon_3 \right) \right) \otimes D_7 \tag{3.44}$$

$$D_{17} = -\frac{1}{2} \Upsilon_3 \otimes D_{16} \tag{3.45}$$

$$D_{18} = (U(k) - \varsigma_3)^2 \otimes D_{16} \tag{3.46}$$

$$D_{19} = 2(U(k) - \varsigma_3) \otimes D_{17} \tag{3.47}$$

$$D_{20} = Z_3^{'} E(k) \tag{3.48}$$

$$D_{21} = D_{20} \otimes \psi(C(k)) \tag{3.49}$$

$$D_{22} = D_{20} \otimes O(k) \tag{3.50}$$

$$D_{23} = \dot{\psi}(C(k)) \otimes D_{22} \tag{3.51}$$

$$D_{24} = S(k) \otimes D_{23} \tag{3.52}$$

$$D_{25} = I(k) \otimes D_{23} \tag{3.53}$$

$$D_{26} = C(k) \otimes D_{23} \tag{3.54}$$

$$D_{27} = F(k) \otimes D_{23} \tag{3.55}$$

$$D_{28}=\dot{\psi}(W^{s}U(k)+V^{s}H(k-1))\otimes D_{25} \tag{3.56}$$

$$D_{29}=\dot{\sigma}(W^{o}U(k)+V^{o}H(k-1))\otimes D_{21} \tag{3.57}$$

$$D_{30}=\dot{\sigma}\left(W^{f}U(k)+V^{f}H(k-1)\right)\otimes D_{26} \tag{3.58}$$

$$D_{31}=\dot{\sigma}\left(W^{i}U(k)+V^{i}H(k-1)\right)\otimes D_{24} \tag{3.59}$$

Y el cambio de pesos, $\Delta W(k)$ de (3.25), utilizando el recorrido del error por toda la red, establecido en (3.29)-(3.59), se define para cada matriz y vector de pesos como:

$$\Delta\varsigma_{1}(k)=-D_{11} \tag{3.60}$$

$$\Delta\Upsilon_{1}(k)=-\frac{1}{2}D_{10} \tag{3.61}$$

$$\Delta\varsigma_{2}(k)=-D_{15} \tag{3.62}$$

$$\Delta\Upsilon_{2}(k)=-\frac{1}{2}D_{14} \tag{3.63}$$

$$\Delta\varsigma_{3}(k)=-D_{19} \tag{3.64}$$

$$\Delta\Upsilon_{3}(k)=-\frac{1}{2}D_{18} \tag{3.65}$$

$$\Delta W^{f}(k)=D_{30}U^{'}(k) \tag{3.66}$$

$$\Delta V^{f}(k)=D_{30}\otimes H(k-1) \tag{3.67}$$

$$\Delta W^{i}(k)=D_{31}U^{'}(k) \tag{3.68}$$

$$\Delta V^{i}(k)=D_{31}\otimes H(k-1) \tag{3.69}$$

$$\Delta W^{s}(k)=D_{28}U^{'}(k) \tag{3.70}$$

$$\Delta V^{s}(k)=D_{28}\otimes H(k-1) \tag{3.71}$$

$$\Delta W^{o}(k)=D_{29}U^{'}(k) \tag{3.72}$$

$$\Delta V^{o}(k)=D_{29}\otimes H(k-1) \tag{3.73}$$

Con estas ecuaciones de los cambios de pesos sinápticos producidos por el algoritmo de entrenamiento, (3.60)-(3.73), es posible trabajar y proponer condiciones de estabilidad para la red que involucren a estos elementos. De esta forma, se puede asegurar un funcionamiento adecuado de la Red Recurrente LSTM Difusa.

3.2.2. Condiciones de estabilidad para la Red Recurrente LSTM Difusa

El Teorema 3.1 da condiciones de estabilidad para el aprendizaje de la red propuesta. Para facilitar su entendimiento, se toman las dimensiones de la red como $m=1, l=1, p=3$ y $\kappa=3$, además:

$$e(k)=\hat{y}(k)-y(k) \tag{3.74}$$

$$e(k)=Z^{'}\left(\widetilde{W}\right)H\left(\widetilde{W}\right)+\mu(k) \tag{3.75}$$

$$Z^{'}\left(\widetilde{W}\right)H\left(\widetilde{W}\right)=Z^{'}(W^{*})H(W^{*})-Z^{'}(W)H(W) \tag{3.76}$$

siendo $\widetilde{W}=W(k)-W^{*}$, $W(k)$ es la representación de los pesos sinápticos de la red y W^{*} su valor óptimo. Las dinámicas no modeladas del sistema por parte de la red son representadas por $\mu(k)$.

Teorema 3.1. *Si los parámetros de aprendizaje para cada arreglo de pesos sinápticos, de la Red Recurrente LSTM Difusa, presentados en (3.25), satisfacen:*

$$\eta_{W_q}(k)=\frac{\eta}{1+\left\|vec\left(\Delta W_q(k)\right)\right\|^2} \tag{3.77}$$

$$\alpha_{W_q}(k)=\frac{\alpha}{1+\left\|vec\left(\Delta W_q(k-1)\right)\right\|^2} \tag{3.78}$$

con $1\geq\eta>0$ *y* $\eta\geq\alpha>0$ *obtenidas por métodos de prueba y error, y* W_q *representando a los arreglos de pesos sinápticos* W^f, W^i, W^s, W^o, V^f, V^i, V^s, V^o, χ_1, χ_2, χ_3, Υ_1, Υ_2, Υ_3, *entonces la norma del error de identificación:*

$$e_N(k)=\sum_{q=1}^{\rho}\left[\frac{\eta e(k)}{1+\max_k\left\|vec\left(\Delta W_q(k)\right)\right\|^2}+\frac{\alpha e(k)}{1+\max_k\left\|vec\left(\Delta W_q(k-1)\right)\right\|^2}\right] \tag{3.79}$$

con $\rho=8+2\kappa$, *cumple el siguiente rendimiento promedio:*

$$\lim_{T\to\infty}\sup\frac{1}{T}\sum_{k=1}^{T}e_N^2(k)\leq(\eta+\alpha)\bar{\mu} \tag{3.80}$$

donde $\bar{\mu}=\max_k\left[\mu^2(k)\right]$, *considerando a (3.75).*

■

Según el teorema, si los parámetros de aprendizaje de cada arreglo de pesos sinápticos cumple con (3.77) y (3.78), entonces la norma del error de identificación (diferencia entre la red y el sistema a identificar) se mantiene bajo (3.80). Como parte de la nomenclatura usada en el teorema, se definió el operador "vec" que reorganiza los elementos de una matriz en un vector. En seguida, el Teorema 3.1 es demostrado.

Demostración. Se inicia con la definición de una función candidata de Liapunov:

$$L(k)=\sum_{q=1}^{\rho} L_q(k) \tag{3.81}$$

$$L_q(k)=\mathrm{tr}\left(\tilde{W}_q^T(k)\tilde{W}_q(k)\right) \tag{3.82}$$

en donde: $\rho=14$ y $L_1(k)$, $L_2(k)$, $L_3(k)$, $L_4(k)$, $L_5(k)$, $L_6(k)$, $L_7(k)$, $L_8(k)$, $L_9(k)$, $L_{10}(k)$, $L_{11}(k)$, $L_{12}(k)$, $L_{13}(k)$ y $L_{14}(k)$ están relacionadas a los arreglos de pesos sinápticos W^f, W^i, W^s, W^o, V^f, V^i, V^s, V^o, ς_1, ς_2, ς_3, Υ_1, Υ_2 y Υ_3, respectivamente.

Se tienen que (3.81) debe cumplir con: $L(k+1)\leq L(k)$, $L(k)>0$ y $L(k)\in\mathbb{R}$. En (3.82), el operador "tr" denota la traza de una matriz que tiene las siguientes propiedades:

$$\mathrm{tr}\left(A'B\right)=\mathrm{tr}\left(AB'\right)=\mathrm{tr}\left(B'A\right)=\mathrm{tr}\left(BA'\right)=\sum_{ij}A_{ij}B_{ij},\quad A,B\in\mathbb{R}^{m\times n} \tag{3.83}$$

cada elemento de esta ecuación puede desarrollarse individualmente, de forma que cumpla con las condiciones requeridas para sí mismo, así como para todo el conjunto. Adicionalmente, el desarrollo de un elemento cualquiera será el mismo que el de cualquier otro. Para ilustrar esto, se desarrolló solo la parte correspondiente a $L_2(k)$, tomando en cuenta a (3.82):

$$L_2(k+1)\leq L_2(k) \tag{3.84}$$

$$L_2(k)=\mathrm{tr}\left\{\widetilde{W^i}'(k)\widetilde{W^i}(k)\right\} \tag{3.85}$$

$$\widetilde{W^i}(k)=\overline{W^i}-W^i(k) \tag{3.86}$$

$\overline{W^i}$ es el valor ideal de los pesos sinápticos W^i que se espera alcanzar con el entrenamiento. A partir de (3.83), y trabajando con (3.84)-(3.86), se obtiene:

$$L_2(k+1)-L_2(k)\leq 0 \quad (3.87)$$

$$\mathrm{tr}\left\{\widetilde{W^i}'(k+1)\widetilde{W^i}(k+1)\right\}-\mathrm{tr}\left\{\widetilde{W^i}'(k)\widetilde{W^i}(k)\right\}\leq 0 \quad (3.88)$$

$$\mathrm{tr}\left\{\left[\left(\overline{W^i}\right)'-\left(W^i(k+1)\right)'\right]\left[\overline{W^i}-W^i(k+1)\right]\right\}-\mathrm{tr}\left\{\left[\left(\overline{W^i}\right)'-\left(W^i(k)\right)'\right]\left[\overline{W^i}-W^i(k)\right]\right\}\leq 0 \quad (3.89)$$

y continuando:

$$\mathrm{tr}\left\{-\left(\overline{W^i}\right)'W^i(k+1)-\left(W^i(k+1)\right)'\overline{W^i}+\left(W^i(k+1)\right)'W^i(k+1)+\left(\overline{W^i}\right)'W^i(k)\cdots\right.$$
$$\left.\cdots+\left(W^i(k)\right)'\overline{W^i}-\left(W^i(k)\right)'W^i(k)\right\}\leq 0 \quad (3.90)$$

si $W^i(k+1)$ es sustituido por $W^i(k+1)=W^i(k)+\eta_{W^i}\Delta W^i(k)+\alpha_{W^i}\Delta W^i(k-1)$:

$$\mathrm{tr}\left\{-\eta_{W^i}\left(\overline{W^i}\right)'\Delta W^i(k)-\alpha_{W^i}\left(\overline{W^i}\right)'\Delta W^i(k-1)-\eta_{W^i}\left(\Delta W^i(k)\right)'\left(\overline{W^i}\right)-\cdots\right.$$
$$\cdots-\alpha_{W^i}\left(\Delta W^i(k-1)\right)'\left(\overline{W^i}\right)+\eta_{W^i}\left(W^i(k)\right)'\Delta W^i(k)+\alpha_{W^i}\left(W^i(k)\right)'\Delta W^i(k-1)+\cdots$$
$$\cdots+\eta_{W^i}\left(\Delta W^i(k)\right)'W^i(k)+\eta_{W^i}^2\left(\Delta W^i(k)\right)'\Delta W^i(k)+\eta_{W^i}\alpha_{W^i}\left(\Delta W^i(k)\right)'\Delta W^i(k-1)+\cdots$$
$$\cdots+\alpha_{W^i}\left(\Delta W^i(k-1)\right)'W^i(k)+\eta_{W^i}\alpha_{W^i}\left(\Delta W^i(k-1)\right)'\Delta W^i(k)+\cdots$$
$$\left.\cdots+\alpha_{W^i}^2\left(\Delta W^i(k-1)\right)'\Delta W^i(k-1)\right\}\leq 0 \quad (3.91)$$

$$-2\eta_{W^i}\mathrm{tr}\left\{\left(\overline{W^i}\right)'\Delta W^i(k)\right\}-2\alpha_{W^i}\mathrm{tr}\left\{\left(\overline{W^i}\right)'\Delta W^i(k-1)\right\}+\cdots$$
$$\cdots+2\eta_{W^i}\mathrm{tr}\left\{\left(W^i(k)\right)'\Delta W^i(k)\right\}+2\alpha_{W^i}\mathrm{tr}\left\{\left(W^i(k)\right)'\Delta W^i(k-1)\right\}+\cdots$$
$$\cdots+2\eta_{W^i}\alpha_{W^i}\mathrm{tr}\left\{\left(\Delta W^i(k)\right)'\Delta W^i(k-1)\right\}+\eta_{W^i}^2\mathrm{tr}\left\{\left(\Delta W^i(k)\right)'\Delta W^i(k)\right\}+\cdots$$
$$\cdots+\alpha_{W^i}^2\mathrm{tr}\left\{\left(\Delta W^i(k-1)\right)'\Delta W^i(k-1)\right\}\leq 0 \quad (3.92)$$

$$-2\eta_{W^i}\mathrm{vec}\left(\overline{W^i}\right)'\mathrm{vec}\left(\Delta W^i(k)\right)-2\alpha_{W^i}\mathrm{vec}\left(\overline{W^i}\right)'\mathrm{vec}\left(\Delta W^i(k-1)\right)+\cdots$$
$$\cdots+2\eta_{W^i}\mathrm{vec}\left(W^i(k)\right)'\mathrm{vec}\left(\Delta W^i(k)\right)+2\alpha_{W^i}\mathrm{vec}\left(W^i(k)\right)'\mathrm{vec}\left(\Delta W^i(k-1)\right)+\cdots$$
$$\cdots+2\eta_{W^i}\alpha_{W^i}\mathrm{vec}\left(\Delta W^i(k)\right)'\mathrm{vec}\left(\Delta W^i(k-1)\right)+\eta_{W^i}^2\mathrm{vec}\left(\Delta W^i(k)\right)'\mathrm{vec}\left(\Delta W^i(k)\right)+\cdots$$
$$\cdots+\alpha_{W^i}^2\mathrm{vec}\left(\Delta W^i(k-1)\right)'\mathrm{vec}\left(\Delta W^i(k-1)\right)\leq 0 \quad (3.93)$$

usando $X^{'}X+Y^{'}Y\geq 2X^{'}Y$ y $X^{'}X+Y^{'}Y\geq -2X^{'}Y$, con $\forall X,Y\in\mathbb{R}^{n}$:

$$\begin{aligned}
&-\eta_{W^i}\mathrm{vec}\left(\overline{W^i}\right)^{'}\mathrm{vec}\left(\overline{W^i}\right)-\alpha_{W^i}\mathrm{vec}\left(\overline{W^i}\right)^{'}\mathrm{vec}\left(\overline{W^i}\right)+\cdots\\
&\cdots+\eta_{W^i}\mathrm{vec}\left(W^i(k)\right)^{'}\mathrm{vec}\left(W^i(k)\right)+\alpha_{W^i}\mathrm{vec}\left(W^i(k)\right)^{'}\mathrm{vec}\left(W^i(k)\right)+\cdots\\
&\cdots+\eta_{W^i}\alpha_{W^i}\left[\mathrm{vec}\left(\Delta W^i(k)\right)^{'}\mathrm{vec}\left(\Delta W^i(k)\right)+\mathrm{vec}\left(\Delta W^i(k-1)\right)^{'}\mathrm{vec}\left(\Delta W^i(k-1)\right)\right]+\cdots\\
&\cdots+\eta_{W^i}^2\mathrm{vec}\left(\Delta W^i(k)\right)^{'}\mathrm{vec}\left(\Delta W^i(k)\right)+\alpha_{W^i}^2\mathrm{vec}\left(\Delta W^i(k-1)\right)^{'}\mathrm{vec}\left(\Delta W^i(k-1)\right)\leq 0
\end{aligned}\tag{3.94}$$

$$\begin{aligned}
&-\eta_{W^i}\left\|\mathrm{vec}\left(\overline{W^i}\right)\right\|^2-\alpha_{W^i}\left\|\mathrm{vec}\left(\overline{W^i}\right)\right\|^2+\eta_{W^i}\left\|\mathrm{vec}\left(W^i(k)\right)\right\|^2+\alpha_{W^i}\left\|\mathrm{vec}\left(W^i(k)\right)\right\|^2+\cdots\\
&\cdots+\eta_{W^i}\alpha_{W^i}\left\|\mathrm{vec}\left(\Delta W^i(k)\right)\right\|^2+\eta_{W^i}\alpha_{W^i}\left\|\mathrm{vec}\left(\Delta W^i(k-1)\right)\right\|^2+\eta_{W^i}^2\left\|\mathrm{vec}\left(\Delta W^i(k)\right)\right\|^2+\cdots\\
&\cdots+\alpha_{W^i}^2\left\|\mathrm{vec}\left(\Delta W^i(k-1)\right)\right\|^2\leq 0
\end{aligned}\tag{3.95}$$

reduciendo a:

$$-(\eta_{W^i}+\alpha_{W^i})\gamma\leq 0\tag{3.96}$$

con:

$$\gamma=\left\|\mathrm{vec}\left(\overline{W^i}\right)\right\|^2-\left\|\mathrm{vec}\left(W^i(k)\right)\right\|^2-\eta_{W^i}\left\|\mathrm{vec}\left(\Delta W^i(k)\right)\right\|^2-\alpha_{W^i}\left\|\mathrm{vec}\left(\Delta W^i(k-1)\right)\right\|^2\tag{3.97}$$

entonces:

$$\left\|\mathrm{vec}\left(\overline{W^i}\right)\right\|^2-\left\|\mathrm{vec}\left(W^i(k)\right)\right\|^2-\eta_{W^i}\left\|\mathrm{vec}\left(\Delta W^i(k)\right)\right\|^2-\alpha_{W^i}\left\|\mathrm{vec}\left(\Delta W^i(k-1)\right)\right\|^2\geq 0\tag{3.98}$$

implicando:

$$\left\|\mathrm{vec}\left(\overline{W^i}\right)\right\|^2\geq\left\|\mathrm{vec}\left(W^i(k)\right)\right\|^2+\eta_{W^i}\left\|\mathrm{vec}\left(\Delta W^i(k)\right)\right\|^2+\alpha_{W^i}\left\|\mathrm{vec}\left(\Delta W^i(k-1)\right)\right\|^2\tag{3.99}$$

y en adición:

$$\eta_{W^i}(k)=\frac{\eta}{1+\|\text{vec}(\Delta W^i(k))\|^2},\quad 0<\eta<1 \tag{3.100}$$

$$\alpha_{W^i}(k)=\frac{\alpha}{1+\|\text{vec}(\Delta W^i(k-1))\|^2},\quad 0<\alpha<1,\quad \eta>\alpha \tag{3.101}$$

por lo que $\Delta L_2(k)$ puede definirse de la siguiente manera:

$$\Delta L_2(k)\leq -\pi_{W^i}e^2(k)+\lambda_{W^i}\mu^2(k) \tag{3.102}$$

donde:

$$\pi_{W^i}=\frac{\eta}{1+\|\text{vec}(\Delta W^i(k))\|^2}+\frac{\alpha}{1+\|\text{vec}(\Delta W^i(k-1))\|^2} \tag{3.103}$$

$$\lambda_{W^i}=\eta+\alpha \tag{3.104}$$

si se establece:

$$n\min\left[\left(\tilde{W}^i\right)^2\right]\leq L_2(k)\leq n\max\left[\left(\tilde{W}^i\right)^2\right] \tag{3.105}$$

siendo $n\min\left[\left(\tilde{W}^i\right)^2\right]$ y $n\max\left[\left(\tilde{W}^i\right)^2\right]$ funciones $\mathcal{K}_\infty$ al igual que (3.103) y (3.104). Luego, si $L_2(k)$ admite una función de Liapunov entrada-estado estable, la dinámica del error de identificación es entrada-estado estable, porque $\Delta L_2(k)$ esta en función de $e(k)$ y $\mu(k)$. La entrada corresponde al segundo término de (3.102) y el estado corresponde al primer término. Ya que la entrada está acotada, y la dinámica es entrada-estado estable, el estado está acotado.

Observación: Las funciones $\mathcal{K}_\infty$ son usadas en el análisis de estabilidad basado en la teoría de estabilidad desarrollada por Liapunov. Se tratan de funciones crecientes, de tal forma que mientras su argumento tiende a infinito el valor de la función también tiende a infinito. Para más detalles consultar el libro de Khalil [124].

Continuando, (3.102) se puede reescribir como:

$$\Delta L_2(k)\leq \frac{\eta e^2(k)}{1+\max\limits_k\|\text{vec}(\Delta W^i(k))\|^2}+\frac{\alpha e^2(k)}{1+\max\limits_k\|\text{vec}(\Delta W^i(k-1))\|^2}+(\eta+\alpha)\bar{\mu} \tag{3.106}$$

y sumando esta nueva ecuación de 1 a T, usando el término $L_2(T)>0$ y el término $L_2(1)$ como

una constante, se obtiene:

$$L_2(T)-L_2(1)\leq\sum_{k=1}^{T}\|e_N(k)\|^2+T(\eta+\alpha)\bar{\mu} \tag{3.107}$$

así

$$\sum_{k=1}^{T}\|e_N(k)\|^2\leq L_2(1)-L_2(T)+T(\eta+\alpha)\bar{\mu} \tag{3.108}$$

$$\sum_{k=1}^{T}\|e_N(k)\|^2\leq L_2(1)+T(\eta+\alpha)\bar{\mu} \tag{3.109}$$

por tanto, (3.80) queda establecida para W^i.

□

Se comprobó que (3.77) y (3.78) son condiciones necesarias de estabilidad, en el aprendizaje, que deben de aplicarse en cada arreglo de pesos sinápticos del algoritmo propuesto, ya que se basan en los parámetros únicos de cada arreglo. Asimismo, otras condicionales que se pueden tener en cuenta, para la estabilidad del algoritmo, son el valor inicial de sus pesos sinápticos (cercanos a 0), así como de los valores de las funciones de membresía que deben estar distribuidas equitativamente en el universo de discurso.

El aprendizaje se ejecuta en una “ventana” de tiempo, que tiene como limites los valores de este proceso en una iteración anterior y la actual. Se hizo de esta manera porque es más efectivo, tal y como documentaron Hénaff et al. [125], Chen et al. [126], Lillicrap y Santoro [127] y Sari et al. [128], ya que si en la ventana se consideran más iteraciones se genera una mayor carga de trabajo en la computadora en que se está ejecutando el algoritmo, lo que se puede convertir en un problema si la computadora no está enfocada exclusivamente en esta tarea.

3.3. Comparación entre la Red Recurrente LSTM Difusa y otros algoritmos inteligentes

Para elaborar un juicio sobre el funcionamiento de la Red Recurrente LSTM Difusa es necesario compararla contra otros sistemas inteligentes. Estos algoritmos deben haber sido

diseñados con una idea similar a la planteada en este trabajo, para mostrar las bondades que ofrece la nueva red. Para dicha comparación, se tienen dos ejercicios que implican la identificación o generación de modelos implementando estructuras cuyo aprendizaje se basa en el gradiente descendiente, con lo cual se podrá mostrar el desempeño de los algoritmos en un marco de operación similar. Los ejercicios mencionados se exhiben en las Subsecciones 3.3.1 y 3.3.2, respectivamente, mientras que los resultados obtenidos de estos son discutidos en la Subsección 3.3.3, por comodidad.

Se escogió la identificación en tiempo real, o también conocida como identificación en linea, porque es uno de los principales problemas que se trabajan dentro de la teoría de control automático; se requieren algoritmos capaces de identificar sistemas en un periodo de tiempo corto, con determinada adaptabilidad y robustez, generando un error relativamente pequeño entre el sistema real y su estimación (error de identificación). Esto es cumplido por los sistemas difusos y redes neuronales, donde el error de identificación se relaciona directamente con el índice de desempeño (3.28) que evalúa este tipo de estructuras , entre más pequeño sea el índice de desempeño menor será el error de identificación.

Los algoritmos inteligentes usados en los ejercicios, además del propuesto, son: la red neuronal propuesta por Baruch et al. [129], conocida como KFRNN (siglas en inglés de “Kalman filter recurrent neural network”, red neuronal recurrente del filtro de Kalman); la red LSTM descrita por (3.7)-(3.13); un ANFIS de orden 0 (simple ponderación de la parte de los antecedentes), con la definición de la parte de los antecedentes similar a la del algoritmo propuesto; un ANFIS de orden 1 (con una estructura semejante a (2.30)-(2.34)); una red difusa que está definida de la misma forma que la propuesta de este trabajo, excepto que, en vez de una red LSTM se hace uso de la KFRNN; y finalmente la red mostrada en [99], a la cual se le denominó en este trabajo como WN difusa, siendo WN las siglas en inglés de “wavelet network”.

Además, los ejercicios fueron programados y probados en MATLAB, que es un software común para este tipo de tareas, ambos ejercicios fueron manejados como aplicaciones en tiempo discreto ejecutándose en línea. En cada ejercicio se hace mención de k iteraciones, pero aun así se tomó en cuenta que el valor del tiempo en que son muestreadas las señales discretas; este tiempo se obtiene con la operación kT, donde T es el periodo de muestreo (dicho muestreo es uniforme y continuo). Detalles de lo anterior se dan en los ejercicios.

3.3.1. Obtención del modelo de un sistema no lineal SISO

Como primer ejercicio, se propuso la generación de un modelo para la ecuación diferencial de Mackey-Glass con retardo en el tiempo, empleando datos que se obtienen a partir de su propio muestreo. Esta serie de tiempo es caótica, sin un período claramente definido, no converge ni diverge y la trayectoria es altamente sensible a las condiciones iniciales. Tal sistema puede ser visto como un sistema SISO (siglas en inglés de "single-input single-output", una entrada una salida) que es descrito por:

$$\dot{x}(t)=\frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)}-0.1x(t) \tag{3.110}$$

en donde: $x(0)=1.2$, $\tau=17$, y $x(t)=0$ para $t<0$.

El ejercicio consistió en muestrear la solución de (3.110) durante 1,200s con un periodo de muestreo $T=1$s, obteniendo 1,201 datos correspondientes a $x(k)$ de los cuales los primeros 601 son utilizados para entrenar a los algoritmos y los restantes son utilizados para comprobar dicho aprendizaje. Así, la estimación se define por (3.112), tomando como entrada a (3.111), con condiciones iniciales menores a 0.1 para los respectivos parámetros (pesos) de los algoritmos. Los resultados de esto se muestran en la Figura 3.4.

$$U_r(k)=\begin{bmatrix} x(k-3) \\ x(k-20) \end{bmatrix} \tag{3.111}$$

$$Y_r(k)=x(k) \tag{3.112}$$

Se tiene que (3.111) es un vector conformado por dos elementos distantes entre si, se definió de esta manera porque en muchos casos parecidos, al hacer uso de datos distantes de la señal que se desea estimar se asegura la creación de modelos confiables de la señal con la que se trabaja. Cabe mencionar que entre más distantes los datos de entrada más robustos serán los resultados obtenidos.

Se efectuaron pruebas con varios datos distantes, pero solo se muestra el caso más extremo que se observó; con datos consecutivos no se notaba mucha diferencia entre los resultados obtenidos, pero con datos separados por un determinado periodo de tiempo relativamente grande, se notó un gran cambio en las estimaciones generadas (definidas como en (3.112)).

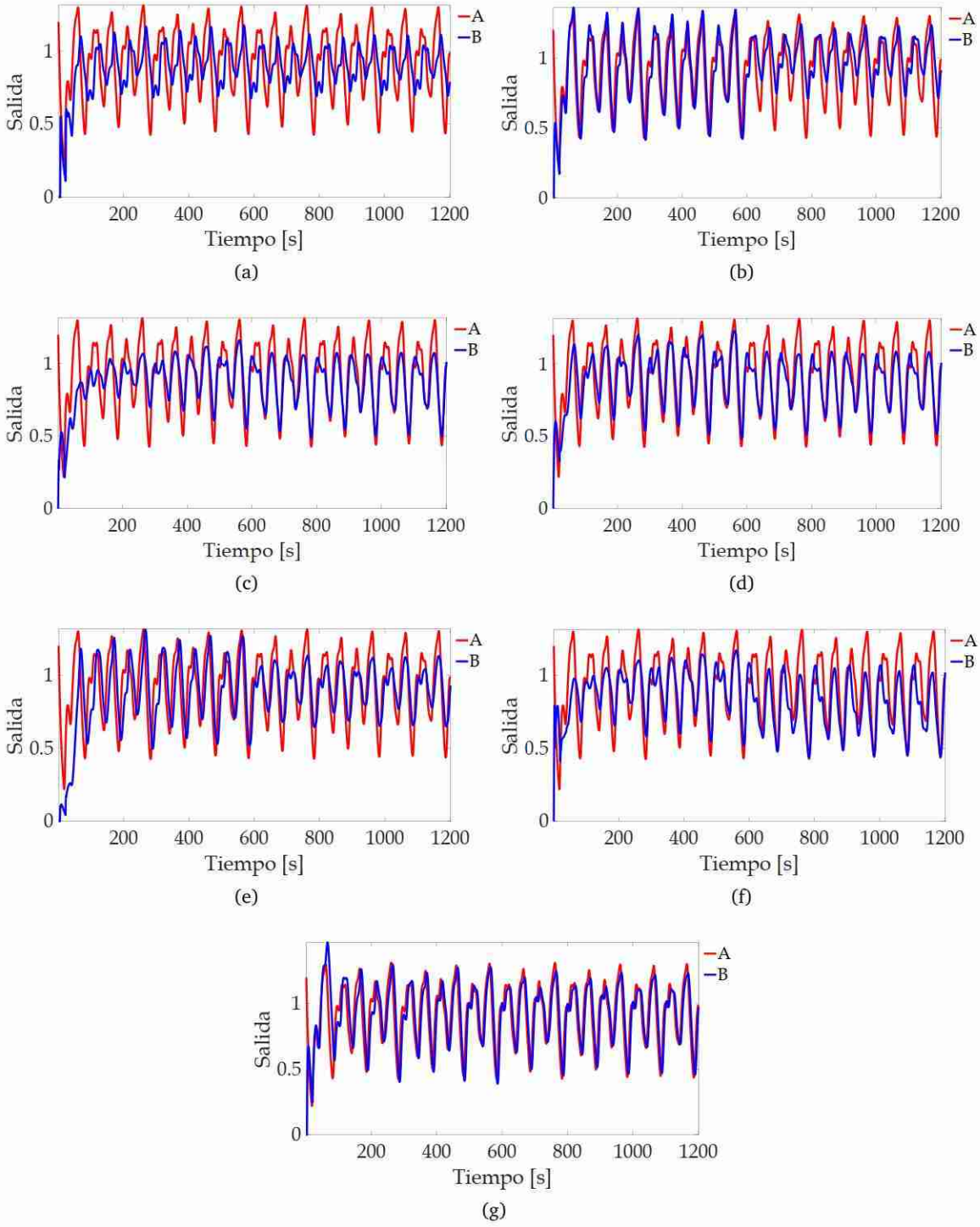


Figura 3.4. Resultados del ejercicio de la Subsección 3.3.1: los resultados con cada algoritmo se muestran en los diferentes incisos, siendo (a) KFRNN, (b) red LSTM, (c) ANFIS de orden 0, (d) ANFIS de orden 1, (e) KFRNN difusa, (f) WN difusa, y (g) Red Recurrente LSTM Difusa. En las subfiguras: la "Salida" representa a $x(k)$, "A" a la salida del sistema y "B" a la salida estimada por el algoritmo correspondiente.

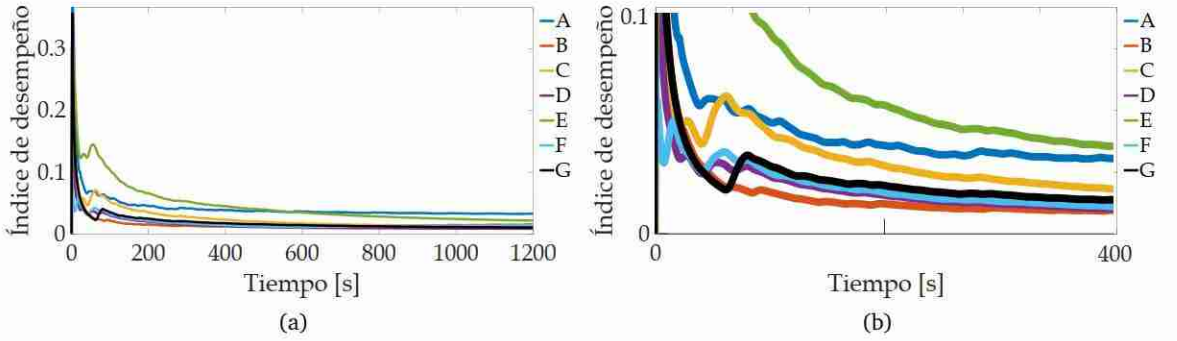


Figura 3.5. Índices de desempeño del ejercicio de la Subsección 3.3.1: (a) entrenamiento y comprobación, (b) ampliación de la gráfica del entrenamiento (primeros 400 s). En las subfiguras: "A" representa a la KFRNN, "B" a la red LSTM, "C" a el ANFIS de orden 0, "D" a el ANFIS de orden 1, "E" a la KFRNN difusa, "F" a la WN difusa, y "G" a la Red Recurrente LSTM Difusa.

Aparte de los seguimientos de trayectoria mostrados en la Figura 3.4, en la Figura 3.5 se muestra la evolución de los índices de desempeño de cada algoritmo; se exhiben el índice obtenido durante el entrenamiento y la comprobación posterior (generalización) en la misma gráfica. Por otra parte, en la Tabla 3.1 se muestra una comparación entre estos índices, pero en la iteración final del entrenamiento (iteración 601) y la comprobación (iteración 1,201). Esto, para observar que tan rápido convergen los índices a un valor cercano a cero y si se pueden mantener (indicio de que un algoritmo aprendió correctamente).

Tabla 3.1. Comparación entre los valores de los índices de desempeño de los algoritmos en el ejercicio de la Subsección 3.3.1.

Sistema	Entrenamiento	Generalización
	Iteración 601	Iteración 1,201
KFRNN	3.62×10^{-2}	3.26×10^{-2}
Red LSTM	1.09×10^{-2}	1.53×10^{-2}
ANFIS de orden 0	1.70×10^{-2}	1.16×10^{-2}
ANFIS de orden 1	0.98×10^{-2}	0.82×10^{-2}
KFRNN difusa	3.47×10^{-2}	2.15×10^{-2}
WN difusa	1.07×10^{-2}	1.35×10^{-2}
Red Recurrente LSTM Difusa	1.41×10^{-2}	1.03×10^{-2}

3.3.2. Obtención del modelo de un sistema no lineal MIMO

El segundo ejercicio consistió en la identificación de un sistema MIMO no lineal, de dos entradas y dos salidas, tomado del trabajo de Sastry et al. [130] y se conforma de la siguiente manera:

$$y_1(k+1)=\frac{0.5y_1(k)}{1+y_2^2(k)+u_1(k)} \tag{3.113}$$

$$y_2(k+1)=\frac{0.5y_1(k)y_2(k)}{1+y_2^2(k)+u_2(k)} \tag{3.114}$$

con: $u_1(k)$, $u_2(k)\in\mathbb{R}$ como las entradas del sistema, y $y_1(k)$, $y_2(k)\in\mathbb{R}$ como las salidas del sistema.

Se generaron dos conjuntos de datos para el aprendizaje y generalización de los algoritmos inteligentes, esto, al cambiar las entradas que alimentan al sistema MIMO en cada proceso. Las ecuaciones de las entradas para el aprendizaje y para la generalización son las siguientes:

$$u_1(k)=24.7\text{sen}\left(\frac{2\pi(kT)}{10}\right)+0.5\cos(2\pi(kT)) \tag{3.115}$$

$$u_2(k)=24.5\text{sen}\left(\frac{\pi(kT)}{10}\right)+0.5\text{sen}(\pi(kT)) \tag{3.116}$$

$$u_1(k)=\begin{cases}\text{Si } 0<kT\leq 50, & u_1(k)=3.3\text{sen}\left(\frac{2\pi(kT)}{10}\right)+0.1\cos(2\pi(kT))\\ \text{Si } 50+n_{u_1}<kT\leq 55+n_{u_1}, & u_1(k)=3.5\\ \text{Si } 55+n_{u_1}<kT\leq 60+n_{u_1}, & u_1(k)=-3.5\\ \text{Si } 100<kT, & u_1(k)=3.6\cos\left(\frac{2\pi(kT)}{10}\right)\end{cases} \tag{3.117}$$

$$u_2(k)=\begin{cases}\text{Si } 0<kT\leq 50, & u_2(k)=3.5\text{sen}\left(\frac{\pi(kT)}{10}\right)+0.3\text{sen}(\pi(kT))\\ \text{Si } 50+n_{u_2}<kT\leq 55+n_{u_2}, & u_2(k)=-3.5\\ \text{Si } 55+n_{u_2}<kT\leq 60+n_{u_2}, & u_2(k)=3.5\\ \text{Si } 100<kT, & u_2(k)=3.6\cos\left(\frac{2\pi(kT)}{10}\right)\end{cases} \tag{3.118}$$

con $n_{u_1}=n_{u_2}=10$, 20, 30 & 40.

El vector de entradas y el vector de salidas a estimar, de los algoritmos a comparar, se definieron acorde a las siguientes ecuaciones, respectivamente:

$$U_r(k)=\begin{bmatrix}u_{r_1}(k)\\ u_{r_2}(k)\end{bmatrix}=\begin{bmatrix}u_1(k)\\ u_2(k)\end{bmatrix}+\Theta_1,\quad U_r(k),\ \Theta_1(k)\in\mathbb{R}^2 \tag{3.119}$$

$$Y_d(k)=\begin{bmatrix}y_{d_1}(k)\\ y_{d_2}(k)\end{bmatrix}=\begin{bmatrix}y_1(k)\\ y_2(k)\end{bmatrix}+\Theta_2,\quad Y_d(k),\ \Theta_2(k)\in\mathbb{R}^2 \tag{3.120}$$

donde $\Theta_1(k)$ y $\Theta_2(k)$ representan perturbaciones en los procesos, sus dimensiones se definen acorde

a la de los vectores de entrada y salida. El valor de cada elemento en estos vectores se tomó de la generación de números aleatorios uniformemente distribuidos en el intervalo de $[-0.5,0.5]$ y $[-0.2,0.2]$ para el aprendizaje y la generalización, respectivamente. Asimismo, se establecieron condiciones iniciales en el rango $[-0.1,0.1]$ para los pesos sinápticos de los algoritmos.

Adicionar perturbaciones al sistema, (3.119) y (3.120), fue para revisar la tolerancia al ruido que tienen los algoritmos. La adaptabilidad se comprobó con el cambio de entrada, (3.115) y (3.116) por (3.117) y (3.118), en (3.113) y (3.114) del sistema. En cuanto a robustez y adaptabilidad, siempre es esperada una respuesta satisfactoria por parte de estos algoritmos, es decir, un valor bajo (cercano a cero) de sus respectivos los índices de desempeño.

Las trayectorias generadas por las salidas sistema y las salidas de los modelos, así como la comparación entre ellas, se muestran así: en la Figura 3.6 están resultados de la KFRNN, en la Figura 3.7 los de la red LSTM, en la Figura 3.8 los del ANFIS de orden 0, en la Figura 3.9 los del ANFIS de orden 1, en la Figura 3.10 los de la KFRNN difusa, en la Figura 3.11 los de la WN difusa, y en la Figura 3.12 los de la Red Recurrente LSTM Difusa.

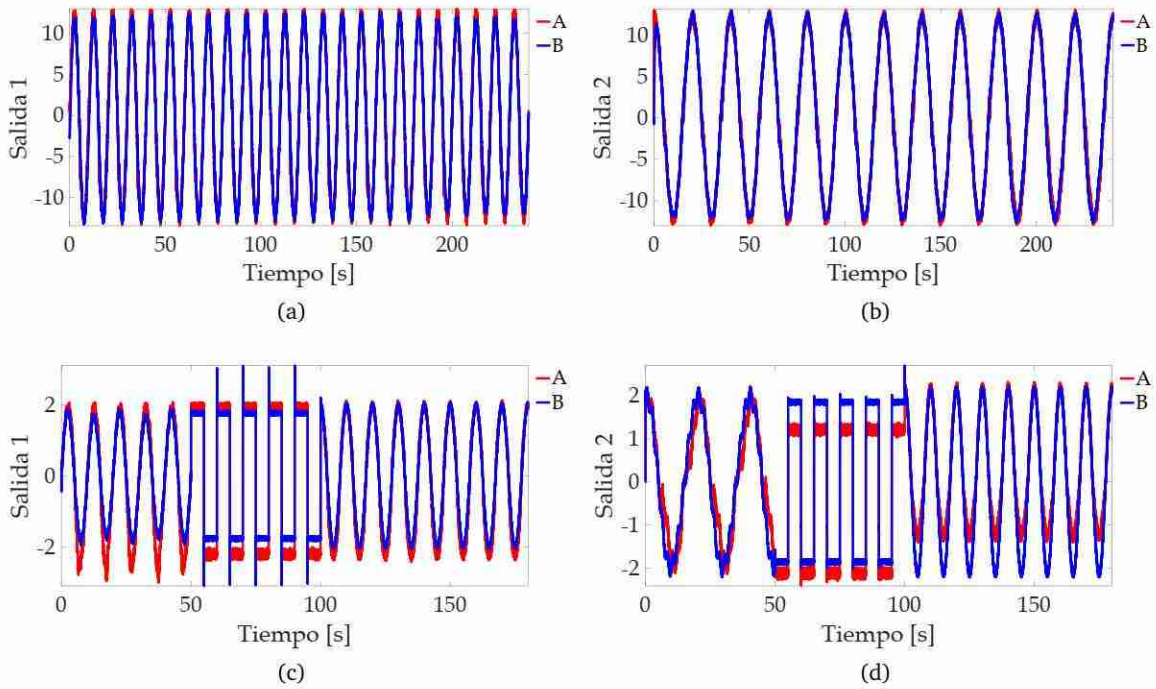


Figura 3.6. Resultados del ejercicio de la Subsección 3.3.2 con la KFRNN: (a) y (b) corresponden al entrenamiento, mientras que (c) y (d) corresponden a la generalización. En las subfiguras: la "Salida 1" representa a $y_{d_1}(k)$, la "Salida 2" a $y_{d_2}(k)$, "A" a la salida del sistema y "B" a la salida estimada por la red.

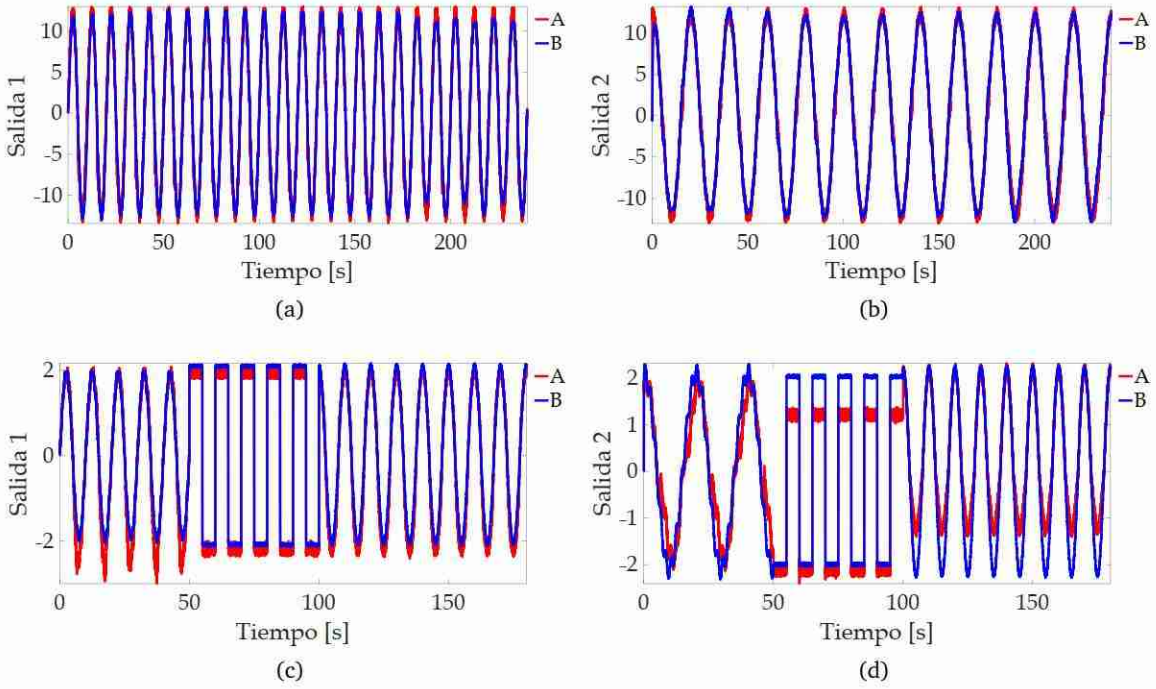


Figura 3.7. Resultados del ejercicio de la Subsección 3.3.2 con la red LSTM: (a) y (b) corresponden al entrenamiento, mientras que (c) y (d) corresponden a la generalización. En las subfiguras: la "Salida 1" representa a $y_{d_1}(k)$, la "Salida 2" a $y_{d_2}(k)$, "A" a la salida del sistema y "B" a la salida estimada por la red.

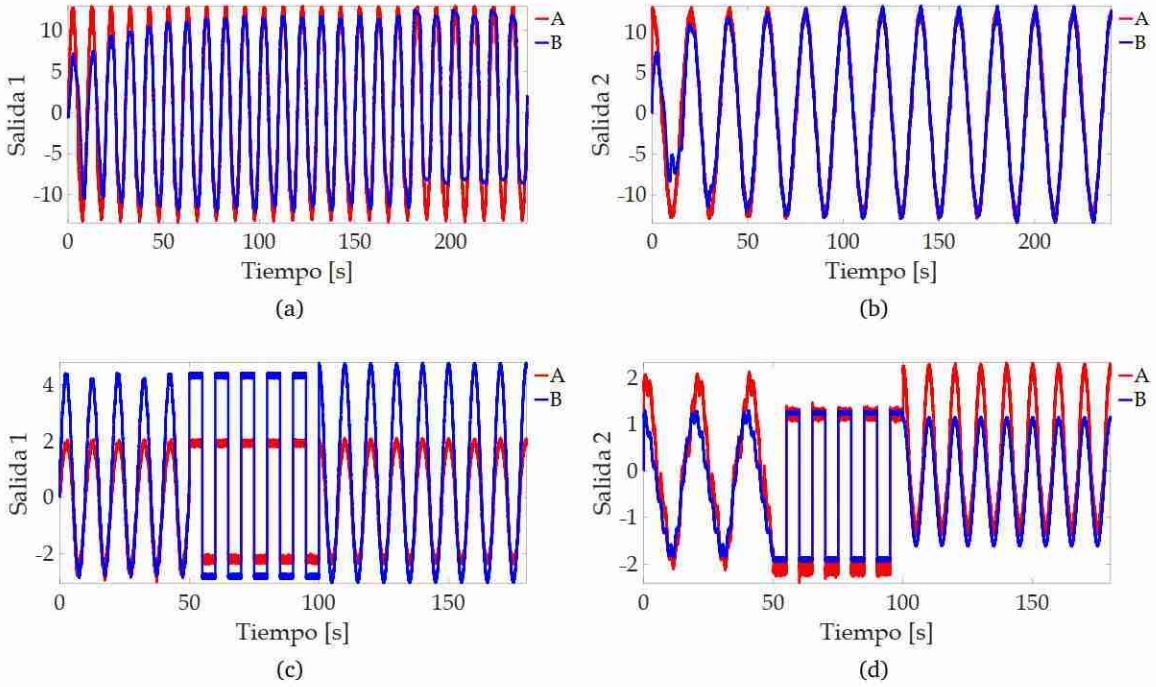


Figura 3.8. Resultados del ejercicio de la Subsección 3.3.2 con el ANFIS de orden 0: (a) y (b) corresponden al entrenamiento, mientras que (c) y (d) corresponden a la generalización. En las subfiguras: la "Salida 1" representa a $y_{d_1}(k)$, la "Salida 2" a $y_{d_2}(k)$, "A" a la salida del sistema y "B" a la salida estimada por el ANFIS.

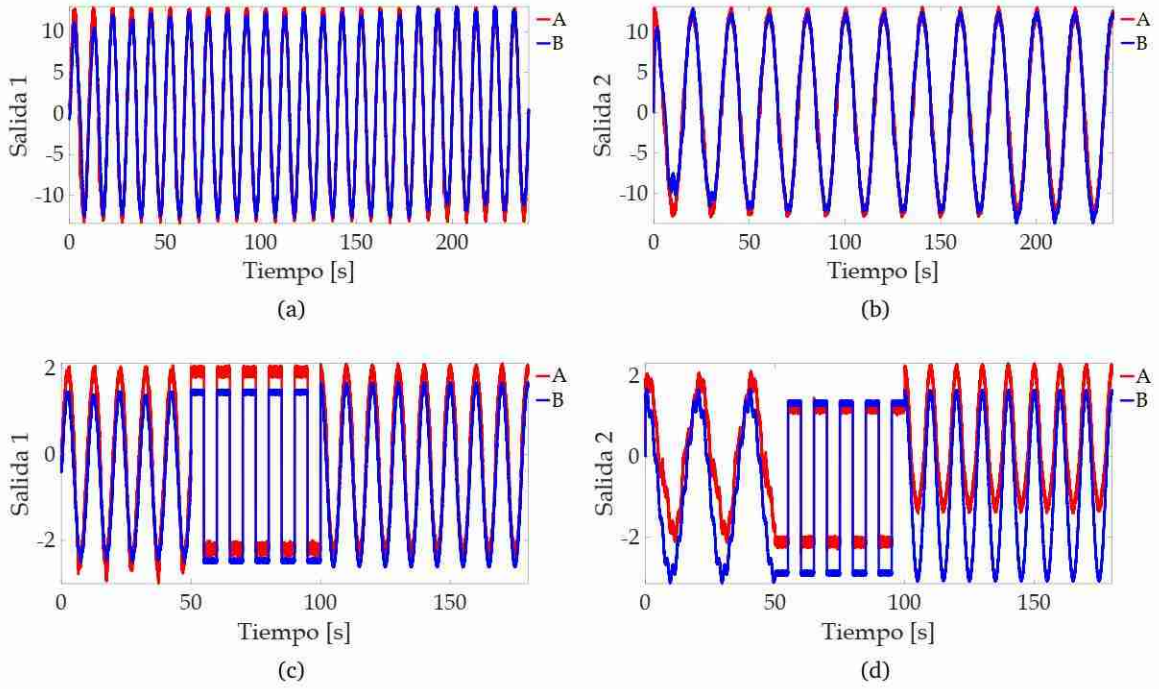


Figura 3.9. Resultados del ejercicio de la Subsección 3.3.2 con el ANFIS de orden 1: (a) y (b) corresponden al entrenamiento, mientras que (c) y (d) corresponden a la generalización. En las subfiguras: la "Salida 1" representa a $y_{d_1}(k)$, la "Salida 2" a $y_{d_2}(k)$, "A" a la salida del sistema y "B" a la salida estimada por el ANFIS.

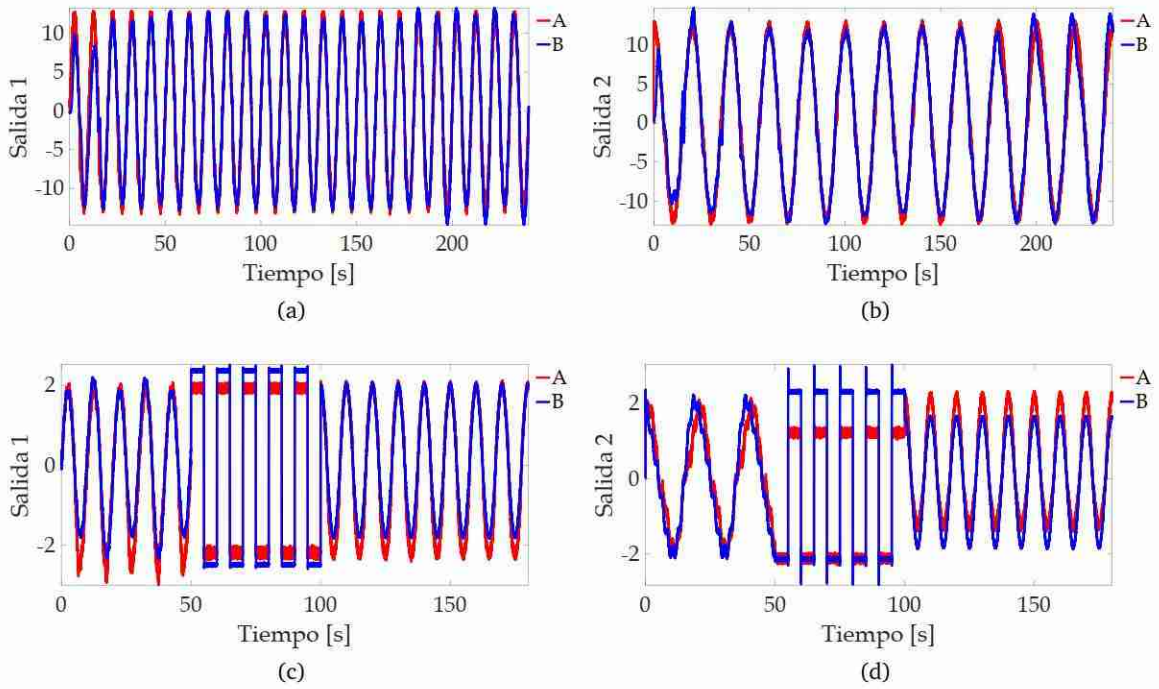


Figura 3.10. Resultados del ejercicio de la Subsección 3.3.2 con la KFRNN difusa: (a) y (b) corresponden al entrenamiento, mientras que (c) y (d) corresponden a la generalización. En las subfiguras: la "Salida 1" representa a $y_{d_1}(k)$, la "Salida 2" a $y_{d_2}(k)$, "A" a la salida del sistema y "B" a la salida estimada por la red.

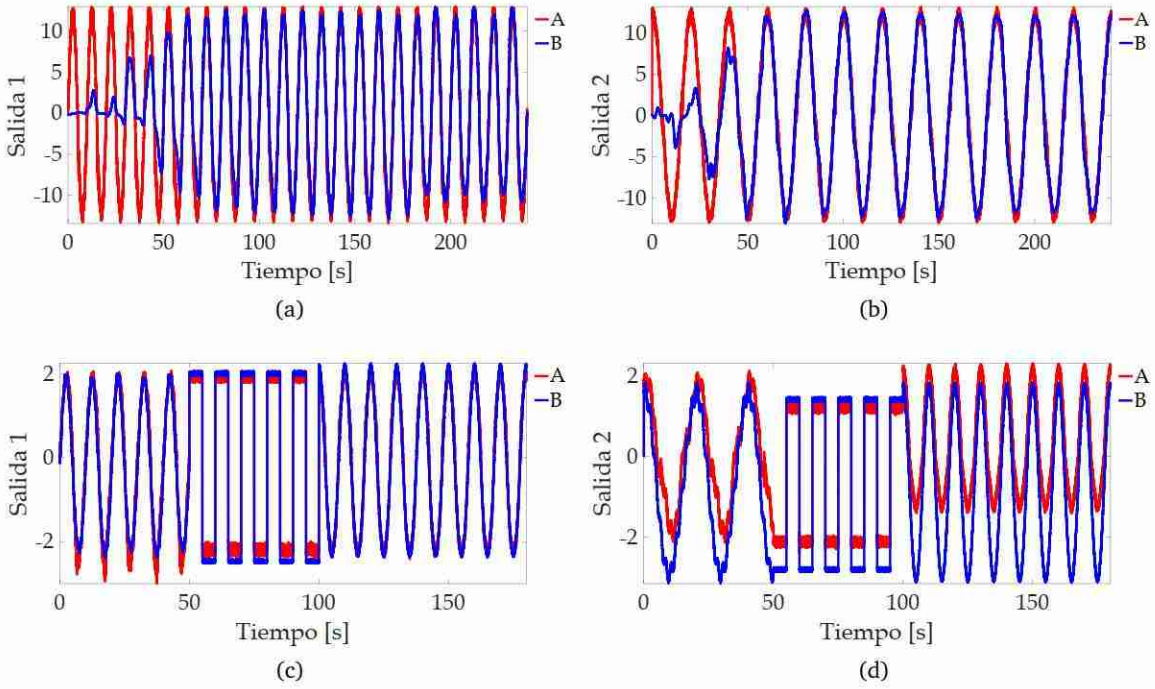


Figura 3.11. Resultados del ejercicio de la Subsección 3.3.2 con la WN difusa: (a) y (b) corresponden al entrenamiento, mientras que (c) y (d) corresponden a la generalización. En las subfiguras: la "Salida 1" representa a $y_{d_1}(k)$, la "Salida 2" a $y_{d_2}(k)$, "A" a la salida del sistema y "B" a la salida estimada por la red.

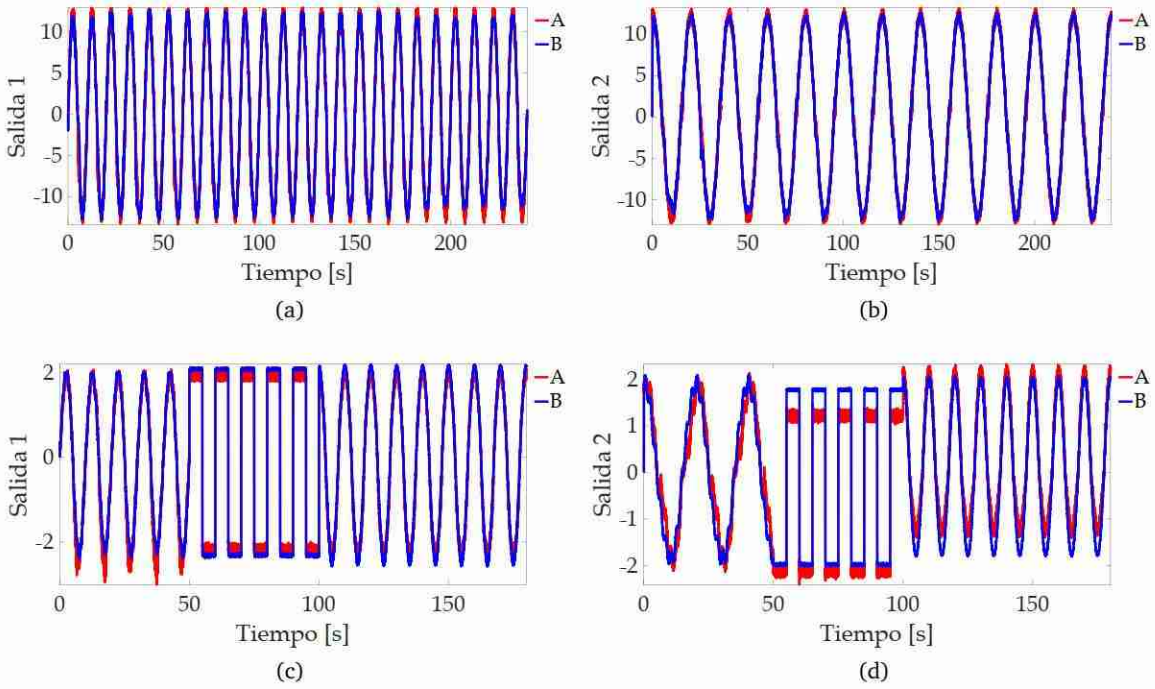


Figura 3.12. Resultados del ejercicio de la Subsección 3.3.2 con la Red Recurrente LSTM Difusa: (a) y (b) corresponden al entrenamiento, mientras que (c) y (d) corresponden a la generalización. En las subfiguras: la "Salida 1" representa a $y_{d_1}(k)$, la "Salida 2" a $y_{d_2}(k)$, "A" a la salida del sistema y "B" a la salida estimada por la red.

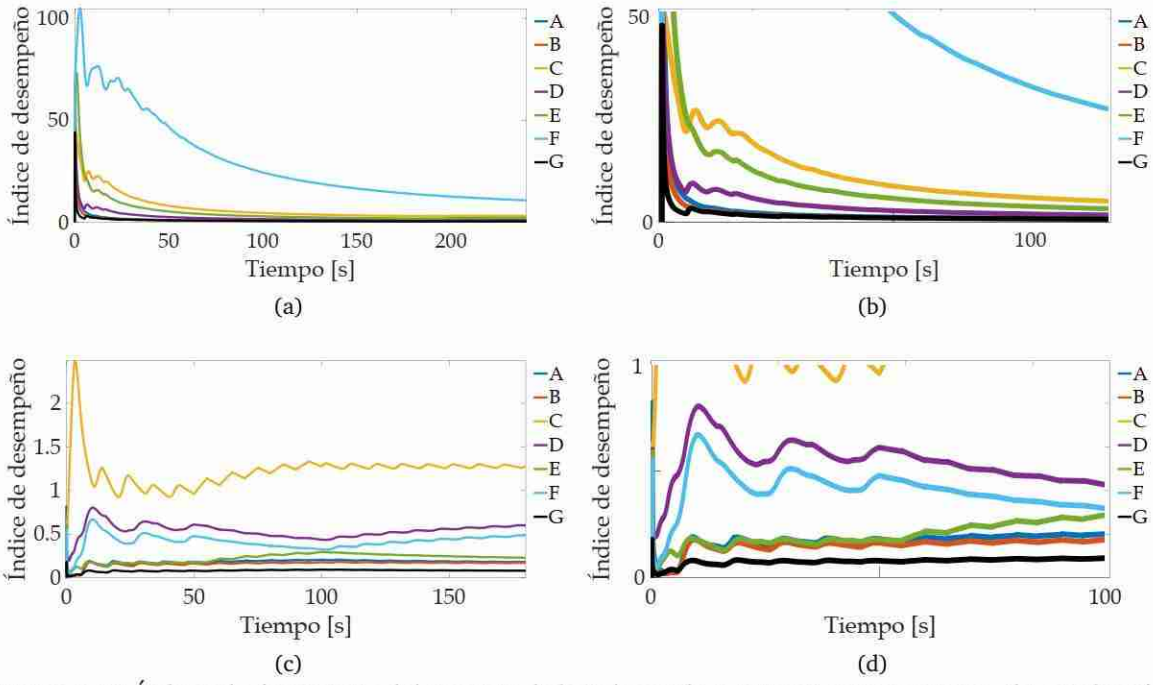


Figura 3.13. Índices de desempeño del ejercicio de la Subsección 3.3.2: (a) entrenamiento, (b) ampliación de la gráfica del entrenamiento (120 s), (c) generalización y (d) ampliación de la gráfica de la generalización (100 s). En las subfiguras: "A" representa a la KFRNN, "B" a la red LSTM, "C" al ANFIS de orden 0, "D" al ANFIS de orden 1, "E" a la KFRNN difusa, "F" a la WN difusa y "G" a la Red Recurrente LSTM Difusa.

Tabla 3.2. Comparación entre los valores de los índices de desempeño de los algoritmos en el ejercicio de la Subsección 3.3.2.

Sistema	Entrenamiento	Generalización	
		Inmediata después del entrenamiento	Con otro tipo de señal de entrada
	Iteración 18,001	Iteración 24,001	Iteración 18,001
KFRNN	52.56×10^{-2}	57.95×10^{-2}	17.95×10^{-2}
Red LSTM	48.21×10^{-2}	58.53×10^{-2}	16.94×10^{-2}
ANFIS de orden 0	303.21×10^{-2}	315.14×10^{-2}	127.88×10^{-2}
ANFIS de orden 1	102.80×10^{-2}	104.38×10^{-2}	59.86×10^{-2}
KFRNN difusa	182.69×10^{-2}	223.30×10^{-2}	22.93×10^{-2}
WN difusa	$1{,}382.32\times10^{-2}$	$1{,}071.40\times10^{-2}$	48.20×10^{-2}
Red Recurrente LSTM Difusa	43.69×10^{-2}	42.06×10^{-2}	8.04×10^{-2}

Para el aprendizaje y la generalización de los algoritmos, se tomó un periodo de muestreo $T=0.01$s. El aprendizaje se realizó en un lapso de 180s (18,001 iteraciones), con una comprobación adicional de 180-240s (6, 000 iteraciones), resultando en un proceso de 240s en total. La generalización se consideró en un lapso de 180s (18,001 iteraciones). La Figura 3.13 muestra la evolución de los índices de desempeño de los algoritmos, la Tabla 3.2 muestra algo similar usando

el valor de los índices de desempeño.

3.3.3. Discusión de los resultados obtenidos

En la Subsección 3.3.1, desempeño entre los algoritmos es muy parecido (ver Figura 3.5 y Tabla 3.1) indicando que estos aprendieron con una eficiencia parecida. Pero, al observar la Figura 3.4, el seguimiento de trayectoria de la propuesta (Figura 3.4 (g)) y el de los demás algoritmos (Figuras 3.4 (a)-(f)) es diferente; con la propuesta se tiene un seguimiento bueno de la serie de tiempo, mientras que los otros algoritmos no.

Por su parte, en la Subsección 3.3.2, en el aprendizaje, así como en la comprobación inmediata de éste, se observa un rendimiento similar entre los algoritmos a excepción de la WN difusa (Figuras 3.6 (a) y (b), 3.7 (a) y (b), 3.8 (a) y (b), 3.9 (a) y (b), 3.10 (a) y (b), 3.11 (a) y (b), 3.12 (a) y (b), 3.13 (a) y (b), y la Tabla 3.2). En la generalización, se observó una diferencia notoria en el rendimiento de los algoritmos; el algoritmo con mejor rendimiento, es decir, el que obtuvo el índice de desempeño más bajo, incluyendo un seguimiento adecuado de la señal deseada, fue la red propuesta (Figuras 3.6 (c) y (d), 3.7 (c) y (d), 3.8 (c) y (d), 3.9 (c) y (d), 3.10 (c) y (d), 3.11 (c) y (d), 3.12 (c) y (d), 3.13 (c) y (d), y la Tabla 3.2).

Finalmente, en la comparación de los sistemas difusos con la propuesta, se nota el efecto de cambiar la forma en que se calcula la parte de ENTONCES de las reglas difusas, retomando a Kabziński y Kacerka [131], los sistemas difusos TSK mejoran su estimación cuando se trata esa parte como una función no lineal. La propuesta en comparación con las redes neuronales, obtuvo un rendimiento similar en cuanto a velocidad de aprendizaje, pero con un modelo más completo del sistema como el que ofrecen los sistemas difusos.

3.4. Conclusiones del capítulo

Se desarrolló la propuesta de esta tesis, la llamada Red Recurrente LSTM Difusa. Esta red ofrece un mejor rendimiento en comparación a otros algoritmos inteligentes; esto se observó en la identificación de sistemas, en donde se compararon algunos algoritmos seleccionados con la nueva red. La mayor ventaja de esta red es su rapidez de aprendizaje, la cual compensa el tamaño de su estructura, que es considerada como su mayor inconveniente.

Capítulo 4

Red Recurrente LSTM Difusa para control de sistemas no lineales

Una vez establecidas las condiciones de operación de la Red Recurrente LSTM Difusa, se tiene la capacidad de usar la red propuesta en distintas aplicaciones, aparte de la identificación de sistemas, como lo es el control de sistemas. En el presente capítulo, se habla de la forma adecuada de realizar dicho control, en específico, se describen tres esquemas en los que la red propuesta puede ser empleada para controlar sistemas no lineales.

4.1. Control de sistemas usando algoritmos inteligentes

El control de sistemas no lineales fue una de las principales motivaciones para el diseño de la Red Recurrente LSTM Difusa, por lo que a continuación se dan las bases para realizar esta tarea. Se trabajó solo con sistemas definidos en tiempo discreto para la fácil aplicación de la red propuesta. Cabe destacar que los esquemas usados para el control de sistemas no lineales también se pueden usar para el control de sistemas lineales, tomando las consideraciones adecuadas.

Controlar un sistema consiste en alterar su comportamiento natural, de manera que la respuesta del sistema cumpla determinados criterios de desempeño. Generalmente, el proceso para controlar un sistema parte de la creación de un nuevo sistema, al que se denomina controlador, que utiliza la información disponible del sistema que se quiere controlar, denominado como planta.

De modo que, al utilizar el controlador en conjunto con la planta, el controlador modificará los parámetros de la planta para obtener una respuesta deseada por parte de ésta.

Entonces, el diseño de controladores inicia con un análisis de la planta, con el fin de obtener parámetros que sirvan para controlarla, para lo cual hay varias técnicas en el control automático. Los parámetros necesarios para controlar una planta se pueden representar por medio de un modelo, en donde se descartan los parámetros que no son de interés para realizar el control. Como se mencionó en el Capítulo 1 y en la Sección 3.3, los sistemas inteligentes son capaces de obtener modelos de un sistema, con el conocimiento de sus señales de entrada y salida. Además, el ajuste de un sistema inteligente, al modelo de la planta de interés, se realiza por medio de procesos de aprendizaje que modifican la estructura interna del sistema.

Entre las diferentes formas en que los sistemas inteligentes pueden crear modelos de una planta destacan tres esenciales: por medio de un modelo funcional de la planta que muestra la relación entre su entrada y salida, que puede resultar en un mapeo lineal o no lineal (dependiendo de la planta) entre dichas señales; por medio de un modelo recursivo, en donde se utilizan señales retardadas de la entrada y salida de la planta para predecir su comportamiento actual o futuro; y por medio de la representación de los estados de la planta. El modelo recursivo es la metodología más popular, ya que es fácil de obtener y se ajusta perfectamente a los sistemas definidos en tiempo discreto.

Al modelo recursivo que usa señales retardadas de la señal de entrada y la señal de salida de un sistema se le ha denominado como modelo NARX, Siegelmann et al. [132], del término en inglés “nonlinear auto-regressive with exogenous inputs model” (modelo no lineal auto-regresivo con entradas exógenas). La palabra exógeno se refiere a algo externo al sistema, pero en realidad se trata de señales relacionadas con el sistema, que se toman en diferentes lapsos de tiempo durante la operación del sistema. Los sistemas inteligentes más adecuados para trabajar con modelos NARX son las redes neuronales y los sistemas difusos.

Con la Red Recurrente LSTM Difusa se pueden generar fácilmente modelos NARX, los cuales se pueden aprovechar en el diseño de controladores. En la siguiente sección se habla de tres esquemas de control para plantas no lineales emplean a la red propuesta, en estos esquemas también se hace mención de un control auxiliar; a pesar de que la red aprende la dinámica del sistema en cuestión con un buen índice de desempeño (valor cercano a cero) éste nunca es cero,

siempre se tendrá un error de modelado muy pequeño en el mejor de los casos, por lo que con el control auxiliar se busca minimizar este error aún más en el contexto del error de control.

4.2. Diseño de controladores con la Red Recurrente LSTM Difusa

En esta sección, se habla de tres tipos de esquemas de control, empleando el algoritmo de la Red Recurrente LSTM Difusa. El proceso de control, en general, se sintetiza así: los controladores se alimentan con datos muestreados de la respuesta generada por la planta a controlar, la cual es modificada con el control propuesto en cada caso. Los errores de los que se hace uso para entrenar y medir la eficacia de los algoritmos, similar a lo planteado en (3.26)-(3.28), son definidos como:

$$E(k)=y(k)-\hat{y}(k) \tag{4.1}$$

$$E_C(k)=r(k)-y(k) \tag{4.2}$$

donde (4.1) corresponde al error de identificación entre la respuesta de la planta y la respuesta del modelo de la planta obtenido con la Red Recurrente LSTM Difusa, y (4.2) corresponde al error de control o seguimiento entre la referencia deseada para la planta y la respuesta de la planta.

También se hace uso de un segundo control que está definido de la siguiente forma:

$$u_A(k)=u_A(k-1)+0.1T\sqrt{|E_C(k-1)|}\text{sign}(E_C(k-1)) \tag{4.3}$$

y que es tratado como un controlador auxiliar en las estrategias propuestas, con la finalidad de garantizar que el valor de (4.2) tienda a un número cercano a cero.

Las Redes Recurrentes LSTM Difusas de las que se usaron en los esquemas de control se alimentan con dos señales de entrada y ofrecen una señal de salida, generando así nueve reglas difusas para la creación del modelo NARX requerido en cada esquema. Esto, de conformidad a lo que se estableció en la Sección 3.2, por lo que las dimensiones de la red en cada caso son fijadas tal que $m=2$, $l=1$ y $p=9$ considerando como mínimo $\kappa=3$ conjuntos difusos.

Los esquemas de control son presentados en el siguiente orden: la Subsección 4.2.1 muestra un control basado en el modelo de la planta a controlar, la Subsección 4.2.2 muestra un control

basado en el error de seguimiento (4.1) , y la Subsección 4.2.3 muestra un control basado en la referencia y el error de seguimiento (4.2).

4.2.1. Control basado en el modelo de la planta a controlar

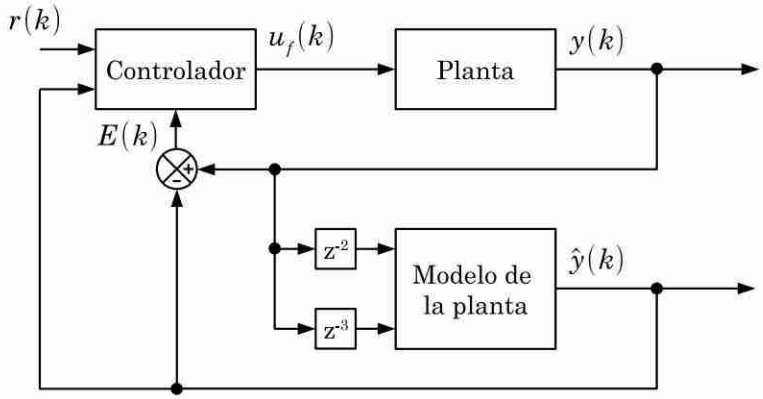


Figura 4.1. Esquema de control basado en el modelo de la planta a controlar: la Red Recurrente LSTM Difusa es usada para obtener un modelo NARX de la planta que sirve para el control de la misma.

Este esquema de control usa un modelo NARX de la planta, ver Figura 4.1, y se basa en el error mostrado en (4.2), de forma que:

$$E_C(k+1)=E_C(k)+u(k) \tag{4.4}$$

$$E_C(k+1)=r(k)-y(k)+u(k) \tag{4.5}$$

Se tiene que (4.4) es un sistema lineal en torno al error de control, $E_C(k)$, lo que se pretende con este sistema es llevar al error de control a un valor cercano a cero por medio de un control $u(k)$. Por tanto, a partir de (4.5) y (4.1), se tiene:

$$u(k)=-r(k)+\hat{y}(k)+E(k) \tag{4.6}$$

y proponiendo un control similar a (4.3):

$$u_f(k)=u_f(k-1)+0.5T\sqrt{|u(k-1)|}\text{sign}(u(k-1)) \tag{4.7}$$

se garantiza que el error de control tienda a cero en el ejercicio propuesto al inicio de la sección.

4.2.2. Control basado en el error de seguimiento

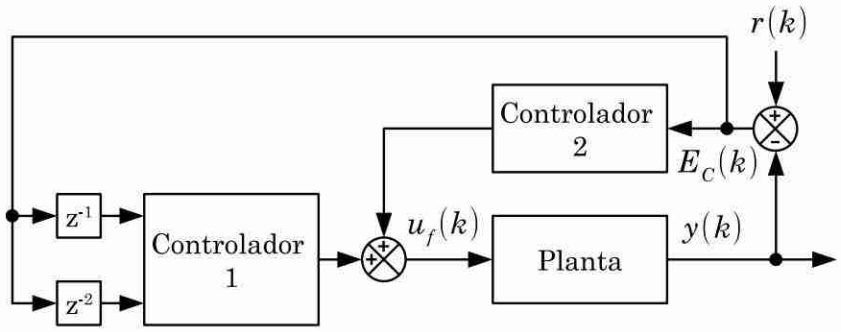


Figura 4.2. Esquema de control basado en el error de seguimiento: la Red Recurrente LSTM Difusa y el control auxiliar generan la señal de control para la planta usando el error de seguimiento.

En este tipo de control se emplea al error de seguimiento, es decir, el conocimiento de los valores de (4.5); el modelo NARX del controlador es entonces estimado por la Red Recurrente LSTM Difusa a partir de este conocimiento. Además, se hace uso de (4.3) como un control auxiliar al control generado por la red, el cual se aplica después del entrenamiento de la red, con el fin de garantizar un error mínimo de seguimiento.

En la Figura 4.2, el "Controlador 1" corresponde a la red y el "Controlador 2" corresponde al control auxiliar. Este esquema se basa en la siguiente idea:

$$x(k+1)=f(k)+g(k)u(k) \tag{4.8}$$

$$u(k)=\frac{1}{g(k)}u_f(k) \tag{4.9}$$

Por otro lado, (4.8) y (4.9) son una representación de un sistema no lineal del tipo afín; si se asume $g(k)\approx 1$, entonces solamente se tiene que preocupar por la obtención de $u_f(k)$ por medio de la Red Recurrente LSTM Difusa.

La red es capaz de obtener la señal de control que minimiza el error de control, si esta red es usada como se sugiere en la Figura 4.2, y que durante su proceso de entrenamiento el error con el que se entrena (índice de desempeño) sea el error descrito por (4.2), el error de seguimiento. El control auxiliar (4.3) se utiliza después de que la red termina su proceso de aprendizaje. Esto,

porque la red no siempre logra captar toda la dinámica de un sistema, en la práctica es común que la red capte la mayor parte del sistema, pero siempre se tendrá un error, la mayoría de ocasiones mínimo, entre la red y el sistema que aprendió. Por lo que el control auxiliar ayuda a que ese error, remanente en la red, no afecte el resultado final del proceso de control.

4.2.3. Control basado en la referencia y el error de seguimiento

Este esquema de control es fundamentalmente igual al de la subsección anterior, pero con un ligero cambio; en vez usar en la red valores retardados de la señal del error de seguimiento, se usan valores retardados de la señal de referencia. Por lo que, en la Figura 4.2, el controlador que corresponde a la Red Recurrente LSTM Difusa, en lugar de ser alimentado con el error de seguimiento, $E_C(k)$, es alimentado con la referencia, $r(k)$, pero manteniéndose lo establecido anteriormente para el entrenamiento de esta red, así como sus parámetros internos y el uso de un control auxiliar.

Comparando esquemas, el de la Subsección 4.2.2 se compone de dos controladores, mientras que el de esta Subsección, a pesar de que se compone también de dos controladores, la información con la que se obtiene la señal de control proviene solamente del error de control; motivo por el cual las señales que se obtienen de los controladores deben de ser distintas. En la siguiente sección se compara el rendimiento de cada controlador expuesto con anterioridad, por medio de la tarea de controlar un sistema no lineal, mostrando el comportamiento de las señales involucradas en cada caso.

4.3. Comparación del rendimiento de cada controlador

Los controladores que se diseñaron con anterioridad se comparan, a través del control de un sistema no lineal, en esta sección. Estos esquemas fueron construidos y simulados en el software MATLAB, tal y como los demás ejercicios mostrados en la tesis hasta el momento. La planta a controlar se tomó de uno de los ejemplos mostrados por Sastry et al. [130] y se expresa en la siguiente ecuación:

$$y(k+1)=0.5\left[\frac{y(k)}{1+y^2(k)}+(1+u(k))u(k)(1-u(k))\right] \tag{4.10}$$

en donde: $y(k)\in\mathbb{R}$ representa a la respuesta de la planta y $u(k)\in\mathbb{R}$ representa la señal de control que modifica a la planta. Entonces, el objetivo de control de los esquemas es que la respuesta de

la planta (4.10), $y(k)$, siga el comportamiento de la siguiente señal de referencia:

$$r(k)=\begin{cases} \text{Si } kT \leq 200, \quad r(k)=0.5\text{sen}\left(\dfrac{2\pi kT}{60}\right) \\ \text{Si } kT > 200 \,\&\, kT \leq 220, \quad r(k)=0.5 \\ \text{Si } kT > 220 \,\&\, kT \leq 240, \quad r(k)=-0.5 \\ \text{Si } kT > 240 \,\&\, kT \leq 260, \quad r(k)=0.5 \\ \text{Si } kT > 260 \,\&\, kT \leq 280, \quad r(k)=-0.5 \\ \text{Si } kT > 280 \,\&\, kT \leq 300, \quad r(k)=0.5 \\ \text{Si } kT > 300, \quad r(k)=0.4\text{sen}\left(\dfrac{2\pi kT}{20}\right) \end{cases} \tag{4.11}$$

siendo: k las iteraciones y T el periodo de muestreo para el proceso, respectivamente.

Las Redes Recurrentes LSTM Difusas que se usaron en los esquemas de control se alimentan con dos señales de entrada y ofrecen una señal de salida, generando así nueve reglas difusas para la creación del modelo NARX requerido en cada esquema. Esto, de conformidad a lo que se estableció en la Sección 3.2, por lo que las dimensiones de las redes en cada caso son fijadas como $m=2$, $l=1$ y $p=9$, considerando un mínimo de $\kappa=3$ conjuntos difusos.

Además, para medir la adaptabilidad y robustez de los esquemas de control propuestos, se agregó una perturbación en cada caso. La perturbación consiste en la generación de números aleatorios, en el rango de valores $[-0.1,0.1]$, a la señal que alimenta a las redes en el periodo de tiempo de 360−380s (2,001 iteraciones). Los esquemas de control son presentados en el siguiente orden: primero el de la Subsección 4.2.1, después el de la Subsección 4.2.2, y por último el de la Subsección 4.2.3

Iniciando con la Subsección 4.2.1, Figura 4.1, y lo dicho al inicio de la sección: el modelo NARX de la planta es generado por la Red Recurrente LSTM Difusa, utilizando retardos de la planta, siendo $U(k)=[y(k-2),y(k-3)]^T$ para la red, además esta red es entrenada con el error mostrado en (4.1). Las variables del proceso de entrenamiento para la red se fijaron en $\eta=0.03$ y $\alpha=0.005$, mientras que los pesos iniciales de la red se definieron aleatoriamente en el intervalo $[-0.1,0.1]$. Los resultados de este ejercicio se muestran en la Figura 4.3.

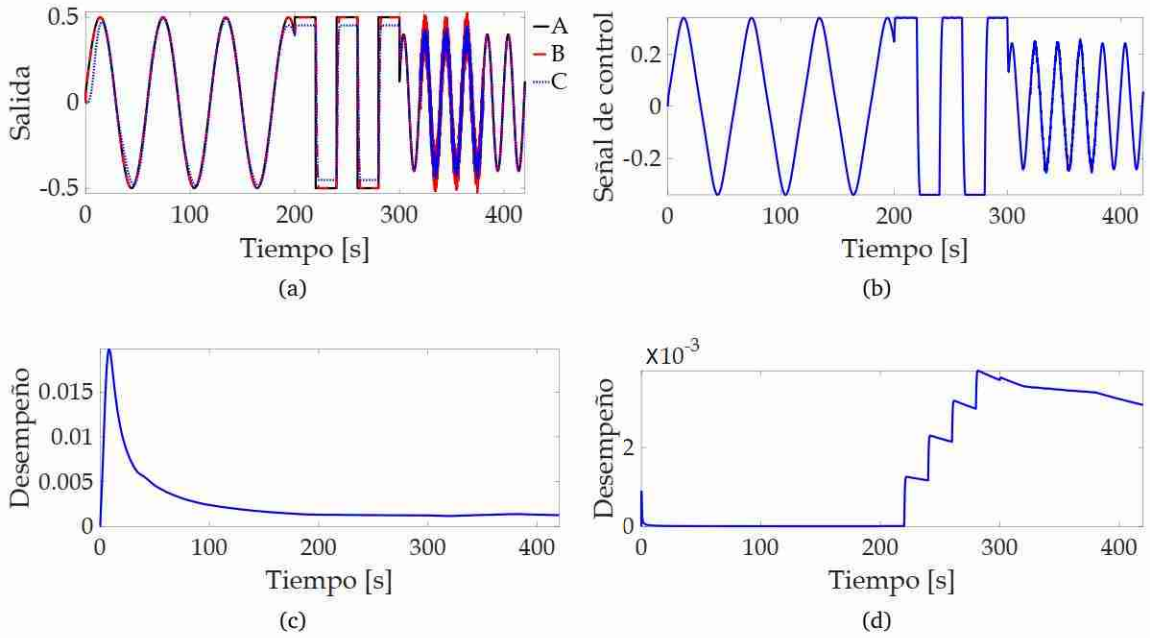


Figura 4.3. Resultados del esquema de control de la Subsección 4.2.1: en (a) se muestra el comportamiento de la referencia "A", la planta "B" y la Red Recurrente LSTM Difusa "C"; en (b) se muestra la señal de control generada por el esquema propuesto; en (c) se muestra el índice de desempeño de la red; y en (d) se muestra el índice de desempeño del controlador.

Tabla 4.1. Índices de desempeño de la Red Recurrente LSTM Difusa y el controlador en la Subsección 4.2.1 durante los eventos destacados del proceso.

Evento	Iteración	Índice de desempeño	
		Red	Controlador
Fin del entrenamiento	18,001	1.40×10^{-3}	3.21×10^{-6}
Antes de la perturbación	36,000	1.10×10^{-3}	3.60×10^{-3}
Después de la perturbación	38,001	1.40×10^{-3}	3.40×10^{-3}
Final del proceso	42,001	1.20×10^{-3}	3.10×10^{-3}

La Figura 4.3 se compone de cuatro subfiguras: en (a) se muestra el seguimiento de la planta y de la red con respecto de la referencia, el cual, en ambos sistemas, es aceptable; en (b) se muestra la señal de control generada por el controlador expuesto en (4.7); en (c) se muestra el índice de desempeño de la red para la identificación del sistema, el cual se mantiene en un valor cercano a cero, que indica que la red aprendió la dinámica de la planta satisfactoriamente; y en

(d) se muestra el índice de desempeño del controlador, que se mantiene en un valor cercano a cero, indicando que el control fue efectivo.

Asimismo, en la Tabla 4.1 se muestra el valor del índice de desempeño de tanto la Red Recurrente LSTM Difusa que se uso para identificar la planta, así como del controlador; se muestra el valor de los índices de desempeño en diferentes situaciones, concordando con lo que se muestra en la Figura 4.3 y sus subfiguras. El valor de ambos índices de desempeño se mantiene cercano a cero, por lo que ambos algoritmos se ejecutaron de manera satisfactoria.

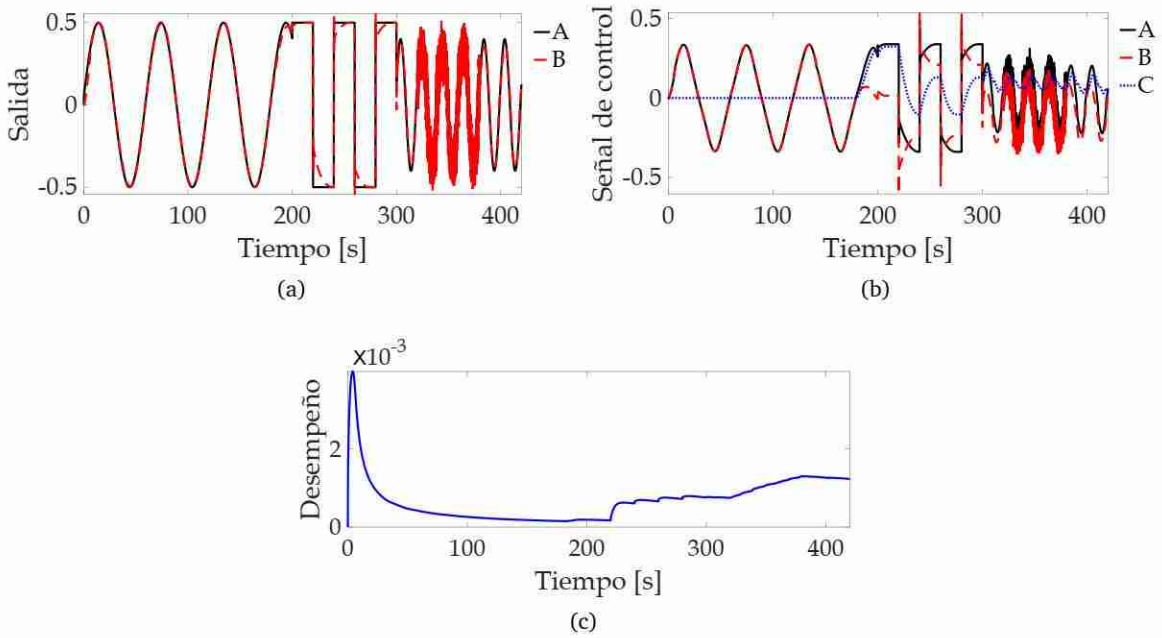


Figura 4.4. Resultados del esquema de control de la Subsección 4.2.2: en (a) se muestra el comportamiento de la referencia "A" y la planta "B"; en (b) se muestra el comportamiento de la señal de control generada por el esquema propuesto "A", la contribución de la red "B" y la contribución del control auxiliar "C"; y en (c) se muestra el índice de desempeño del controlador.

Tabla 4.2. Índices de desempeño de la Red Recurrente LSTM Difusa en el esquema de control de la Subsección 4.2.2 durante los eventos destacados del proceso.

Evento	Iteración	Índice de desempeño
Fin del entrenamiento	18,001	1.51×10^{-4}
Antes de la perturbación	36,000	7.46×10^{-4}
Después de la perturbación	38,001	1.30×10^{-3}
Final del proceso	42,001	1.20×10^{-3}

Luego, para el control de la Subsección 4.2.1, los valores de los parámetros de la Red Recurrente LSTM Difusa se seleccionaron $\eta=0.7$ y $\alpha=0.01$, mientras que los valores de los pesos sinápticos se establecieron aleatorios en el rango $[-0.1,0.1]$. La entrada de la red se cambia de tal forma que $U(k)=[E_C(k-1),E_C(k-2)]^T$ para la red. Los resultados de este ejercicio se muestran en la Figura 4.4: en (a) se muestra el seguimiento de la planta con respecto de la referencia, el cual es aceptable; en (b) se muestra la señal de control generada por la red y el control auxiliar, el control auxiliar se activa después del entrenamiento de la red y su participación en la señal de control total es pequeña en comparación a la de la red; y en (c) se muestra el índice de desempeño del control, que se mantiene en un valor cercano a cero, indicando que el control fue efectivo.

La principal diferencia entre este esquema y el anterior es la generación de la señal de control; mientras que en el esquema de la Subsección 4.2.1 la señal proviene de un solo controlador y no es tan abrupta, en este esquema la señal se compone de dos controladores y tiene un comportamiento más abrupto. La Tabla 4.2 muestra el valor del índice de desempeño del controlador en diferentes situaciones, concordando con lo que se muestra en cada subfigura de la Figura 4.4, el valor del índice se mantiene cercano a cero indicando que el control fue exitoso.

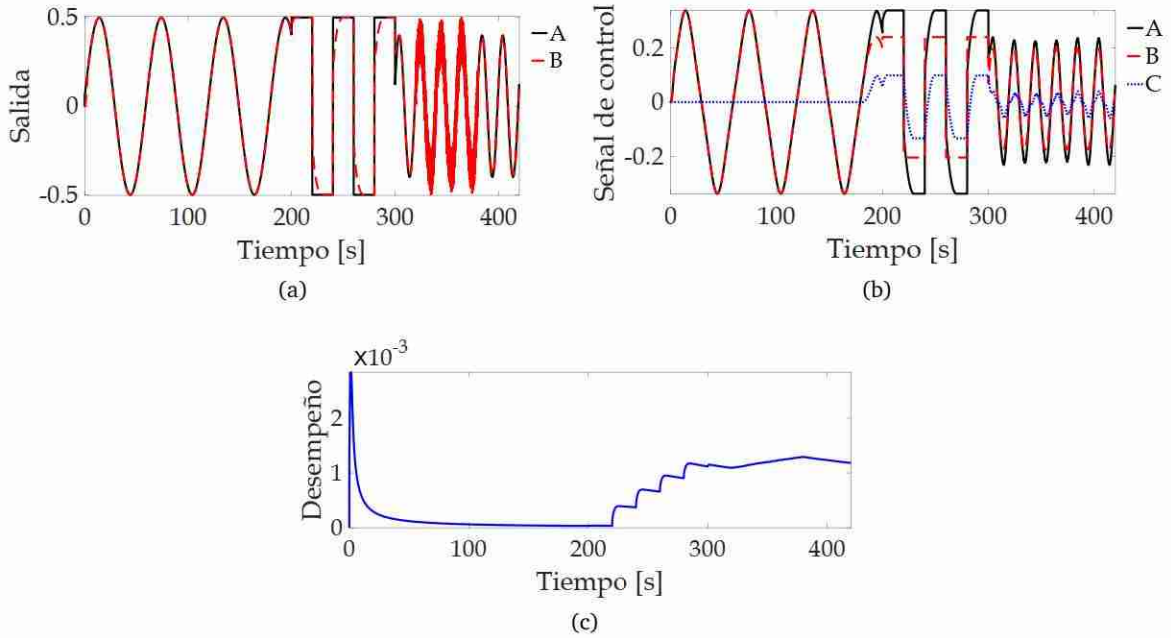


Figura 4.5. Resultados del esquema de control de la Subsección 4.2.3: en (a) se muestra el comportamiento de la referencia "A" y la planta "B"; en (b) se muestra la señal de control generada por el esquema propuesto"A", la contribución de la red "B" y la contribución del control auxiliar "C"; y en (c) se muestra el índice de desempeño del controlador.

Tabla 4.3. Índices de desempeño de la Red Recurrente LSTM Difusa en el esquema de control de la Subsección 4.2.3 durante los eventos destacados del proceso.

Evento	Iteración	Índice de desempeño
Fin del entrenamiento	18,001	3.57×10^{-5}
Antes de la perturbación	36,000	1.10×10^{-3}
Después de la perturbación	38,001	1.30×10^{-3}
Final del proceso	42,001	1.20×10^{-3}

De forma similar, para el esquema de la Subsección 4.2.2, se mantienen las mismas condiciones que en el caso anterior para la red, excepto la entrada que cambia a $U(k)=[r(k-1),r(k-2)]^T$, y los resultados de este ejercicio se muestran en la Figura 4.5: en (a) se muestra el seguimiento de la planta con respecto de la referencia, el cual es aceptable; en (b) se muestra la señal de control generada por la red y el control auxiliar, el control auxiliar se activa después del entrenamiento de la red y su participación en la señal de control total es pequeña en comparación a la de la red, como fue visto en la antes; y en (c) se muestra el índice de desempeño del control, que se mantiene en un valor cercano a cero, indicando que el control fue efectivo.

La diferencia más notoria que existe entre este esquema de control y el anterior está en la forma de la señal de control. En el esquema de la Subsección 4.2.2 la señal de control se compone de dos controladores y tiene un comportamiento abrupto, mientras que en este esquema, a pesar de que la señal de control se compone también de dos controladores, la señal tiene un comportamiento menos abrupto, similar al de la Subsección 4.2.1 que solo utiliza un controlador. En general, se busca que la señal de control obtenida de los controladores no sea tan abrupta, esto, con el fin de mantener en buen estado a los componentes físicos de la planta que son los que interactúan con esta señal, pero también dependerá de la aplicación en particular si la señal de control deseada es abrupta o no.

Por último, en la Tabla 4.3 se muestra el valor del índice de desempeño del controlador; se muestra el valor del índice de desempeño en diferentes situaciones, concordando con lo que se muestra en cada subfigura de la Figura 4.5, el valor del índice se mantiene cercano a cero indicando que el control fue exitoso. Si se comparan las Tablas 4.1, 4.2 y 4.3, el comportamiento de los tres esquemas de control es satisfactorio, ya que en todos los casos se presenta un buen seguimiento

de la referencia deseado por parte de la salida de la planta y además se tiene robustez ante la perturbación que se agrega en las simulaciones.

4.4. Conclusiones del capítulo

Tres esquemas para control de sistemas no lineales fueron establecidos, estos esquemas se benefician de la capacidad de aprendizaje de la Red Recurrente LSTM Difusa y de un control auxiliar. Estos esquemas pueden ser replicados para diversos tipos de sistemas no lineales, y al igual que en los resultados mostrados en este capítulo, se puede esperar un error aceptable entre la respuesta del sistema a controlar y el comportamiento deseado para éste.

Capítulo 5

Evaluación de estructuras con la Red Recurrente LSTM Difusa

Una área de investigación relevante en la actualidad es la salud estructural, dado que ésta permite extender la vida útil de estructuras físicas y crear un ambiente seguro en torno a ellas. La salud estructural es una materia interdisciplinaria, en la cual se combinan elementos de la mecánica con elementos de la informática. En este capítulo, se habla de una metodología que incorpora a la Red Recurrente LSTM Difusa en esta área.

5.1. Sistemas inteligentes en el monitoreo de salud estructural

El monitoreo de salud estructural se ha convertido en una necesidad en aplicaciones pertenecientes a la ingeniería aeroespacial, la ingeniería mecánica y la ingeniería civil. Su utilidad radica en la validación de estructuras físicas, es decir, evalúa si una estructura es capas de realizar o seguir realizando las funciones para las que fue diseñada. Salud estructural se puede definir como la adquisición, validación y análisis de datos técnicos para facilitar las decisiones de la gestión del ciclo de vida de una estructura; implica la creación de sistemas capaces de detectar cambios en una estructura debido al daño provocado por fenómenos externos o las operaciones en dicha estructura.

Una estructura puede cambiar su comportamiento o actuación a base de un cambio súbito o gradual en su estado, condiciones de carga, y sus mecanismos de respuesta. Dentro de las principales

ventajas de desarrollar un sistema de monitoreo de salud estructural se tienen: el crear un ambiente seguro, incrementar la confiabilidad de operación de la estructura, extender la vida útil de los componentes de la estructura, reducir las pruebas y mantenimiento de la estructura, reducir el tiempo muerto en el uso de la estructura y, por tanto, reducir costos que involucran la operación de la misma.

Algunos términos importantes y recurrentes, dentro del área de la salud estructural, son los siguientes:

- Daño: deficiencia o deterioro en la composición de una estructura, causado por cargas externas, condiciones ambientales o errores humanos.
- Detección: identificación de la existencia de condiciones anómalas en un sistema.
- Diagnóstico: identificación de la causa de condiciones anómalas.
- Pronostico: predicción de la cantidad restante de vida útil de un sistema.
- Monitoreo: se refiere al análisis de los datos obtenidos de sensores colocados en una estructura para determinar el estado de la misma.
- Falla: cualquier acción que conduzca a una incapacidad en alguna parte, o en su totalidad, de una estructura o una máquina para funcionar de manera prevista.

El daño puede ser visible, como lo son las grietas, las deformaciones, la reducción de superficies o la exfoliación; asimismo, también puede ser imperceptible, como en el caso de estructuras complejas en las que resulta difícil observar los elementos que las componen. Generalmente, las estructuras tienen cierta tolerancia al daño, pero si el daño se sale de la tolerancia una detección a tiempo puede impedir una falla. La detección implica la recopilación y procesamiento de datos, ya sea de forma activa o pasiva.

Un problema clave, en salud estructural, es la detección y aislamiento de daño en una estructura a partir de datos resultantes del análisis de mediciones realizadas en diferentes partes específicas de esta estructura. Usualmente, los datos obtenidos de mediciones están contaminados con ruido, así como que el número de mediciones realizadas, en ocasiones, es limitado. Para hacer una evaluación del daño, entonces, un modelo debe ser construido a partir de los datos recopilados y procesados las mediciones. Este modelo del que se habla debe de efectuar un reconocimiento de patrones, el cual mapea la localización del daño estructural, o simplemente la presencia de daño, y los datos que son introducidos al modelo.

A lo largo del tiempo se han desarrollado diversos enfoques para tratar los problemas presentes en la salud estructural, resaltando el uso de los sistemas inteligentes, sobre todo de los sistemas difusos. Los modelos difusos, que se han utilizado para salud estructural, abordan directamente las incertidumbres presentes en los datos de las mediciones, a través de sus procesos de fusificación y defusificación. Además, estos modelos se acoplan bien a la salud estructural, ya que se pueden crear guías fáciles de interpretar debido al empleo de variables lingüísticas.

Entre los trabajos que hacen uso de sistemas inteligentes en salud estructural, se tiene un algoritmo neuro-difuso para detección de daño, que se describe por Ramu y Johnson [133]. En este trabajo se emplea un método conocido como “vertex” para ajustar los parámetros del algoritmo, en particular los parámetros de las funciones de pertenencia de la parte difusa (valores discretos para las funciones de pertenencia), además de dividir el algoritmo en módulos para facilitar el aprendizaje del mismo, obteniendo buenos resultados.

Una propuesta destacada de un algoritmo difuso es hecha por Sawyer y Rao [134], en donde se presenta un procedimiento de detección de fallas estructurales. La lógica difusa, en conjunto con mecanismos de detección continua de fallas, son utilizados para procesar y analizar las incertidumbres dentro de los datos obtenidos de estructuras dañadas. Las asociaciones difusas, entre las respuestas de la estructura y las condiciones de daño, se generaron con simulaciones de elemento finito; estas asociaciones se codificaron en un banco de memoria difuso para formar una base de conocimiento. El algoritmo de inferencia difuso consulta a la base de conocimiento, de tal forma que se infiere la posible localización del daño y el nivel de éste.

Zhao y Chen [135] proponen un sistema difuso de inferencia basado en reglas para diagnosticar y predecir daño en un puente. Esta propuesta es capaz de brindar a los diseñadores de puentes valiosa información, la cual se relaciona con el impacto de los factores del diseño en el deterioro del puente. Otro algoritmo usado para diagnosticar daño en puentes es el que muestran Meyyappan et al. [136], este algoritmo opera en tiempo real y se compone de un sistema difuso y una red neuronal; el sistema difuso determina la presencia de daño, mientras que la red neuronal predice la cantidad de daño en el lugar que indica el sistema difuso.

Por otro lado, Dempsey y Afjeh [137] desarrollaron una herramienta de diagnóstico para detectar daño en ruedas dentadas. Se combinó la lógica difusa con datos obtenidos de dos

tecnologías de medición, análisis de residuos de aceite y vibración; los datos de las vibraciones fueron extraídos de acelerometros y los datos de los residuos de aceite de un sensor comercial especializado. Esta herramienta resultó más exacta que las implementaciones pasadas, las cuales no se basan en sistemas inteligentes.

Soh y Bhalla [138] proponen un modelo de daño para concreto, el cual involucra a un sistema difuso basado en probabilidad, tomando como referencia la rigidez de la estructura bajo estudio para calcular la extensión del daño y también el uso de datos de sensores de impedancia colocados en la estructura. Esto permitió la calibración de transductores de piezo-impedancia en términos de la severidad del daño, sirviendo así como un modelo de daño fenomenológico empírico-práctico para estimar cuantitativamente la severidad del daño en el concreto.

Otro algoritmo neuro-difuso importante es el mostrado por Wang et al. [139], el cual fue desarrollado para el monitoreo de la salud estructural de sistemas de engranajes. Este algoritmo exhibe gran robustez ante el ruido presente en las señales obtenidas de los sensores, que obtienen datos del sistema en cuestión, logrando así mejores resultados en el diagnóstico de daño, incluso mejores que otros algoritmos similares a éste. De manera similar y tomando ventaja de las bases de datos que se pueden crear para sistemas difusos, Zio y Gola [140] desarrollan una red neuro-difusa para la detección de daño en máquinas rotatorias.

Una red especializada conocida como GRNNFA (siglas en inglés de “general regression neural network and fuzzy ART <<adaptive resonance theory>>”, red neuronal de regresión general y difusa ART), que se estableció por Lee y Lam [141], y que combina características de las redes neuronales de regresión y los sistemas difusos ART; obteniendo un entrenamiento rápido y estable para la red, además del incremento mesurado de la estructura de la misma. Esta red fue aplicada a la identificación de daño en un edificio, y al igual que la mayoría de los algoritmos mencionados en este capítulo, ofreció buenos resultados a pesar de las incertidumbres presentes en los datos obtenidos de la estructura.

La combinación entre algoritmos genéticos y sistemas difusos también han sido empleada para la detección de daño, como es el caso del trabajo de Pawar y Ganguli [142], en donde se monitorea la salud estructural en tiempo real de las aspas de un helicóptero. La detección de daño se realiza por medio de mediciones, de desplazamiento y fuerza, derivadas de condiciones de

daño y condiciones sin daño. La combinación de métodos probabilistas y sistemas difusos se puede apreciar en el trabajo de Taha y Lucero [143], en donde se propone un algoritmo de uso general para el reconocimiento de daño estructural; la propuesta consiste en transformar los escasos datos obtenidos de mediciones en un esquema de actualización Bayesiano en conjuntos difusos, para ser usados en una métrica de semejanza para identificar el estado de la estructura.

Por medio de un ANFIS, Zhu y Wu [144] proponen la detección rápida de daño estructural. Este ANFIS fue ideado para monitorear la integridad física de edificios posterior a un sismo, siendo capaz de determinar si es seguro para las personas regresar a un edificio después de un sismo. El algoritmo difuso toma como base a los datos de la dinámica de vibraciones de la estructura de un edifico, los cuales son registrados en determinados intervalos de tiempo. El ANFIS realiza tareas de identificación no paramétrica y predicción de la respuesta de la estructura, tomando en cuenta los datos arrojados por sensores ubicados en diferentes partes del edificio, para determinar en donde se encuentra el daño.

Algunos trabajos en donde se comparan las diversas técnicas de detección y diagnóstico de fallas, incluyendo el uso de la lógica difusa en ello, se tienen a los de Lopez y Sarigul-Klijn [145], Cordón [146], Jiang et al. [147], Thenozhi y Yu [148] y Gordan et al. [149]. Estos trabajos resaltan la importancia de usar varios métodos de detección de daño, con el fin de obtener resultados mas sólidos, razón por la cual los algoritmos difusos en combinación con otros algoritmos de la computación flexible han sido empleados para estas tareas últimamente. Además de otros algoritmos inteligentes, se pueden utilizar otras metodologías, como métodos probabilistas y modelos matemáticos, en combinación de los sistemas difusos para mejorar la toma de decisiones en torno al daño.

La combinación de varios tipos de algoritmos inteligentes ha permitido una mejora en la detección de daño en diversas aplicaciones, estos algoritmos se ajustan a los requerimientos del monitoreo de salud estructural, como lo es principalmente el tratamiento de datos con incertidumbres. Tomando esto como motivación, dado que la Red Recurrente LSTM Difusa tiene mejoras en comparación a otros algoritmos inteligentes similares, la red propuesta se probó en esta área obteniendo buenos resultados. Además, en la metodología que se propuso se hace uso de teléfonos celulares, ya que con la tecnología que poseen estos dispositivos dentro de si mismos se permite la recolección de información de manera eficiente y a bajo costo.

5.1.1. Uso de teléfonos inteligentes en la evaluación de daño estructural

Los trabajos que hacen uso de los teléfonos celulares, o teléfonos inteligentes, en la detección de daño estructural han aumentado en años recientes. Por ejemplo, Zhao et al. [150] comprueban la fiabilidad de los datos obtenidos con los teléfonos celulares en una estación de pruebas para movimientos sísmicos, con resultados favorables; de los datos obtenidos de los teléfonos celulares se obtiene información que indica la presencia de daño estructural en una estación de pruebas que simula ser un edificio, los datos que se obtienen son la variación de las frecuencias modales y el desplazamiento residual.

Paul et al. [151] proponen una metodología de detección de daño estructural en tiempo real, donde se hace uso del llamado internet de las cosas. De esta forma, se genera una red de comunicación entre un dispositivo que analiza los datos en tiempo real de un edificio, teléfonos celulares, y una computadora remota (nube). Esto para tener un sistema de monitoreo continuo, que se alimenta de varias fuentes de información, dando como resultado análisis más robustos en la detección de daño del edificio en cuestión.

Otro sistema de monitoreo es el presentado por Abdullahi et al. [152], en donde se hace uso de las plataformas de software de LabVIEW y de ARDUINO. En este trabajo, de manera similar al anterior, se genera un sistema de monitoreo en tiempo real. Existe una comunicación entre computadoras remotas, teléfonos celulares y otros elementos dentro del edificio, como lo son acelerometos y demás dispositivos móviles.

Una red de microcomputadoras y teléfonos inteligentes es presenta por Morgenthal et al. [153], en la cual estos dispositivos son usados para adquirir y mostrar datos del movimiento de un edificio. En este trabajo, se comprueba que el uso de teléfonos celulares en este tipo de tareas tiene un costo-beneficio positivo, en comparación con alternativas tradicionales. También se exponen algunas desventajas del uso de teléfonos inteligentes, pero se muestran medidas para mitigar tales desventajas, entre las desventajas se tienen: la calidad de la medición varía de dispositivo en dispositivo, así como los tiempos desiguales de muestreo.

En el trabajo de Guzman-Acevedo et al. [154] se hace énfasis en los sensores internos de los teléfonos celulares, como lo son el GPS (siglas en inglés de “global positioning system”, sistema de

posicionamiento global) y los acelerometros, buscando la forma más eficiente de obtener información de ellos. En el artículo se estudia el caso particular de un puente con problemas de vibración, en donde se obtienen datos de desplazamiento estático y dinámico con los cuales se determina si existe un comportamiento estructural indebido en el puente, lo que se definió como daño.

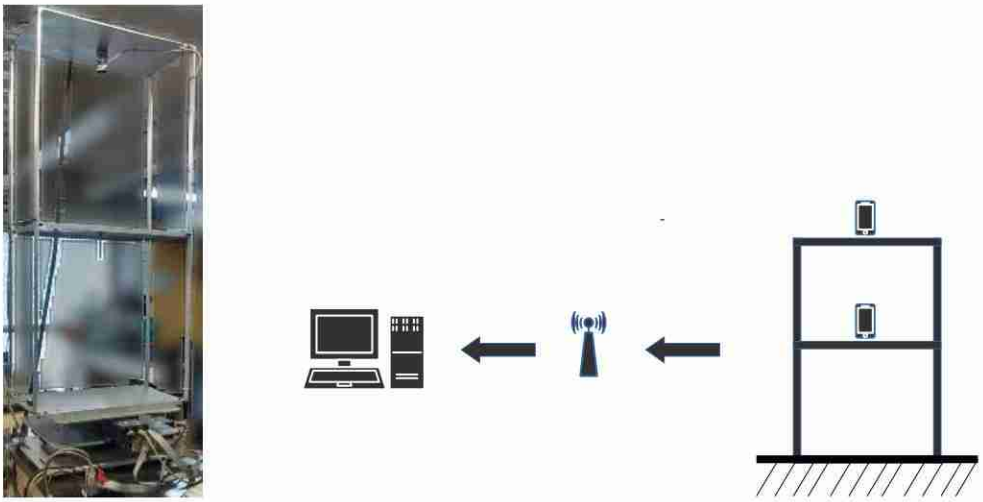
Por otra parte, Ozer y Feng [155] proponen la creación de una red comunitaria formada de teléfonos celulares, en la que varios dispositivos dentro de un edificio aportan información para poder determinar el estado de éste. Similarmente, Zhang et al. [156], por medio de una aplicación en el sistema operativo de teléfonos celulares android, generan la conexión entre varios dispositivos para realizar el diagnóstico de daño en edificios a partir de datos recolectados por los mismos. Cabe mencionar que estos trabajos hacen uso de algoritmos similares a los mencionados anteriormente para llevar a cabo el diagnóstico de daño estructural.

Más trabajos en los que se plantea el uso de redes comunitarias de teléfonos celulares, para detección de daño, se tiene a los elaborados por: Sharma et al. [157], que presentan una red de teléfonos celulares, cámaras inteligentes y drones, en donde el procesamiento de información se hace por medio de inteligencia artificial y aprendizaje profundo aplicado en redes neuronales de tipo convolucional; Alzughaibi et al. [158], en donde se presenta un proyecto a gran escala para monitorear el daño en edificios, el cual está basado en la participación de varias personas dentro de una localidad que aportan información a través de sus teléfonos celulares, además de la adición de dispositivos con acceso a internet como cámaras, tabletas y sensores convencionales colocados en los edificios de interés; Han y Zhao [159], en el que se hace uso de iphones(que usan el sistema operativo iOs) y sensores convencionales conectados a internet para conformar una red de monitoreo en tiempo real de un edificio, siendo relevante el hecho de que el uso de teléfonos celulares no se limita a aquellos dispositivos con android como sistema operativo.

5.2. Aplicación de la Red Recurrente LSTM Difusa en la evaluación de daño estructural

Para observar las capacidades de la Red Recurrente LSTM Difusa en la evaluación de daño en estructuras se desarrolló una aplicación en particular: esta red se implementó en una estación de

pruebas dedicada para probar algoritmos de detección de daño estructural en edificios ocasionado por sismos. La estación está construida de tal forma que se asemeja a un modelo a escala de un edificio, ver Figura 5.1 (a), el cual consta de tres niveles: planta baja, primer y segundo piso. La estructura se mantiene unida por medio de varios tornillos y reposa sobre una base, la base es una plataforma solida de metal unida a dos motores ubicados conforme al plano cartesiano X,Y, el movimiento de los motores simula entonces los movimientos que se generan en un sismo para la estación.



(a) (b)

Figura 5.1. Estación de pruebas para probar algoritmos de detección de daño estructural en edificios debido a sismos: (a) fotografiá del la estación ubicada en el Departamento de Control Automático, (b) esquema para detección de daño estructural propuesto en esta sección que plantea la comunicación inalámbrica entre los dispositivos en la estación y una computadora.

En la estación de pruebas se puede simular el daño estructural de diferentes maneras, por ejemplo, se pueden aflojar los tornillos de determinada sección y hacer que los motores generen un movimiento semejante al registrado por un sismo con anterioridad, y a partir de ello se puede ver que tanto se desajusta la estructura en función de los movimientos a los que se somete. La estación cuenta con sensores en la planta y los dos pisos para medir el efecto del movimiento generado por los motores, pero también se tiene posibilidad de colocar más sensores, según sea el caso del algoritmo que se pruebe en la estación.

El control, y monitoreo, de los motores y los sensores de la estación se realiza a través de una interfaz conectada una computadora ejecutando el programa MATLAB, además, en esta

computadora se tienen varias bases de datos que contienen información de movimientos sísmicos registrados a través del tiempo, así como análisis que se han hecho con anterioridad en la estación.

En la aplicación, se recopilan datos de los dos pisos que componen a la estación, sin considerar a la base o planta baja. Los datos son obtenidos de las mediciones realizadas por medio de teléfonos celulares, en específico de los acelerometros que tienen integrados los teléfonos. La información obtenida es transmitida de la estación de pruebas a una computadora para su procesamiento a través de un módem inalámbrico, ver Figura 5.1 (b), en donde se ejecuta el algoritmo que determina la presencia de daño en la estación. Con lo anterior, se estableció una metodología para la detección del daño estructural, que se basa en la recopilación fácil y rápida de información para su análisis inmediato.

Se tienen varias ventajas que se pueden aprovechar con el uso de los teléfonos celulares, pero también se tienen algunos inconvenientes. Entre las ventajas, se tiene que los teléfonos celulares son dispositivos que integran varios tipos de sensores, así como diferentes tipos de protocolos de comunicación, dando así la posibilidad de desarrollar diferentes aplicaciones que hagan uso de los sensores de los dispositivos: los teléfonos son compactos, su interfaz es sencilla y además son relativamente accesibles económicamente hablando. Las desventajas que se tienen es que los teléfonos necesitan recargar su batería interna para seguir funcionando, no todas las aplicaciones funcionan correctamente en todos los dispositivos, no todos los celulares tienen la misma versión de sistema operativo y, dependiendo de la marca del teléfono, se construyen con diferentes conjuntos de hardware.

La metodología propuesta de detección de daño estructural contempla la colocación de un teléfono al centro de cada piso de la estación de pruebas, ver la Figura 5.1, para recopilar información de la aceleración lineal (con respecto a un eje cartesiano X,Y) en cada piso al momento de simular el movimiento provocado por un sismo. Utilizando una aplicación en los dispositivos, los teléfonos envían datos de la simulación a una computadora, inalambricamente por medio de un módem Wi-Fi exclusivo para esta tarea, en donde estos son almacenados y después procesados para aplicados en la Red Recurrente LSTM Difusa. Aplicando un entrenamiento y ciertas consideraciones especiales en la interpretación de la información obtenida, la red puede ofrecer un diagnóstico que indique la presencia de daño en la estructura de la estación.

Los teléfonos tienen una batería que promedia doce horas de uso continuo, con una capacidad de 1750 mAh, y corresponden a equipos de la marca SONY modelo XPERIA S comercializados en el año 2012, estos equipos tienen el sistema operativo android versión 2.3 y cuentan con un procesador de la marca Qualcomm modelo Snapdragon S3 MSM8260 a 1.5 GHz doble núcleo, con 1 GB de memoria RAM y 32 GB de memoria para almacenamiento interno. Además, los teléfonos tienen integrados los siguientes sensores: un acelerometro, un sensor de campo electromagnetico que se utiliza como sensor de proximidad, un sensor de luz ambiental, un magnetometro, un GPS y un giroscopio. Los parámetros de los sensores de inercia fueron determinados de manera experimental, el 0g offset del acelerometro es 0.12 V en el eje X, 0.71 V en el eje Y.

Por otra parte, el módem que se usa para la comunicación inalámbrica entre los teléfonos y la computadora corresponde a un equipo de la marca Nexxt Solutions, modelo NEBULA 150, el cual cuenta con cuatro puertos LAN de 10/100 Mbps, así como puertos IEEE802.11n inalámbricos. Asimismo, la computadora en la que se recolecta la información, y en donde se ejecuta el algoritmo de detección de daño estructural, fue proporcionada por el departamento de control automático, la cual cuenta con el sistema operativo Windows 10, implementando una copia de la versión 2020 del software MATLAB.

Entonces, se toman mediciones de la estación de pruebas durante un movimiento sísmico considerando dos casos: cuando la estructura no tiene daño y cuando la estructura tiene daño. De esta forma, se obtienen cuatro vectores de medicines (correspondientes a las aceleraciones en X y Y en cada piso) para ambos casos, con los cuales se generan dos modelos usando a la Red Recurrente LSTM Difusa. Las medicines se toman con un periodo de muestro $T = 0.001$s, durante 33s, obteniendo 33,001 datos por medición. Los modelos generados se comparan entre si, con el fin de decidir cual es el mejor para determinar la presencia de daño en la estructura.

5.2.1. Tratamiento de las señales provenientes de los teléfonos en la estructura

Antes de poder describir la operación de la red para detectar daño, primero se aclara el procesamiento de datos con los que trabaja la red. Los celulares envían datos a través de señales de Wi-Fi, por medio de un módem exclusivo para ello, a una computadora que se encarga del procesamiento de datos y de la ejecución del algoritmo de detección de daño. El problema con los

datos recopilados es que no se puede modificar el instante en que se realiza una medición; se puede configurar que los dos teléfonos inicien a enviar datos al mismo tiempo, pero tendrán un periodo de muestreo variable. En algunos casos un dispositivo recolectará más datos que otro, dependiendo de la configuración interna del hardware de cada celular, ya que cada dispositivo ejecuta tareas en segundo plano simultáneamente.

La forma en que se decidió abordar el problema de recolección de datos fue mediante un método de reconstrucción de la señal, concretamente el método conocido como interpolación lineal:

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} \tag{5.1}$$

donde (5.1), el valor a interpolar, depende de dos puntos conocidos dados por las coordenadas (x_0, y_0) y (x_1, y_1), la interpolación lineal es la línea recta entre estos puntos. Para un valor x en el intervalo (x_0, x_1) , el valor y a lo largo de la recta viene dado por la ecuación anterior.

Si se utiliza un periodo de muestreo suficientemente pequeño, se puede generar un gran número de mediciones para eventos que ocurren en periodos cortos de tiempo, como los terremotos. Esto es aprovechado por el método de interpolación lineal, ya que cuanto mayor sea el número de muestras disponibles, mayor será la precisión del método para recrear la señal con la que se trabaja. Por este motivo, se seleccionó este método, además de que es un algoritmo relativamente fácil de aplicar y, por tanto, ayuda a agilizar el análisis de daños de la estructura.

La precisión de los datos obtenidos con mediciones realizadas con teléfonos es comparable a los datos obtenidos con acelerómetros, esto se ha demostrado en los trabajos citados que hacen uso de teléfonos inteligentes, además de que los datos obtenidos con teléfonos se han comparado con los obtenidos de los sensores que inicialmente tenía la estación de pruebas. Asimismo, el experimento que se muestra más adelante fue ejecutado anteriormente, pero con una metodología que utiliza sensores convencionales, con la metodología propuesta se obtuvieron los mismos resultados que en aquel experimento.

Además, a las mediciones se les aplico un filtro de paso bajo para mejorar la calidad de las mediciones. El filtro se definió mediante la siguiente ecuación diferencial discreta:

$$x_0(kT) = e^{-\omega_0 T} x_0((k-1)T) + \left(1 - e^{-\omega_0 T}\right) x_i(kT) \tag{5.2}$$

donde la variable x representa la señal a la que se aplica el filtro. Para (5.2), x_i representa el valor de la señal sin aplicar el filtro, x_o representa el valor de la señal cuando se aplica el filtro, ω_0 representa la frecuencia de corte del filtro, y T representa el número total de iteraciones k en el momento de aplicar el filtro.

Se implementó el filtro discreto a la señal reconstruida porque esta señal puede ser tratada como una función definida en tiempo discreto, que tiene un valor en cada instante de tiempo que viene definido por el periodo de muestreo que se utiliza en las mediciones. Tras la reconstrucción y el filtrado de las señales obtenidas a partir de las mediciones del celular, estas nuevas señales se someten a un método de reducción de señales conocido como análisis de componentes principales. Este método se usa para reducir las dimensiones de grandes conjuntos de datos, el método convierte grandes grupos de datos en grupos más pequeños que conservan la información principal de los grupos originales.

Para la aplicación del método de componentes principales, las mediciones se organizan primero en una matriz conocida como matriz de observación:

$$X = \begin{bmatrix} x_{11} x_{12} \cdots x_{1m} \\ x_{21} x_{22} \cdots x_{2m} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ x_{n1} x_{n2} \cdots x_{nm} \end{bmatrix} \tag{5.3}$$

con $X \in \mathbb{R}^{n \times m}$, donde n indica el número de variables medidas y m indica el número de medidas. Los componentes principales en el método son una combinación lineal de las variables originales donde cada z_i no está correlacionado con otro z_i, como:

$$\begin{aligned} z_1 &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m \\ z_2 &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m \\ &\vdots \\ z_p &= a_{p1}x_1 + a_{p2}x_2 + \cdots + a_{pm}x_m \end{aligned} \tag{5.4}$$

donde a_{ij} son los coeficientes a determinar al aplicar la metodología. Cada z_i se determina por la combinación lineal que tiene la máxima varianza de todas las combinaciones lineales posibles para $x_1,\ldots,x_m$. Además, si la varianza del componente principal i-ésimo se define como $\text{var}(z_i) = \lambda_i$,

entonces se cumple que:

$$\lambda_1 > \lambda_2 > \cdots > \lambda_p$$

donde $p \leq m$. Se puede demostrar que las varianzas de los componentes principales corresponden a los valores propios de la matriz de covarianza de las variables originales:

$$\Lambda = U \Sigma U^T \tag{5.5}$$

siendo $\Lambda \in \mathbb{R}^{m \times m}$, $U \in \mathbb{R}^{n \times m}$ y $UU^T = I$ cuyas columnas son los vectores propios que definen los componentes principales, Σ es una matriz diagonal formada por los valores propios en orden descendente de la matriz Λ. Los coeficientes de los componentes principales corresponden a los vectores propios de (5.5), que están definidos por una transformación lineal ortonormal aplicada a (5.3) definida por:

$$Z = U^T X$$

El número de componentes principales puede determinarse por la contribución de la varianza acumulada por las k primeras componentes principales y se calcula como sigue:

$$\frac{\Sigma_j^k = 1 \lambda_j}{\Sigma_j^m = 1 \lambda_j} 100\%$$

el número de componentes principales utilizados corresponde a los que acumulan una contribución de entre el 70% y el 90%. Los datos originales X pueden reconstruirse de la siguiente forma:

$$X = Z + \Theta$$

donde Θ es la matriz residual correspondiente a la información no retenida por los componentes principales y que probablemente representa el ruido de la información.

El funcionamiento del sistema de monitoreo de salud estructural propuesto en este trabajo emplea las metodologías mencionadas anteriormente, basándose en los datos de las mediciones de los teléfonos. La información resultante se utiliza para entrenar la Red Recurrente LSTM Difusa, con la cual se determina la presencia de daños en el edificio en cuestión, según un análisis del

comportamiento de la red. Se usan los datos extraídos del edificio en dos situaciones: sin presencia de daños y con presencia de daños, para ambos casos tras un movimiento sísmico. La red difusa será entrenada con el historial de datos del edificio, cuando se quiera analizar un determinado evento la red hará uso de los datos de ese evento en su proceso de generalización, y en función del error de generalización de la red difusa determinará la presencia de daño estructural.

Además, se comprobó la eficacia de la red para la tarea asignada en función de si los resultados coinciden con el historial del edificio, lo que determinará su fiabilidad para su uso en el diagnóstico de daños en el edificio de forma regular en el futuro. Esto se detalla en la siguiente sección con los resultados experimentales.

5.2.2. Detección de daño con la Red Recurrente LSTM Difusa

Al comparar los resultados, resultó que la red alimentada durante su proceso de aprendizaje con los datos sin daños (información de cuando el edificio sufre un terremoto pero no tiene daños previos y no está dañado) es la más adecuada para la detección de daños. Esto se debe a que la red, al ser entrenada con dichos datos, es más sensible a los cambios bruscos en el comportamiento del edificio, que se traducen en daños estructurales. Asimismo, aunque se mida cada piso, no es posible decir en qué parte del edificio está el daño con esta propuesta, se sacrificó la precisión en la localización del daño estructural por la rapidez del algoritmo para detectar el daño.

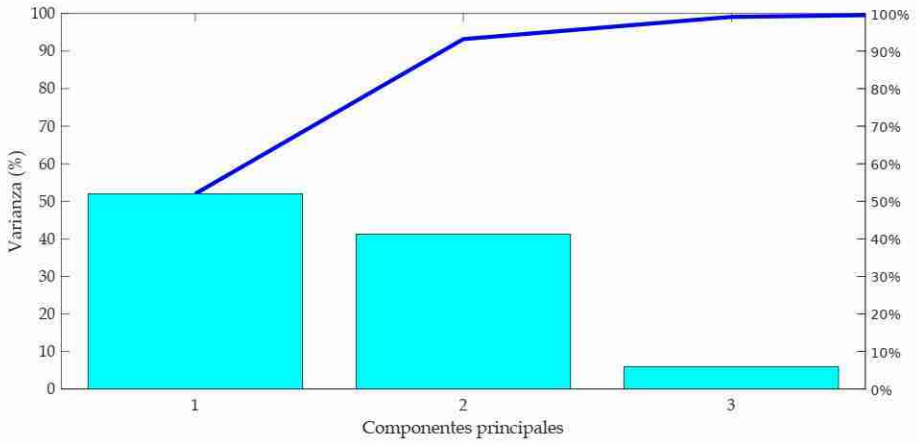


Figura 5.2. Porcentaje de varianza de cada componente principal de los datos medidos.

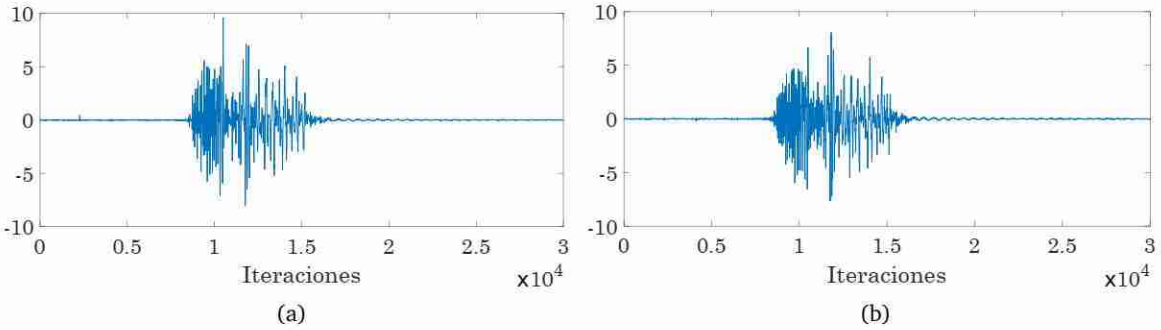


Figura 5.3. Datos medidos en función de los dos componentes principales de la información analizada: (a) datos en función del primer componente, (b) datos en función del segundo componente.

Entonces, aplicando el método de componentes principales, a los datos de medición que han sido reconstruidos con la interpolación lineal y filtrados con el filtro pasa bajas descritos con anterioridad, la información está definida por los dos primeros elementos de los componentes principales de los cuatro que se tienen. Los dos primeros componentes tienen más del 10% de la varianza, el tercer componente tiene un porcentaje muy bajo y el cuarto componente tiene un 0%, como se puede ver en la Figura 5.2. En las Figuras 5.3 (a) y (b) se muestran los datos de medición en función de los dos primeros componentes principales.

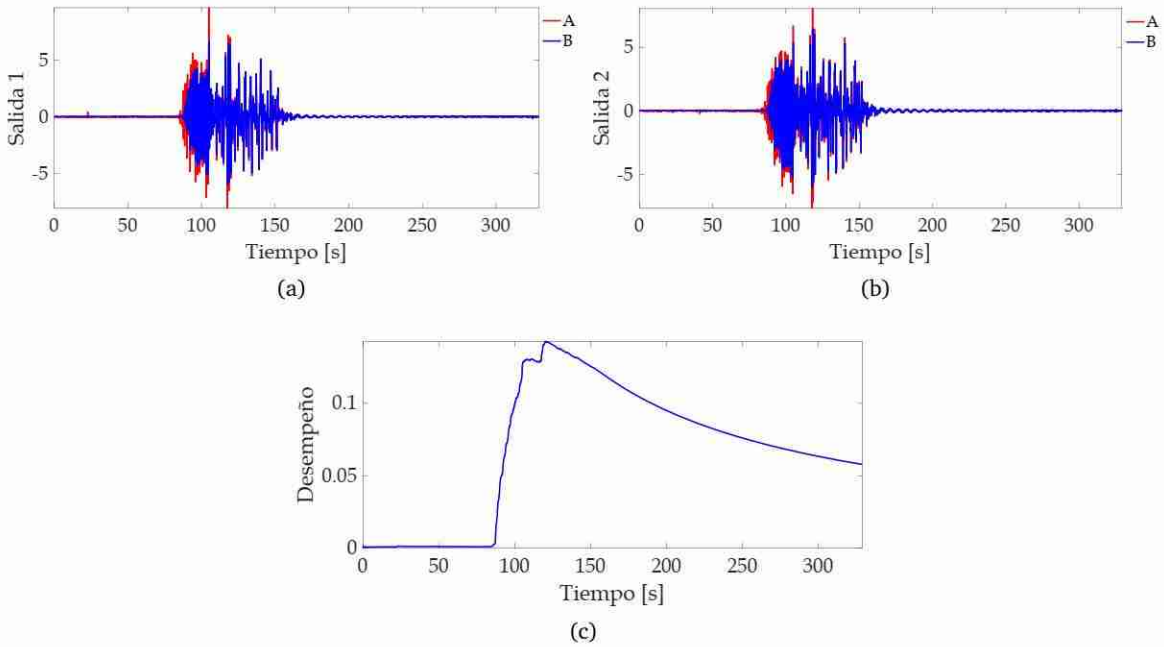


Figura 5.4. Desempeño de la red difusa usada para monitoreo de salud estructural durante su proceso de aprendizaje: (a) medición en función del primer componente principal, “A” datos introducidos en la red y “B” datos generados por la red; (b) medición en función del segundo componente principal, “A” datos introducidos en la red y “B” datos generados por la red; (c) error de la red durante su proceso de aprendizaje.

Con la nueva información se entrenó la red difusa, definiendo su entrada como $U_r = [u_{r1}(k-1), u_{r2}(k-1)]^T$. Para el proceso de entrenamiento, se consideraron los primeros 120s del proceso (o 12,001 iteraciones de datos de medición), los datos restantes se utilizaron para la prueba de la red. El rendimiento de la red durante su proceso de entrenamiento se muestra en la Figura 5.4, a medida que el proceso avanza, la red emula el proceso del que aprende. El error mostrado en la Figura 5.4 (c) corresponde al error de seguimiento de la red e indica lo bien que aprendió la red, en este caso el proceso se realizó correctamente, ya que el error tiende a cero a medida que el proceso continúa.

Para poner en perspectiva los resultados obtenidos con la red difusa, la Figura 5.5 muestra los resultados obtenidos con una otra red que también se basa en la modificación de una red LSTM, en condiciones similares. Esta red de compuertas fue propuesta por Gonzalez y Yu [160] y es una versión corta de una red LSTM. Si se comparan los resultados generados por ambas redes, Figuras 5.4 y 5.5, así como la Tabla 5.1, se aprecia que la red de compuertas es menos eficiente en este tipo de tareas que la red difusa. Dentro de la tabla, las iteraciones representan los siguientes eventos: 8,001 antes del terremoto, 12,001 final del entrenamiento, 16,001 después del terremoto y 33,001 final del proceso.

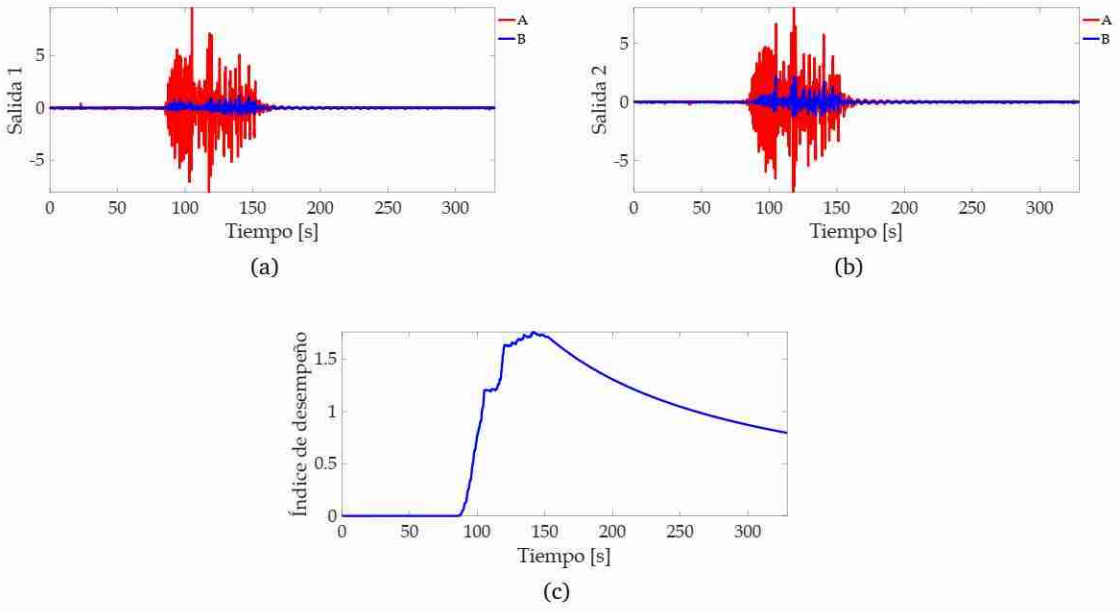


Figura 5.5. Desempeño de la red de compuertas usada para monitoreo de salud estructural durante su proceso de aprendizaje: (a) medición en función del primer componente principal, "A" datos introducidos en la red y "B" datos generados por la red; (b) medición en función del segundo componente principal, "A" datos introducidos en la red y "B" datos generados por la red; (c) error de la red durante su proceso de aprendizaje.

Tabla 5.1. Índices de desempeño de la red de compuertas y de la red difusa, usadas para monitoreo de salud estructural, durante los eventos destacados de su proceso de entrenamiento

Iteración	Red de compuertas	Red difusa
8,001	97.08×10^{-5}	97.57×10^{-5}
12,001	14.23×10^{-2}	16.37×10^{-1}
16,001	11.88×10^{-2}	16.33×10^{-1}
33,001	57.81×10^{-3}	79.55×10^{-2}

Siguiendo con la metodología de la red difusa, en el proceso de generalización de la red, Figura 5.6, se utilizaron datos del edificio con daños por movimiento sísmico para alimentar a la red, dejando fijos los parámetros que la red difusa aprendió anteriormente. Se puede observar como la red es sensible al cambio brusco de los datos, ya que estos datos, comparados con los anteriores, tienen una dinámica diferente y más brusca debido a la aparición de daños en la estructura del edificio. Esta es la parte que representa la detección de daño, lo que da lugar a un análisis rápido tras el procesamiento de los datos provenientes de las mediciones obtenidas de los teléfonos.

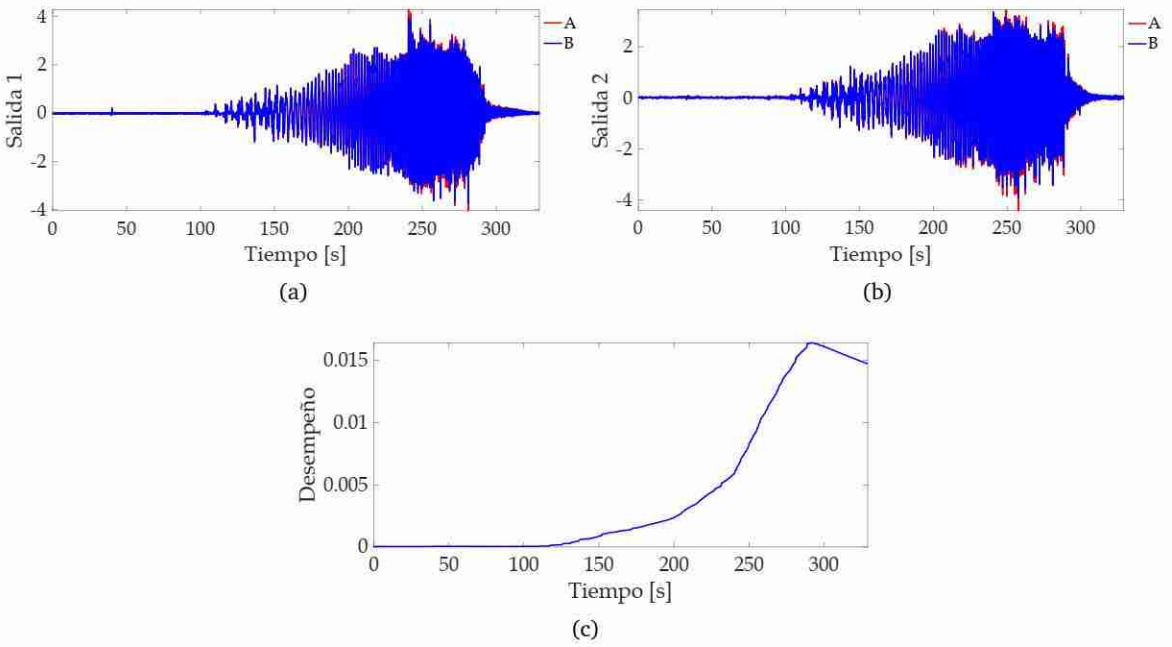


Figura 5.6. Desempeño de la red difusa usada para monitoreo de salud estructural durante su proceso de generalización: (a) medición en función del primer componente principal, “A” datos introducidos en la red y “B” datos generados por la red; (b) medición en función del segundo componente principal, “A” datos introducidos en la red y “B” datos generados por la red; (c) error de la red durante su proceso de aprendizaje.

Tabla 5.2. Índices de desempeño de la red difusa, usada para monitoreo de salud estructural, durante los eventos destacados de su proceso de generalización.

Iteración	Índice de desempeño
10,001	4.794×10^{-5}
20,001	23.68×10^{-4}
25,001	82.90×10^{-4}
31,001	15.64×10^{-3}
33,001	14.74×10^{-3}

Tomando como referencia la Figura 5.6 (c), se aprecia mejor la sensibilidad de la red a los datos con daños. Al principio del gráfico, cuando no hay daños estructurales, el error se mantiene cercano a cero, pero cuando se generan daños en la estructura, el error aumenta. En la Tabla 5.2, las iteraciones representan: 10,001 antes del terremoto, 20,001 en algún momento del terremoto, 25,001 en otro momento del terremoto, 31,001 final del terremoto y 33,001 final del proceso. Este cambio en el error es el que indica la presencia de daños en el edificio.

Finalmente, los daños detectados en el experimento indican que la estructura del edificio, en su estado actual, no es segura para que las personas vuelvan a sus actividades. Esto se sabe por qué en la estación de prueba para emular este tipo de daño, fue necesario aflojar parte de los tornillos que mantienen fijas las bases de los pisos en esta estructura. De esta forma, se comprobó que los resultados de la red difusa coinciden con la realidad de la situación en la que se encuentra la estación de pruebas. Este sistema de monitoreo estructural puede ampliarse a estructuras más grandes, es decir, con un mayor número de pisos, ya que la metodología análisis de componentes principales permite reducir los datos obtenidos de las mediciones sin perder la información que aportan, además, la red difusa puede adaptarse fácilmente para trabajar con más datos de entrada.

5.3. Conclusiones del capítulo

La Red Recurrente LSTM Difusa se utilizó en la detección de daño en estructuras físicas, aplicada en una estación de pruebas que ya había sido empleada en otros trabajos para probar algoritmos diseñados con el mismo fin, obteniendo buenos resultados. Se ideó una metodología, que involucra el uso de la red propuesta y mediciones provenientes de los sensores de teléfonos celulares colocados dentro de la estructura a analizar, para detectar el daño estructural debido a sismos.

Capítulo 6

Red Recurrente LSTM Difusa de tipo 2

Dadas las capacidades de la Red Recurrente LSTM Difusa, mostradas en los capítulos anteriores, en este capítulo se habla de una mejora a esta red, la cual involucra un cambio en los antecedentes de las reglas difusas del algoritmo. Asimismo, se muestran comparaciones entre el algoritmo original y el modificado en la identificación y control de sistemas no lineales.

6.1. Desarrollo de la Red Recurrente LSTM Difusa de tipo 2

Otra forma de minimizar el error de modelado y hacer más eficiente el uso de la Red Recurrente LSTM Difusa, para la identificación y el control de sistemas, es cambiar este sistema difuso de tipo 1 a un sistema de tipo 2. En la teoría de sistemas difusos, los sistemas difusos de tipo 2 se consideran una extensión de los sistemas difusos de tipo 1, esto debido a que el valor de pertenencia de un conjunto difuso de tipo 2 es un número difuso de tipo 1. Las funciones de pertenencia de tipo 1 están bien definas, mientras que las funciones de pertenencia de tipo 2 son difusas; hay infinitas funciones de pertenencia de tipo 1 en la huella de incertidumbre característica de una función de pertenencia de tipo 2.

El empleo de funciones de pertenencia de tipo 2 en reglas difusas, dada su naturaleza, ha probado ser mejor en el manejo de datos con ruido e incertidumbres. En la Figura 6.1 se muestra una función de pertenencia de tipo 2, esta función corresponde a una función Gaussiana con desviación estándar desconocida, la cual es acotada superiormernte e inferiormente por funciones de pertenencia de tipo 1. El área acotada por las funciones de pertenencia de tipo 1 corresponde

a la huella de incertidumbre de la función de pertenencia de tipo 2, dentro de esta área las variables que se fusifican y están definidas en el universo de discurso de la función de pertenencia de tipo 2 toman su valor de pertenencia.

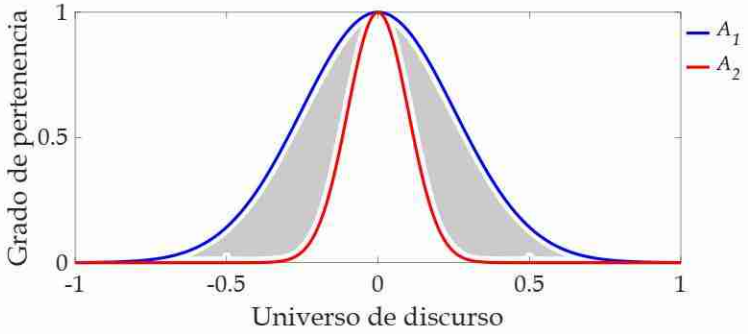


Figura 6.1. Función de pertenecía de un sistema difuso de tipo 2: función de tipo Gaussiana acotada superiormente por "A_1" e inferiormente por "A_2". La zona gris que se distingue en la figura corresponde a la huella de incertidumbre característica de la función de pertenencia, en esta zona es donde las variables definida en el universo de discurso toman su valor de pertenencia.

Entonces, las funciones de pertenencia de tipo 2 son más completas, matemáticamente hablando, que las funciones de pertenencia de tipo 1, de ahí la robustez en los resultados obtenidos con sistemas difusos de tipo 2. En la sección 2.1.4 de este trabajo se habla de algunos trabajos en los que se emplean sistemas difusos de tipo 2, además en el trabajo de Tavoosi et al. [161] se hace un recuento de trabajos destacados acerca de sistemas difusos de tipo 2. En este trabajo se mencionan ventajas y desventajas de los sistemas difusos de tipo 2, aplicaciones notorias en las que se han usado este tipo de sistemas difusos y como aplicar estos sistemas difusos, principalmente en la identificación de sistemas.

La modificación de la Red Recurrente LSTM Difusa, para convertirla en un sistema difuso 2, se enfoca en el cambio de las funciones de pertenencia de tipo 1, definidas en el Capítulo 3, por funciones de pertenencia de tipo 2. Las reglas difusas, de tanto los sistemas difusos de tipo 1 y de tipo 2, tienen la misma sintaxis, por lo que (3.19) será la misma para las Redes Recurrentes LSTM Difusas de tipo 1 y de tipo 2.

Las ecuaciones de los antecedentes de la red difusa de tipo 1 se alteran de la siguiente manera: comenzando por las funciones de pertenencia, se hace uso de funciones Gaussianas como la de la Figura 6.1, organizándolas de forma similar a la mostrada en la Figura 3.3, en este caso cada función esta acotada superior e inferiormente por una función Gaussiana, respectivamente,

las cuales ajustan su ancho (desviación estándar) y la localización de su centro compartido en conformidad con el algoritmo de entrenamiento al que se someterán. Por tanto, de (3.15) para las funciones que delimitan superiormente, a la función de pertenencia de tipo 2, se tiene:

$$\overline{Z}_1=\overline{\mu}(k)_{U(k),A_1},\quad \overline{Z}_2=\overline{\mu}(k)_{U(k),A_2},\quad \&\quad \overline{Z}_3=\overline{\mu}(k)_{U(k),A_3} \tag{6.1}$$

y para las funciones que delimitan inferiormente:

$$\underline{Z}_1=\underline{\mu}(k)_{U(k),A_1},\quad \underline{Z}_2=\underline{\mu}(k)_{U(k),A_2},\quad \&\quad \underline{Z}_3=\underline{\mu}(k)_{U(k),A_3} \tag{6.2}$$

además, en vez de (3.16)-(3.18), se generan las siguientes ecuaciones:

$$\overline{Z}_1=\exp\left((U(k)-\varsigma_1)^2\otimes\left(-\frac{1}{2}\overline{\Upsilon}_1\right)\right) \tag{6.3}$$

$$\overline{Z}_2=\exp\left((U(k)-\varsigma_2)^2\otimes\left(-\frac{1}{2}\overline{\Upsilon}_2\right)\right) \tag{6.4}$$

$$\overline{Z}_3=\exp\left((U(k)-\varsigma_3)^2\otimes\left(-\frac{1}{2}\overline{\Upsilon}_3\right)\right) \tag{6.5}$$

$$\underline{Z}_1=\exp\left((U(k)-\varsigma_1)^2\otimes\left(-\frac{1}{2}\underline{\Upsilon}_1\right)\right) \tag{6.6}$$

$$\underline{Z}_2=\exp\left((U(k)-\varsigma_2)^2\otimes\left(-\frac{1}{2}\underline{\Upsilon}_2\right)\right) \tag{6.7}$$

$$\underline{Z}_3=\exp\left((U(k)-\varsigma_3)^2\otimes\left(-\frac{1}{2}\underline{\Upsilon}_3\right)\right) \tag{6.8}$$

en donde: ς_1, ς_2 y ς_3 son los vectores de los centros de sus respectivas funciones de pertenencia; $\overline{\Upsilon}_1$, $\overline{\Upsilon}_2$ y $\overline{\Upsilon}_3$ son los vectores de los anchos de las funciones que acotan superiormente a sus respectivas funciones de pertenencia; $\underline{\Upsilon}_1$, $\underline{\Upsilon}_2$ y $\underline{\Upsilon}_3$ son los vectores de los anchos de las funciones que acotan inferiormente a sus respectivas funciones de pertenencia.

Las reglas difusas tienen la misma sintaxis en el caso de sistemas difusos de tipo 1, así como de tipo 2, por lo que el total de reglas se calcula como anteriormente se hizo en la Sección 3.2, siendo determinado por la expresión $p=l(3^m)$. Asimismo, las transformaciones que se aplicaron para obtener el vector que representa la contribución de los antecedentes, (3.20)-(3.23), pueden aplicarse de manera similar a las funciones que acotan a la función de pertenencia de tipo 2; se obtendrán $\overline{Z}_4$, $\overline{Z}_5$, $\overline{Z}_6$ y $\overline{Z}$ para las funciones que acotan superirormente y $\underline{Z}_4$, $\underline{Z}_5$, $\underline{Z}_6$ y $\underline{Z}$ para las

funciones que acotan inferiormente.

La parte de las consecuencias de las reglas difusas se queda sin modificar, por lo que para ambos tipos de sistemas difusos se emplea a la red LSTM, formada con (3.7)-(3.12), para definir la defusificación. Por último, las salidas estimadas por la red difusa de tipo 2 son calculadas por (6.9), la cual efectuá de la misma manera la operación de defusificación que se mencionó para (3.24), que se refiere a la media ponderada de cualquier sistema TSK pero en forma vectorial, pero considerando las funciones que acotan a las funciones de pertenencia de tipo 2.

$$Y_r(k)=\left[\overline{Z}'(k)+(1-\beta)\underline{Z}'(k)\right]H(k) \tag{6.9}$$

en esta nueva ecuación, el término que contiene al parámetro β es un parámetro de diseño del sistema difuso de tipo 2, el cual pondera el compartimiento de los niveles de activación inferior y superior de cada regla activada. Con la descripción anterior queda definida la Red Recurrente LSTM Difusa de tipo 2, en donde el principal cambio con respecto a la red difusa de tipo 1 es el cálculo de los antecedentes de las reglas difusas, lo que aumenta la robustez en el procesamiento de información de red, a la vez que aumenta la complejidad del algoritmo original.

El entrenamiento de la red difusa de tipo 2 es similar al de la red de tipo 1, por lo que se puede aplicar lo establecido en la Subsección 3.2.1, solo que se tiene que contemplar en el procedimiento a las funciones que acotan a las funciones de pertenencia. Para ejemplificar esto, si se define una red con las siguientes dimensiones: $m=1$, $l=1$ y $\kappa>1$ (donde κ representa el número de conjuntos difusos para los antecedentes), el ajuste de cada elemento de la matriz W^i se representa de la siguiente manera:

$$\Delta w_p^i=\frac{\partial \xi(k)}{\partial e(k)}\frac{\partial e(k)}{\partial y_r(k)}\frac{\partial y_r(k)}{\partial h_p(k)}\frac{\partial h_p(k)}{\partial \psi\left(c_p(k)\right)}\frac{\partial \psi\left(c_p(k)\right)}{\partial c_p(k)}\frac{\partial c_p(k)}{\partial i_p(k)}\frac{\partial i_p(k)}{\partial w_p^i(k)}$$

y en forma vectorial:

$$\Delta W^i(k)=\left(\dot{\sigma}\left(W^iU_r(k)+V^iH(k-1)\right)\otimes S(k)\otimes\dot{\psi}(C(k))\otimes\left(\overline{Z}'(k)+(1-\beta)\underline{Z}'(k)\right)e(k)\otimes O(k)\right)U_r(k)$$

de la misma manera, las condiciones de estabilidad obtenidas en la Subsección 3.2.2 se mantienen,

ya que dependen directamente de cada arreglo de parámetros de la red.

6.2. Comparación entre Redes Recurrentes LSTM Difusas de tipo 1 y tipo 2

Para observar las diferencias entre las Redes Recurrentes LSTM Difusas de tipo 1 y tipo 2 se comparó el funcionamiento de dichas redes en la identificación y control de sistemas. Se desarrollan en las siguientes subsecciones dos ejercicios diferentes, con la finalidad de poder observar las mejoras que ofrece la modificación al algoritmo original de la Red Recurrente LSTM Difusa.

6.2.1. Identificación de un sistema no lineal

El primer ejercicio consiste en la generación de un modelo para un sistema no lineal, que se define como:

$$y(k+1)=0.72y(k)+0.025y(k-1)u(k-1)+0.01u^2(k-2)+0.2u(k-3) \tag{6.10}$$

con:

$$u(k)=\begin{cases} \text{sen}\left(\frac{\pi}{25}k\right), & k\leq 250 \\ 1, & 250<k\leq 500 \\ -1, & 500<k\leq 750 \\ 0.3\text{sen}\left(\frac{\pi}{25}k\right)+0.1\text{sen}\left(\frac{\pi}{32}k\right)+0.6\text{sen}\left(\frac{\pi}{10}k\right), & 750<k \end{cases}$$

donde $y(0)=0$, $u(0)=0$, además se considera un tiempo de muestreo de $T=1$s.

Así, (6.10) se resolvió por un periodo de tiempo de 1,200s, muestreando al sistema se creó el vector $y(k)$ con $k=1,...1,201$. Se tomaron los valores de $y(k)$ para definir el vector de entrada de la red $U_r(k)=[y(k-4),y(k-8)]^T$, el cual se utilizó para hacer la estimación $\hat{y}(k)$ del sistema original. Se emplearon las primeras 601 iteraciones para entrenar a las redes, mientras que el resto de los datos se usaron para probar la generalización de los algoritmos.

Asimismo, se establecieron $p=9$ reglas difusas para las redes ($m=2$, $\kappa=3$, $l=1$). Aquí, el error de modelado $E(k)$ al final de cada fase se define como en (4.1) y representa el rendimiento de

los algoritmos, un valor bajo indica un mejor rendimiento. Se utilizo a (4.1) para el entrenamiento de las redes porque es el error que se quiere minimizar.

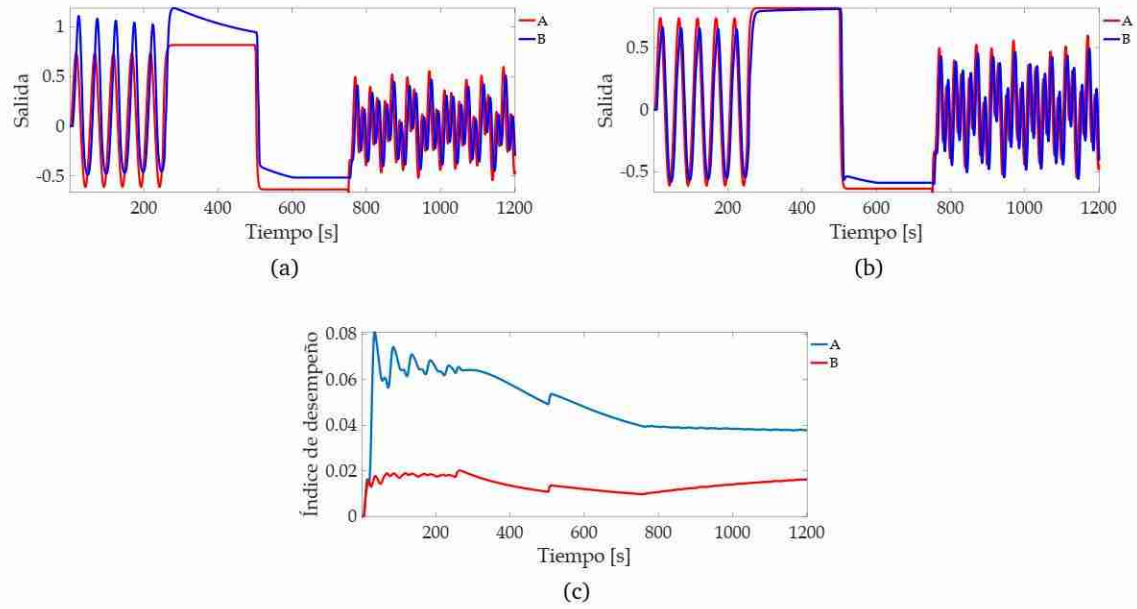


Figura 6.2. Comparación entre las Redes Recurrentes LSTM Difusas de tipo 1 y tipo 2 en la Subsección 6.2.1 : (a) identificación del sistema con la red de tipo 1, el sistema corresponde a "A" y la red a "B"; (b) identificación del sistema con la red de tipo 2, el sistema corresponde a "A" y la red a "B"; (c) índices de desempeño de las redes, "A" corresponde a la Red Recurrente LSTM Difusa de tipo 1 y "B" a la Red Recurrente LSTM Difusa de tipo 2.

Tabla 6.1. Índices de desempeño de las Redes Recurrentes LSTM Difusas de tipo 1 y de tipo 2 en la Subsección 6.2.1.

Red difusa	Entrenamiento	Generalización
Red de tipo 1	4.80×10^{-2}	3.78×10^{-2}
Red de tipo 2	1.19×10^{-2}	1.62×10^{-2}

Los resultados del entrenamiento y la generalización se muestran en la Tabla 6.1 y en la Figura 6.2, la figura tiene tres partes: (a) muestra la salida del sistema y la salida de la red de tipo 1, (b) muestra la salida del sistema y la salida de la red de tipo 2, y (c) muestra una comparación de rendimiento entre las redes.

Se puede observar que ambas redes tienen un comportamiento similar, sin embargo, la red de tipo 2 aprendió la dinámica del sistema más rápido en comparación con la red de tipo 1, por

lo que la primera tiene un menor error de modelado o mejor rendimiento durante todo el proceso.

6.2.2. Control de un sistema no lineal

Por medio de un esquema de control, similar al presentado en la Subsección 4.2.2, exceptuando el uso del control auxiliar (el Controlador 2 en la Figura 4.2), en este ejercicio se requiere controlar al siguiente sistema:

$$y(k+1)=\frac{a(k)y(k)}{1+y^2(k)}+\theta(k)+u(k) \tag{6.11}$$

siendo $y(0)=0, u(k)=r(k)-\hat{y}(k)$ y:

$$a(k)=\begin{cases}1, & kT\leq 250\\ \text{sen}\left(\frac{kT}{10}\right), & 250<kT\end{cases}$$

$$\theta(k)=\begin{cases}\text{valor aleatorio en } [-0.1,0.1], & 320\leq kT\leq 380\\ 0, & \text{en cualquier otro caso}\end{cases}$$

este sistema es una modificación de (4.10), en donde $r(k)$ está definida acorde a (4.11), $a(k)$ es una variable del sistema en cuestión que cambia en determinado tiempo, y $\theta(k)$ representa la presencia de ruido en la señal $y(k)$ del sistema.

Asimismo, (6.11) está diseñada de tal forma que se puedan apreciar de forma clara las capacidades de adaptabilidad y robustez de las redes. Así, la tarea consiste en que el modelo, $\hat{y}(k)$, generado por las redes compense la variable del sistema, $y(k)$, para que la respuesta de ésta se asemeje a la de la referencia, $r(k)$, lo más cercano posible.

Se estableció una duración de 500s para todo el proceso, tomando muestras de la señal del sistema con un periodo de muestreo $T = 0.1$s, obteniendo de esta forma un total de $k = 1,\ldots,5,001$ iteraciones del proceso, el vector de entradas de las redes se definió como $U_r(k) = [E_C(k-1),E_C(k-2)]^{'}$, siendo $E_C(k)$ como en (4.2). Además, para ambas redes, se establecieron $p=9$ reglas difusas, siendo $m=2$, $\kappa=3$ y $l=1$, así como parámetros iniciales menores a 0.1 para los pesos sinápticos.

El desempeño de las Redes Recurrentes LSTM Difusas de tipo 1 y tipo 2 se midió con el error mostrado en (4.2), de la forma de (3.27), creando un índice de desempeño del actuar de las redes en el control del sistema. Un valor bajo del índice de desempeño indica que las redes cumplieron satisfactoriamente la tarea asignada, por lo que entre más pequeño el valor mejor. Asimismo, las redes se entrenaron con el error de la forma (4.2), ya que este es el error que se quiere minimizar.

Se trabajo con las redes de la siguiente manera: el entrenamiento de los algoritmos se realizo durante 100s, equivalente a 1,001 iteraciones. La generalización de las redes se efectuó inmediatamente después durante los 400s restantes del proceso, es decir, este proceso se desarrollo durante unas 4,000 iteraciones más. Los resultados de estos procesos se muestran en la Figura 6.3 y en la Tabla 6.2, en donde se muestra las diferencia durante los procesos de estos dos algoritmos.

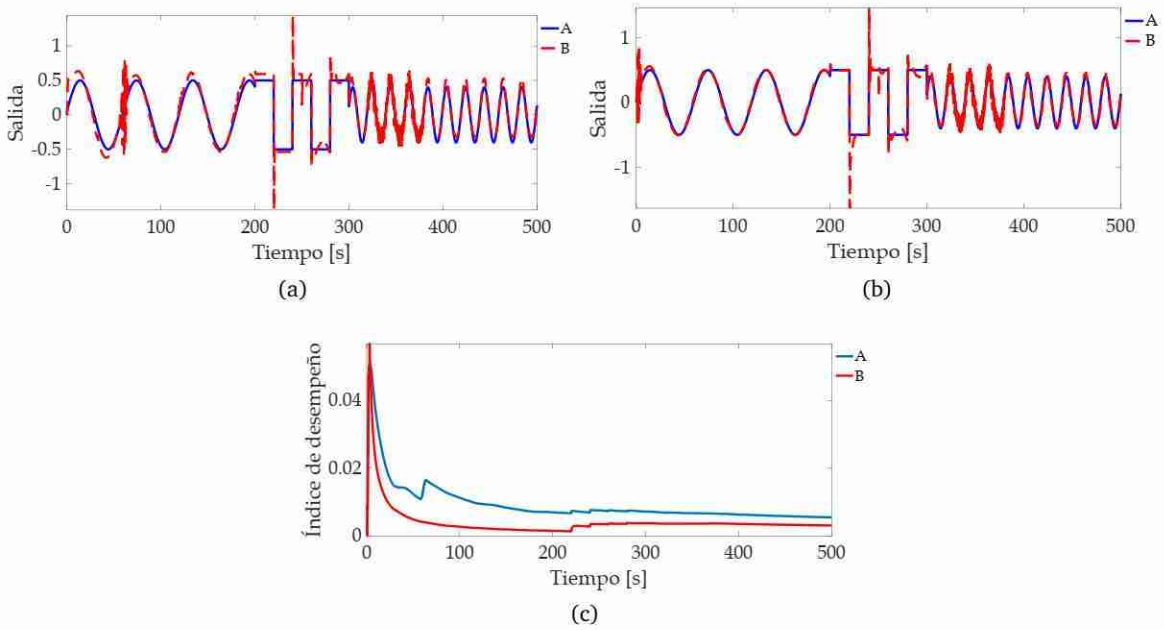


Figura 6.3. Comparación entre las Redes Recurrentes LSTM Difusas de tipo 1 y tipo 2 en la Subsección 6.2.2: (a) seguimiento de la referencia por parte de la planta con el control obtenido de la red de tipo 1, la referencia corresponde a "A" y la planta a "B"; (b) seguimiento de la referencia por parte de la planta con el control obtenido de la red de tipo 2, la referencia corresponde a "A" y la planta a "B"; (c) índices de desempeño de las redes, "A" corresponde a la Red Recurrente LSTM Difusa de tipo 1 y "B" a la Red Recurrente LSTM Difusa de tipo 2.

Los resultados de este ejercicio que se muestran en la Figura 6.3 se describen de la siguiente manera: en (a) se muestra el seguimiento de la planta con respecto de la referencia, con el control generado por la Red Recurrente LSTM Difusa de tipo 1; en (b) se muestra el seguimiento de la

planta con respecto de la referencia, con el control generado por la Red Recurrente LSTM Difusa de tipo 2; y en (c) se muestran los índices de desempeño de ambos controles, que se mantienen en un valor cercano a cero, indicando que el control en cada caso fue efectivo.

En la Tabla 6.2, se muestra el valor del índice de desempeño de las Redes Recurrentes LSTM Difusas de tipo 1 y tipo 2 (red de tipo 1 y red de tipo 2, respectivamente) como único control del sistema (6.11); se muestran los valores de los índices de desempeño en diferentes situaciones, relevantes durante el periodo toral en que se realizo el control del sistema, concordando con lo que se muestra en cada subfigura de la Figura 6.3, el valor de los índices se mantiene cercano a cero indicando que el control fue exitoso, en el sentido que el sistema logro seguir a la referencia con cierto grado de error en cada caso.

Tabla 6.2. Índices de desempeño de las Redes Recurrentes LSTM Difusas de tipo 1 y de tipo 2 durante los eventos destacados en la Subsección 6.2.2.

Evento	Iteración	Índice de desempeño	
		Red de tipo 1	Red de tipo 2
Fin del entrenamiento	1,001	1.11×10^{-2}	0.27×10^{-2}
Cambio en $a(k)$	2,501	0.75×10^{-2}	0.34×10^{-2}
Inicio de $\theta(k)$	3,201	0.68×10^{-2}	0.36×10^{-2}
Final de $\theta(k)$	3,801	0.65×10^{-2}	0.36×10^{-2}
Final del proceso	5,001	0.54×10^{-2}	0.31×10^{-2}

Al revisar los resultados del ejercicio propuesto en esta sección, se nota que ambas redes tuvieron un comportamiento similar a lo largo del ejercicio, pero la Red Recurrente LSTM Difusa de tipo 2 entregó resultados más robustos. Esto, debido a que con el control obtenido por la red de tipo 2 el sistema tuvo el mejor seguimiento de trayectoria de la referencia. Aunado al hecho de que ambas redes trabajaron bajo las mismas condiciones, tanto en condiciones iniciales, número de reglas y tiempo de entrenamiento.

Dada la flexibilidad del algoritmo propuesto en esta tesis fue posible el cambio de éste aun sistema de tipo 2, obteniendo mejores resultados en cuando a la reducción del error de modelado presente en la obtención de cualquier modelo, en comparación al sistema difuso inicial. Esto se comprobó en el control de sistemas, ya que en la Sección 4.2 los controladores obtenidos con la propuesta original requieren de un control auxiliar para generar un control adecuado, con el cual

el error de control se mantienen en valores mínimos. Mientras que en esta sección, para mejorar el modelo de controladores obtenidos con la propuesta, es decir, no hacer uso de controladores auxiliares ajenos a la Red Recurrente LSTM Difusa, se tuvo que aumentar el orden de la parte difusa del algoritmo lo que implica aumentar la complejidad de éste.

Con la tecnología actual de la computación se puede hacer frente al problema de complejidad que presenta el uso de la Red Recurrente LSTM Difusa de tipo 2, ya que conforme avanza el tiempo van surgiendo procesadores capaces de realizar una gran cantidad de operaciones complejas, de manera eficiente, en periodos de tiempo más cortos. Esto se debe de tener en cuenta si se quiere usar la modificación de la Red Recurrente LSTM Difusa, porque es una desventaja considerable de dicha modificación. Para la mayoría de aplicaciones de identificación de sistemas el algoritmo original tendrá un buen desempeño, por lo que la elección del tipo de algoritmo a emplear en determinada situación dependerá de la complejidad de las tareas que se busquen cumplir.

6.3. Conclusiones del capítulo

Se mostró una modificación al algoritmo original de la Red Recurrente LSTM Difusa, la cual consiste en cambiar la red de un sistema difuso de tipo 1 a un sistema difuso de tipo 2, lo que aumenta la complejidad del algoritmo pero al mismo tiempo mejora la respuesta de la red en el contexto en el que se emplee.

Conclusiones generales

Se cumplieron los objetivos de esta tesis, planteados en el Capítulo 1, por medio del establecimiento de la Red Recurrente LSTM Difusa. Por lo que la hipótesis, guía de esta investigación, resulto ser afirmativa. Esta red se diseñó a partir del Capítulo 2 y se desarrolló en el Capítulo 3.

La red a través de sus secciones procesa datos, con los cuales es alimentada, de forma paralela y luego junta los resultados de cada sección para obtener una respuesta global. Los modelos obtenidos con la red corresponden a un conjunto de reglas causa-efecto, las cuales relacionan los datos de entrada con funciones no lineales. Se incluye un algoritmo de entrenamiento, con condiciones de estabilidad, para ajustar los parámetros de la red; se trata de una variación del algoritmo BPTT que considera una "ventana" de datos de la red, en su iteración actual e iteración posterior, permitiendo de esta manera la operación de la red en tiempo real. Entonces, la implementación de la red, debido a su estructura y entrenamiento, no requiere de una gran capacidad computacional.

El rendimiento de la red, así como de los otros algoritmos mencionados en el trabajo, se midió conforme a la cercanía a cero de su error de modelado (el algoritmo más eficiente es el que genera el error más pequeño). En el Capítulo 3, se mostró la mayor eficacia de la red en la identificación de sistemas no lineales en comparación a otros algoritmos similares. En el Capítulo 4, se hizo el control de sistemas no lineales con la red por medio de tres esquemas de control. Asimismo, en el Capítulo 5, se realizó una metodología de detección de daño estructural que involucra a la red. Con lo anterior, se aprecia que la red es apta en el manejo de sistemas con cambios bruscos de comportamiento.

Por último, en el Capítulo 6, se habló de una modificación al algoritmo de la red para mejorar su rendimiento principalmente en el control de sistemas. Se logró una mejora substancial, pero con la desventaja de que la red aumenta la complejidad de su estructura. Por otra parte, esto da pie a que se siga trabajando en el algoritmo, buscando su simplificación sin reducir su rendimiento.

Trabajo a futuro

Con los resultados favorables, que se mostraron en este trabajo, de la Red Recurrente LSTM Difusa que exhiben las ventajas de la red propuesta sobre otros algoritmos inteligentes, es posible que esta nueva red pueda aplicarse en varias áreas, aparte del monitoreo de salud estructural. Áreas que involucren la simplificación de modelos, la identificación y control de sistemas con ruido persistente, o el procesamiento de grandes cantidades de datos.

Otros trabajos a futuro sobre esta red pueden contemplar la simplificación de la estructura del algoritmo, la modificación del paradigma de su aprendizaje, así como su expansión e incorporación con otras metodologías de la computación inteligente. Lo anterior, en pro de mejorar el rendimiento del algoritmo y que éste pueda ser usado en más aplicaciones.

Bibliografía

[1] L. A. Zadeh, "Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems", *Soft Computing-A fusion of foundations, methodologies and applications*, vol. 2, n.º 1, págs. 23-25, 1998.

[2] L. A. Zadeh, "A critical view of our research in automatic control", *IRE Transactions on Automatic Control*, vol. 7, n.º 3, págs. 74-75, 1962.

[3] L. A. Zadeh, "The evolution of systems analysis and control: a personal perspective", *IEEE Control Systems Magazine*, vol. 16, n.º 3, págs. 95-98, 1996.

[4] L. Magdalena, "What is soft computing? revisiting possible answers", *International Journal of Computational Intelligence Systems*, vol. 3, n.º 2, págs. 148-159, 2010.

[5] F. Amara, K. Agbossou, A. Cardenas, Y. Dubé y S. Kelouwani, "Comparison and simulation of building thermal models for effective energy management", *Smart Grid and renewable energy*, vol. 6, n.º 04, págs. 95-112, 2015.

[6] D. Ibrahim, "An overview of soft computing", *Procedia Computer Science*, vol. 102, págs. 34-38, 2016.

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1994.

[8] F. Musumeci et al., "An overview on application of machine learning techniques in optical networks", *IEEE Communications Surveys & Tutorials*, vol. 21, n.º 2, págs. 1383-1408, 2018.

[9] D. Miljković, "Brief review of self-organizing maps", en *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, 2017, págs. 1061-1066.

[10] M. Rehman y N. Nawi, "Improving the accuracy of gradient descent back propagation algorithm (GDAM) on classification problems", *International Journal on New Computer Architectures and Their Applications*, vol. 1, n.º 4, págs. 838-847, 2011.

[11] M. A. W. Saduf y A Wani, "Comparative study of back propagation learning algorithms for neural networks", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, n.º 12, 2013.

[12] R. Chaudhary, H. Patel y M. Scholar, "A survey on backpropagation algorithm for neural networks", *Int. J. Technol. Res. Eng*, vol. 2, n.º 7, 2015.

[13] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey", *IEEE Transactions on Neural networks*, vol. 6, n.º 5, págs. 1212-1228, 1995.

[14] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach*. GMD-Forschungszentrum Informationstechnik Bonn, 2002, vol. 5.

[15] M. I. Jordan y T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects", *Science*, vol. 349, n.º 6245, págs. 255-260, 2015.

[16] Y. LeCun, Y. Bengio y G. Hinton, "Deep learning", *nature*, vol. 521, n.º 7553, págs. 436-444, 2015.

[17] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning", *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014.

[18] Z. C. Lipton, J. Berkowitz y C. Elkan, "A critical review of recurrent neural networks for sequence learning", *arXiv preprint arXiv:1506.00019*, 2015.

[19] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu y F. E. Alsaadi, "A survey of deep neural network architectures and their applications", *Neurocomputing*, vol. 234, págs. 11-26, 2017.

[20] M. Shafique et al., "An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the IoT era", en *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2018, págs. 827-832.

[21] M. Islam, G. Chen y S. Jin, "An Overview of Neural Network", *American Journal of Neural Networks and Applications*, vol. 5, n.º 1, págs. 7-11, 2019.

[22] M. Z. Alom et al., "A state-of-the-art survey on deep learning theory and architectures", *Electronics*, vol. 8, n.º 3, pág. 292, 2019.

[23] Q. Zhang, L. T. Yang, Z. Chen y P. Li, "A survey on deep learning for big data", *Information Fusion*, vol. 42, págs. 146-157, 2018.

[24] L.-X. Wang, *A Course in Fuzzy Systems and Control*. Prentice Hall International, Inc., 1997.

[25] M. M. Gupta y J Qi, "Theory of T-norms and fuzzy inference methods", *Fuzzy sets and systems*, vol. 40, n.º 3, págs. 431-450, 1991.

[26] A. Tserkovny, "A Fuzzy Logic Based Resolution Principal for Approximate Reasoning", *Journal of Software Engineering and Applications*, vol. 10, n.º 10, pág. 793, 2017.

[27] O. A. M. Ali, A. Y. Ali y B. S. Sumait, "Comparison between the effects of different types of membership functions on fuzzy logic controller performance", *International Journal*, vol. 76, págs. 76-83, 2015.

[28] A. Kaur y A. Kaur, "Comparison of mamdani-type and sugeno-type fuzzy inference systems for air conditioning system", *International journal of soft computing and engineering*, vol. 2, n.º 2, págs. 323-325, 2012.

[29] M. Blej y M. Azizi, "Comparison of Mamdani-type and Sugeno-type fuzzy inference systems for fuzzy real time scheduling", *International Journal of Applied Engineering Research*, vol. 11, n.º 22, págs. 11 071-11 075, 2016.

[30] M.-L. Lee, H.-Y. Chung y F.-M. Yu, "Modeling of hierarchical fuzzy systems", *Fuzzy sets and systems*, vol. 138, n.º 2, págs. 343-361, 2003.

[31] D. Wang, X.-j. Zeng y J Keane, "A survey of hierarchical fuzzy systems", *International journal of computational cognition*, vol. 4, n.º 1, págs. 18-29, 2006.

[32] J.-S. Jang, "ANFIS: adaptive-network-based fuzzy inference system", *IEEE transactions on systems, man, and cybernetics*, vol. 23, n.º 3, págs. 665-685, 1993.

[33] N. Walia, H. Singh y A. Sharma, "ANFIS: Adaptive neuro-fuzzy inference system-a survey", *International Journal of Computer Applications*, vol. 123, n.º 13, 2015.

[34] S.-i. Horikawa, T. Furuhashi e Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm", *IEEE transactions on Neural Networks*, vol. 3, n.º 5, págs. 801-806, 1992.

[35] J.-S. Jang, "Self-learning fuzzy controllers based on temporal backpropagation", *IEEE Transactions on neural networks*, vol. 3, n.º 5, págs. 714-723, 1992.

[36] M. Panella, "A hierarchical procedure for the synthesis of ANFIS networks", *Advances in Fuzzy Systems*, vol. 2012, pág. 20, 2012.

[37] A. Prieto et al., "Neural networks: An overview of early research, current frameworks and new challenges", *Neurocomputing*, vol. 214, págs. 242-268, 2016.

[38] H. Salehinejad, S. Sankar, J. Barfett, E. Colak y S. Valaee, "Recent advances in recurrent neural networks", *arXiv preprint arXiv:1801.01078*, 2017.

[39] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv:1704.04861*, 2017.

[40] R. Wang y X. He, "Face Detection Based on Template Matching and Neural Network", en *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, IEEE, 2019, págs. 547-550.

[41] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng y Z. Zhang, "The application of two-level attention models in deep convolutional neural network for fine-grained image classification", en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, págs. 842-850.

[42] J. Donahue et al., "Long-term recurrent convolutional networks for visual recognition and description", en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, págs. 2625-2634.

[43] B. Shi, X. Bai y C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition", *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, n.º 11, págs. 2298-2304, 2016.

[44] S. Khristoforov y V. Bochkarev, "Estimation of Geomagnetic and Solar Indices by Global Ionospheric Maps With Use of Neural Networks", en *2019 Russian Open Conference on Radio Wave Propagation (RWP)*, IEEE, vol. 1, 2019, págs. 123-126.

[45] J. Anderson, S. Sun, Y. Alkabani, V. Sorger y T. El-Ghazawi, "Photonic Processor for Fully Discretized Neural Networks", en *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, IEEE, vol. 2160, 2019, págs. 25-32.

[46] J. B. Ali, N. Fnaiech, L. Saidi, B. Chebel-Morello y F. Fnaiech, "Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals", *Applied Acoustics*, vol. 89, págs. 16-27, 2015.

[47] S.-C. Lin, S.-F. Su e Y. Huang, "A Time-frequency Signal-based Convolutional Neural Network Algorithm for Fault Diagnosis of Gasoline Engine Fuel Control System", en *2019 International Conference on System Science and Engineering (ICSSE)*, IEEE, 2019, págs. 81-87.

[48] M.-C. Le, S.-F. Su, V.-T. Nguyen, L.-W. Chen y V.-Y. Nguyen, "Adaptive Neural Network Controller-Based Chattering-Free Sliding Mode for 6-Dof Industrial Manipulators", en *2019 International Conference on System Science and Engineering (ICSSE)*, IEEE, 2019, págs. 75-80.

[49] L. L. N. Bao, D. A. Nguyen y V. H. Hai, "Command-Based Autopilot System for Ships Using Neural Network-PID Controller", en *2019 International Conference on System Science and Engineering (ICSSE)*, IEEE, 2019, págs. 586-591.

[50] H. Faris, I. Aljarah y S. Mirjalili, "Training feedforward neural networks using multi-verse optimizer for binary classification problems", *Applied Intelligence*, vol. 45, n.º 2, págs. 322-332, 2016.

[51] O. Bektas, J. A. Jones, S. Sankararaman, I. Roychoudhury y K. Goebel, "A neural network filtering approach for similarity-based remaining useful life estimation", *The International Journal of Advanced Manufacturing Technology*, vol. 101, n.º 1-4, págs. 87-103, 2019.

[52] O. Ogunmolu, X. Gu, S. Jiang y N. Gans, "Nonlinear systems identification using deep dynamic neural networks", *arXiv preprint arXiv:1610.01439*, 2016.

[53] J. Chung, C. Gulcehre, K. Cho e Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", *arXiv preprint arXiv:1412.3555*, 2014.

[54] M. Zaheer, A. Ahmed y A. J. Smola, "Latent LSTM allocation joint clustering and non-linear dynamic modeling of sequential data", en *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, págs. 3967-3976.

[55] F. Nicola, Y. Fujimoto y R. Oboe, "A LSTM Neural Network applied to Mobile Robots Path Planning", en *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, IEEE, 2018, págs. 349-354.

[56] Y. Liu, Y. Zhou y X. Li, "Attitude Estimation of Unmanned Aerial Vehicle Based on LSTM Neural Network", en *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018, págs. 1-6.

[57] L. Li e Y. Jiang, "Biomedical named entity recognition based on the two channels and sentence-level reading control conditioned LSTM-CRF", en *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2017, págs. 380-385.

[58] M. Coto-Jimenez, J. Goddard-Close, L. Di Persia y H. L. Rufiner, "Hybrid Speech Enhancement with Wiener filters and Deep LSTM Denoising Autoencoders", en *2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI)*, IEEE, 2018, págs. 1-8.

[59] A. I. Aviles, S. M. Alsaleh, J. K. Hahn y A. Casals, "Towards retrieving force feedback in robotic-assisted surgery: A supervised neuro-recurrent-vision approach", *IEEE transactions on haptics*, vol. 10, n.º 3, págs. 431-443, 2016.

[60] Y. Wang, K. Velswamy y B. Huang, "A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems", *Processes*, vol. 5, n.º 3, pág. 46, 2017.

[61] Y. Wang, "A new concept using lstm neural networks for dynamic system identification", en *2017 American Control Conference (ACC)*, IEEE, 2017, págs. 5324-5329.

[62] S. Li, W. Li, C. Cook, C. Zhu e Y. Gao, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn", en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, págs. 5457-5466.

[63] H. Sang, C. Yang, F. Liu, J. Yun y G. Jin, "A fuzzy neural network sliding mode controller for vibration suppression in robotically assisted minimally invasive surgery", *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 12, n.º 4, págs. 670-679, 2016.

[64] A. I. Aviles, S. M. Alsaleh, E. Montseny, P. Sobrevilla y A. Casals, "A deep-neuro-fuzzy approach for estimating the interaction forces in robotic surgery", en *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 2016, págs. 1113-1119.

[65] M. Yousefi, M. Yousefi, R. P. M. Ferreira, J. H. Kim y F. S. Fogliatto, "Chaotic genetic algorithm and Adaboost ensemble metamodeling approach for optimum resource planning in emergency departments", *Artificial intelligence in medicine*, vol. 84, págs. 23-33, 2018.

[66] C.-H. Lu, W.-C. Wang, C.-C. Tai y T.-C. Chen, "Design of a heart rate controller for treadmill exercise using a recurrent fuzzy neural network", *Computer methods and programs in biomedicine*, vol. 128, págs. 27-39, 2016.

[67] J.-B. Huang, K.-Y. Young y C.-H. Ko, "Effective control for an upper-body exoskeleton robot using ANFIS", en *2016 International Conference on System Science and Engineering (ICSSE)*, IEEE, 2016, págs. 1-4.

[68] K. Li, H. Su y J. Chu, "Forecasting building energy consumption using neural networks and hybrid neuro-fuzzy system: A comparative study", *Energy and Buildings*, vol. 43, n.º 10, págs. 2893-2899, 2011.

[69] H. M. Sri, P. Rao, P. K. Kammardi, S. S. Shekar, S. Kathavate y K. Gowranga, "A smart adaptive LSTM technique for electrical load forecasting at source", en *2017 2nd IEEE*

International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE, 2017, págs. 1717-1721.

[70] W. Yao, P. Huang y Z. Jia, "Multidimensional LSTM Networks to Predict Wind Speed", en *2018 37th Chinese Control Conference (CCC)*, IEEE, 2018, págs. 7493-7497.

[71] L. Cheng et al., "Ensemble recurrent neural network based probabilistic wind speed forecasting approach", *Energies*, vol. 11, n.º 8, pág. 1958, 2018.

[72] N. Hosseinzadeh y P. Wolfs, "Battery time of discharge setting for maximum effectiveness in a distribution smart grid application", en *2014 International Conference on Renewable Energy Research and Application (ICRERA)*, IEEE, 2014, págs. 535-538.

[73] H. Rashidy, S. Rezeka, A. Saafan y T. Awad, "A hierarchical neuro-fuzzy system for identification of simultaneous faults in hydraulic servovalves", en *Proceedings of the 2003 American Control Conference, 2003.*, IEEE, vol. 5, 2003, págs. 4269-4274.

[74] K. Zhang, B. Jiang y P. Shi, "Fault estimation observer design for discrete-time Takagi–Sugeno fuzzy systems based on piecewise Lyapunov functions", *IEEE Transactions on Fuzzy systems*, vol. 20, n.º 1, págs. 192-200, 2011.

[75] Y. Wei, J. Qiu y H. R. Karimi, "Reliable output feedback control of discrete-time fuzzy affine systems with actuator faults", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, n.º 1, págs. 170-181, 2016.

[76] R. Yera y L. Martinez, "Fuzzy tools in recommender systems: A survey", *International Journal of Computational Intelligence Systems*, vol. 10, n.º 1, págs. 776-803, 2017.

[77] A. Fernandez, V. Lopez, M. J. del Jesus y F. Herrera, "Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges", *Knowledge-Based Systems*, vol. 80, págs. 109-121, 2015.

[78] M. B. Kamal, G. J. Mendis y J. Wei, "Intelligent soft computing-based security control for energy management architecture of hybrid emergency power system for more-electric aircrafts", *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, n.º 4, págs. 806-816, 2018.

[79] H. Huang, M. Pasquier y C. Quek, "Financial market trading system with a hierarchical coevolutionary fuzzy predictive model", *IEEE transactions on Evolutionary Computation*, vol. 13, n.º 1, págs. 56-70, 2009.

[80] M. Boroushaki, M. B. Ghofrani, C. Lucas y M. J. Yazdanpanah, "Identification and control of a nuclear reactor core (VVER) using recurrent neural networks and fuzzy systems", *IEEE Transactions on Nuclear Science*, vol. 50, n.º 1, págs. 159-174, 2003.

[81] A. Ahmadian, S. Salahshour y C. S. Chan, "Fractional differential systems: a fuzzy solution based on operational matrix of shifted Chebyshev polynomials and its applications", *IEEE Transactions on Fuzzy Systems*, vol. 25, n.º 1, págs. 218-236, 2016.

[82] A. V. Kostikova, P. V. Tereliansky, A. V. Shuvaev, V. N. Parakhina y P. N. Timoshenko, "Expert fuzzy modeling of dynamic properties of complex systems", *ARPN J. Eng. Appl. Sci*, vol. 11, n.º 17, págs. 10 601-10 608, 2016.

[83] K.-S. Tang, K. F. Man, G. Chen y S. Kwong, "An optimal fuzzy PID controller", *IEEE transactions on industrial electronics*, vol. 48, n.º 4, págs. 757-765, 2001.

[84] V. Azimi, P. Akhlaghi y M. H. Kazemi, "Robust multi objective H2/H∞ control of nonlinear uncertain systems using multiple linear model and ANFIS", en *2011 Chinese Control and Decision Conference (CCDC)*, IEEE, 2011, págs. 2641-2646.

[85] J. Yu y M. Hu, "Fuzzy Constrained Min-Max Model Predictive Control Research of Networked System via Piecewise Lyapunov Functions", en *2018 Chinese Automation Congress (CAC)*, IEEE, 2018, págs. 3109-3114.

[86] W. Ji, M. Wang y J. Qiu, "Fuzzy-model-based output feedback controller design for discrete-time non-affine nonlinear systems via piecewise lyapunov functions", en *2018 Eighth International Conference on Information Science and Technology (ICIST)*, IEEE, 2018, págs. 288-293.

[87] J. Qiu, H. Gao y S. X. Ding, "Recent advances on fuzzy-model-based nonlinear networked control systems: A survey", *IEEE Transactions on Industrial Electronics*, vol. 63, n.º 2, págs. 1207-1217, 2015.

[88] O. Castillo, L. Amador-Angulo, J. R. Castro y M. Garcia-Valdez, "A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems", *Information Sciences*, vol. 354, págs. 257-274, 2016.

[89] S.-J. Heo, Z. Chunwei y E. Yu, "Response Simulation, Data Cleansing and Restoration of Dynamic and Static Measurements Based on Deep Learning Algorithms", *International Journal of Concrete Structures and Materials*, vol. 12, n.º 1, pág. 82, 2018.

[90] S. Mitra e Y. Hayashi, "Neuro-fuzzy rule generation: survey in soft computing framework", *IEEE transactions on neural networks*, vol. 11, n.º 3, págs. 748-768, 2000.

[91] S. Ding, H. Jia, J. Chen y F. Jin, "Granular neural networks", *Artificial Intelligence Review*, vol. 41, n.º 3, págs. 373-384, 2014.

[92] R. Babuška y H. Verbruggen, "Neuro-fuzzy methods for nonlinear system identification", *Annual reviews in control*, vol. 27, n.º 1, págs. 73-85, 2003.

[93] H. Takagi e I. Hayashi, "NN-driven fuzzy reasoning", *International Journal of Approximate Reasoning*, vol. 5, n.º 3, págs. 191-212, 1991.

[94] J. J. Buckley e Y. Hayashi, "Fuzzy neural networks: A survey", *Fuzzy sets and systems*, vol. 66, n.º 1, págs. 1-13, 1994.

[95] N. K. Kasabov, "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems", *Fuzzy sets and Systems*, vol. 82, n.º 2, págs. 135-149, 1996.

[96] J. M. Benítez, J. L. Castro e I. Requena, "Are artificial neural networks black boxes?", *IEEE Transactions on neural networks*, vol. 8, n.º 5, págs. 1156-1164, 1997.

[97] S. Wu y M. J. Er, "Dynamic fuzzy neural networks-a novel approach to function approximation", *IEEE transactions on systems, man, and cybernetics, part B (cybernetics)*, vol. 30, n.º 2, págs. 358-364, 2000.

[98] J. Dong, Y. Wang y G.-H. Yang, "Output feedback fuzzy controller design with local nonlinear feedback laws for discrete-time nonlinear systems", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, n.º 6, págs. 1447-1459, 2010.

[99] S. Ganjefar y M. Tofighi, "Single-hidden-layer fuzzy recurrent wavelet neural network: Applications to function approximation and system identification", *Information Sciences*, vol. 294, págs. 269-285, 2015.

[100] Z. J. Viharos y K. B. Kis, "Survey on Neuro-Fuzzy systems and their applications in technical diagnostics and measurement", *Measurement*, vol. 67, págs. 126-136, 2015.

[101] K. Shihabudheen y G. Pillai, "Recent advances in neuro-fuzzy system: A survey", *Knowledge-Based Systems*, vol. 152, págs. 136-162, 2018.

[102] M. K. Bhuyan, D. P. Mohapatra y S. Sethi, "Software Reliability Prediction using Fuzzy Min-Max Algorithm and Recurrent Neural Network Approach.", *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 6, n.º 4, 2016.

[103] M. Pratama, J. Lu, E. Lughofer, G. Zhang y M. J. Er, "An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks", *IEEE Transactions on Fuzzy Systems*, vol. 25, n.º 5, págs. 1175-1192, 2016.

[104] M. Pratama, P. P. Angelov, J. Lu, E. Lughofer, M. Seera y C. P. Lim, "A randomized neural network for data streams", en *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, págs. 3423-3430.

[105] P. A. Mastorocostas y C. S. Hilas, "ReNFFor: a recurrent neurofuzzy forecaster for telecommunications data", *Neural Computing and Applications*, vol. 22, n.º 7-8, págs. 1727-1734, 2013.

[106] C. H. Vallejos de Schatz, F. K. Schneider, P. J. Abatti y J. C. Nievola, "Dynamic fuzzy-neural based tool formonitoring and predicting patients conditions using selected vital signs", *Journal of Intelligent & Fuzzy Systems*, vol. 28, n.º 6, págs. 2579-2590, 2015.

[107] B.-M. Chang, "A Neuro-Fuzzy System Combined with Particle Swarm Optimization for Handwritten Character Recognition", *Fundamenta Informaticae*, vol. 133, n.º 4, págs. 345-366, 2014.

[108] Q. Li y R.-C. Lin, "A new approach for chaotic time series prediction using recurrent neural network", *Mathematical Problems in Engineering*, vol. 2016, 2016.

[109] D. Krleža y K. Fertalj, "Graph matching using hierarchical fuzzy graph neural networks", *IEEE Transactions on Fuzzy Systems*, vol. 25, n.º 4, págs. 892-904, 2016.

[110] T. Obo, N. Kubota y C. K. Loo, "Evolutionary ensemble learning of fuzzy randomized neural network for posture recognition", en *2016 World Automation Congress (WAC)*, IEEE, 2016, págs. 1-6.

[111] J. Tang, F. Liu, Y. Zou, W. Zhang e Y. Wang, "An improved fuzzy neural network for traffic speed prediction considering periodic characteristic", *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, n.º 9, págs. 2340-2350, 2017.

[112] T. Zhao, P. Li y J. Cao, "Soft sensor modeling of chemical process based on self-organizing recurrent interval type-2 fuzzy neural network", *ISA transactions*, vol. 84, págs. 237-246, 2019.

[113] C.-H. Lee y C.-C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks", *IEEE Transactions on fuzzy systems*, vol. 8, n.º 4, págs. 349-366, 2000.

[114] R. H. Abiyev, "Recurrent neural network based fuzzy inference system for identification and control of dynamic plants", en *Proceedings of International XII Turkish Symposium on Artificial Intelligence and Neural Networks*, vol. 1, 2003, págs. 31-39.

[115] I. Baruch, C.-R. Mariaca-Gaspar, R. G. Guerra et al., "A fuzzy-neural hierarchical multi-model for systems identification and direct adaptive control", en *Analysis and Design of Intelligent Systems using Soft Computing Techniques*, Springer, 2007, págs. 163-172.

[116] C.-H. Lee, T.-W. Hu, C.-T. Lee e Y.-C. Lee, "A recurrent interval type-2 fuzzy neural network with asymmetric membership functions for nonlinear system identification", en *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, IEEE, 2008, págs. 1496-1502.

[117] Y.-x. Zhang, "Fuzzy neural network adaptive robust control of a class of nonlinear interconnected system based on TS fuzzy model", en *2010 Chinese Control and Decision Conference*, IEEE, 2010, págs. 2561-2564.

[118] F. Zahedi y Z. Zahedi, "A review of neuro-fuzzy systems based on intelligent control", *Journal of Electrical and Electronic Engineering*, págs. 58-61, 2015.

[119] S. Hochreiter y J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, n.º 8, págs. 1735-1780, 1997.

[120] F. Gers, J. Schmidhuber y F. Cummins, "Learning to forget: continual prediction with LSTM", *IET Conference Proceedings*, págs. 850-855, ene. de 1999.

[121] T. Ergen y S. S. Kozat, "Efficient online learning algorithms based on LSTM neural networks", *IEEE transactions on neural networks and learning systems*, vol. 29, n.º 8, págs. 3772-3783, 2017.

[122] Y. Gal y Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks", en *Advances in neural information processing systems*, 2016, págs. 1019-1027.

[123] Y. Jin y B. Sendhoff, "Extracting interpretable fuzzy rules from RBF networks", *Neural Processing Letters*, vol. 17, n.º 2, págs. 149-164, 2003.

[124] H. K. Khalil, *Nonlinear systems*. Prentice hall Upper Saddle River, NJ, 2002.

[125] P. Hénaff, V. Scesa, F. B. Ouezdou y O. Bruneau, "Real time implementation of CTRNN and BPTT algorithm to learn on-line biped robot balance: Experiments on the standing posture", *Control Engineering Practice*, vol. 19, n.º 1, págs. 89-99, 2011.

[126] K. Chen, Z.-J. Yan y Q. Huo, "A context-sensitive-chunk BPTT approach to training deep LSTM/BLSTM recurrent neural networks for offline handwriting recognition", en *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2015.

[127] T. P. Lillicrap y A. Santoro, "Backpropagation through time and the brain", *Current Opinion in Neurobiology*, vol. 55, págs. 82-89, 2019.

[128] D. K. Sari, D. P. Wulandari e Y. K. Suprapto, "Training Performance of Recurrent Neural Network using RTRL and BPTT for Gamelan Onset Detection", *Journal of Physics: Conference Series*, vol. 1201, n.º 1, pág. 012 046, 2019.

[129] I. Baruch, C. Mariaca-Gaspar y J. Barrera-Cortes, "Recurrent neural network identification and adaptive neural control of hydrocarbon biodegradation processes", en *Recurrent Neural Networks*, IntechOpen, 2008.

[130] P. Sastry, G Santharam y K. Unnikrishnan, "Memory neuron networks for identification and control of dynamical systems", *IEEE transactions on neural networks*, vol. 5, n.º 2, págs. 306-319, 1994.

[131] J. Kabziński y J. Kacerka, "TSK Fuzzy Modeling with Nonlinear Consequences", en *IFIP International Conference on Artificial Intelligence Applications and Innovations*, Springer, 2014, págs. 498-507.

[132] H. T. Siegelmann, B. G. Horne y C. L. Giles, "Computational capabilities of recurrent NARX neural networks", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, n.º 2, págs. 208-215, 1997.

[133] S. A. Ramu y V. Johnson, "Damage assessment of composite structures-A fuzzy logic integrated neural network approach", *Computers & structures*, vol. 57, n.º 3, págs. 491-502, 1995.

[134] J. P. Sawyer y S. Rao, "Structural damage detection and identification using fuzzy logic", *AIAA journal*, vol. 38, n.º 12, págs. 2328-2335, 2000.

[135] Z. Zhao y C. Chen, "A fuzzy system for concrete bridge damage diagnosis", *Computers & structures*, vol. 80, n.º 7-8, págs. 629-641, 2002.

[136] L. Meyyappan, M. Jose, C. Dagli, P. Silva y H. Pottinger, "Fuzzy-neuro system for bridge health monitoring", en *22nd International Conference of the North American Fuzzy Information Processing Society, NAFIPS 2003*, IEEE, 2003, págs. 8-13.

[137] P. J. Dempsey y A. A. Afjeh, "Integrating oil debris and vibration gear damage detection technologies using fuzzy logic", *Journal of the American Helicopter Society*, vol. 49, n.º 2, págs. 109-116, 2004.

[138] C. K. Soh y S. Bhalla, "Calibration of piezo-impedance transducers for strength prediction and damage assessment of concrete", *Smart materials and structures*, vol. 14, n.º 4, pág. 671, 2005.

[139] W. Wang, F. Ismail y F. Golnaraghi, "A neuro-fuzzy approach to gear system monitoring", *IEEE transactions on Fuzzy Systems*, vol. 12, n.º 5, págs. 710-723, 2004.

[140] E. Zio y G. Gola, "A neuro-fuzzy technique for fault diagnosis and its application to rotating machinery", *Reliability Engineering & System Safety*, vol. 94, n.º 1, págs. 78-88, 2009.

[141] E. W. Lee y H.-F. Lam, "ANN-based structural damage diagnosis using measured vibration data", en *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, Springer, 2004, págs. 373-379.

[142] P. M. Pawar y R. Ganguli, "Genetic fuzzy system for online structural health monitoring of composite helicopter rotor blades", *Mechanical Systems and Signal Processing*, vol. 21, n.º 5, págs. 2212-2236, 2007.

[143] M. R. Taha y J. Lucero, "A generic fuzzy metric for damage recognition in structural health monitoring systems", en *2005 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, vol. 2, 2005, págs. 1518-1523.

[144] F. Zhu e Y. Wu, "A rapid structural damage detection method using integrated ANFIS and interval modeling technique", *Applied Soft Computing*, vol. 25, págs. 473-484, 2014.

[145] I. Lopez y N. Sarigul-Klijn, "A review of uncertainty in flight vehicle structural damage monitoring, diagnosis and control: Challenges and opportunities", *Progress in Aerospace Sciences*, vol. 46, n.º 7, págs. 247-273, 2010.

[146] O. Cordón, "A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems", *International journal of approximate reasoning*, vol. 52, n.º 6, págs. 894-913, 2011.

[147] S.-F. Jiang, C.-M. Zhang y S. Zhang, "Two-stage structural damage detection using fuzzy neural networks and data fusion techniques", *Expert systems with applications*, vol. 38, n.º 1, págs. 511-519, 2011.

[148] S. Thenozhi y W. Yu, "Advances in modeling and vibration control of building structures", *Annual Reviews in Control*, vol. 37, n.º 2, págs. 346-364, 2013.

[149] M. Gordan, H. A. Razak, Z. Ismail y K. Ghaedi, "Recent developments in damage identification of structures using data mining", *Latin American Journal of Solids and Structures*, vol. 14, n.º 13, págs. 2373-2401, 2017.

[150] X. Zhao, R. Han, B. Xie, J. Li, J. Ou y K. J. Loh, "Shaking table tests for evaluating the damage features under earthquake excitations using smartphones", en *Health Monitoring of Structural and Biological Systems XII*, T. Kundu, ed., SPIE, 2018.

[151] P. Paul et al., "An Internet of Things (IoT) Based System to Analyze Real-time Collapsing Probability of Structures", en *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, IEEE, 2018.

[152] S. I. Abdullahi, N. A. C. Mustapha, M. H. Habaebi y M. R. Islam, "Accelerometer Based Structural Health Monitoring System on the Go: Developing Monitoring Systems with NI LabVIEW", *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 15, n.º 07, pág. 32, 2019.

[153] G. Morgenthal, J. F. Eick, S. Rau y J. Taraben, "Wireless Sensor Networks Composed of Standard Microcomputers and Smartphones for Applications in Structural Health Monitoring", *Sensors*, vol. 19, n.º 9, pág. 2070, 2019.

[154] G. M. Guzman-Acevedo et al., "GPS, Accelerometer, and Smartphone Fused Smart Sensor for SHM on Real-Scale Bridges", *Advances in Civil Engineering*, vol. 2019, págs. 1-15, 2019.

[155] E. Ozer y M. Q. Feng, "Structural Reliability Estimation with Participatory Sensing and Mobile Cyber-Physical Structural Health Monitoring Systems", *Applied Sciences*, vol. 9, n.º 14, pág. 2840, 2019.

[156] D. Zhang, J. Tian y H. Li, "Design and Validation of Android Smartphone Based Wireless Structural Vibration Monitoring System", *Sensors*, vol. 20, n.º 17, pág. 4799, 2020.

[157] V. B. Sharma et al., "Recent Advancements in AI-Enabled Smart Electronics Packaging for Structural Health Monitoring", *Metals*, vol. 11, n.º 10, pág. 1537, 2021.

[158] A. A. Alzughaibi, A. M. Ibrahim, Y. Na, S. El-Tawil y A. M. Eltawil, "Community-Based Multi-Sensory Structural Health Monitoring System: A Smartphone Accelerometer and Camera Fusion Approach", *IEEE Sensors Journal*, vol. 21, n.º 18, págs. 20 539-20 551, 2021.

[159] R. Han y X. Zhao, "Shaking Table Tests and Validation of Multi-Modal Sensing and Damage Detection Using Smartphones", *Buildings*, vol. 11, n.º 10, pág. 477, 2021.

[160] J. Gonzalez y W. Yu, "Non-linear system modeling using LSTM neural networks", *IFAC-PapersOnLine*, vol. 51, n.º 13, págs. 485-489, 2018.

[161] J. Tavoosi, A. Mohammadzadeh y K. Jermsittiparsert, "A review on type-2 fuzzy neural networks for system identification", *Soft Computing*, vol. 25, n.º 10, págs. 7197-7212, 2021.