CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE CONTROL AUTOMÁTICO

Pronóstico de series de tiempo empleando redes neuronales y meta-transferencia de aprendizaje.

TESIS

Que presenta

Mario Cesar Maya Rodriguez

para obtener el Grado de

DOCTOR EN CIENCIAS

EN LA ESPECIALIDAD DE

CONTROL AUTOMÁTICO

Directores de la Tesis:

Dr. Wen Yu Liu
Dr. Floriberto Ortiz Rodríguez

Ciudad de México Mayo, 2022

Agradecimientos

A mis padres David y Delia que con esfuerzo me dieron la oportunidad de llegar a este punto de mi vida. A mi familia que siempre me apoyo en todo momento.

A mis grandes amigos que me gustaría mencionarlos, pero esto no es posible. Sin embargo, cada uno de ellos sin temor a equivocarme estoy seguro que saben quienes son. También para aquellos que hoy ya no nos acompañan en este plano físico pero que aún se encuentran en mis recuerdos.

A quién fue, es y sera. A ella, que aunque ya no me es posible estar a su lado pero siempre tendrá un lugar en mi corazón, en mi pensamiento y en mis palabras.

A mis directores de tesis el Dr. Wen Yu Liu y el Dr. Floriberto Ortíz Rodríguez por brindarme su apoyo, tiempo y guía para la realización de este trabajo.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) y al Centro de Investigación y Estudios Avanzados (CINVESTAV), por el apoyo económico para la realización de mis estudios de posgrado.

Sin ustedes este trabajo no seria posible.

A todos, Muchas Gracias.

Resumen

En este trabajo de tesis se proponen modificaciones a los métodos de Meta-Aprendizaje (ML) y Transferencia-Aprendizaje (TL), así como la generación de un nuevo algoritmo basado en los concepto anteriores llamado Meta-Transferencia de Aprendizaje (MTA) que además cuenta con un algoritmo de búsqueda de características especiales basado en la Transformada *Wavelet* (TW) para la identificación de sistemas no lineales y predicción de series de tiempo caóticas, todo esto a través del uso de una topología clásica de una Red Neuronal Artificial (RNA) como lo es el Perceptrón Multicapa (MLP). También, se hace la demostración matemática de las propiedades de estabilidad y convergencia (débil y fuerte) para la etapa de entrenamiento. Finalmente, se proponen una serie de casos basados en problemas del mundo real: identificación de sistemas no lineales, predicción de corto y largo plazo para la magnitud de terremotos y el pronóstico de contaminantes ambientales bajo la ausencia de información debido a fallas en el sistema de adquisición de datos, además, el desempeño del algoritmo propuesto es comparado con redes neuronales clásicas como el (MLP) y arquitecturas de aprendizaje profundo. Por otra parte, se hace una modificación al control Proporcional-Integral-Derivativo (PID) para el rechazo de perturbaciones. También se definen los algoritmos que permiten su implementación y su posterior ejecución.

Abstract

In this work modifications are proposed to the methods of Meta-Learning (ML) and Transfer-Learning (TL), as well as the generation of a new algorithm based on the previous concepts called Meta-Transfer of learning (MTA) that also contain with a search algorithm for special characteristics based on the Wavelet Transform for the identification of non-linear systems and prediction of chaotic time series, all this through the use of a classical artificial neural network (RNA) topology such as the multilayer perceptron (MLP). On the other hand, the mathematical demonstration of the properties of stability and convergence (weak and strong) is made for the training stage. Finally, a series of cases based on real-world problems are proposed: identification of non-linear systems, prediction of short and long term of the magnitude of earthquakes and the forecast of environmental pollutants in the absence of information due to failures in the data acquisition system, its performance is also compared with classic neural networks such as MLP and deep learning architectures; on the other hand, a modification is made to the proportional-integral-derivative control (PID). The algorithms that allow its implementation and execution to solve the proposed problems are also defined.

Índice general

Índice de Figuras

Índice de Tablas

Abreviaturas

ML	*Meta-Learning.* Meta-Aprendizaje.
TL	*Transfer-Learning* Transferencia-Aprendizaje.
MTA	Meta-Transferencia de Aprendizaje.
TW	Transformada *Wavelet*.
RNA	Red Neuronal Artificial.
MLP	*Multi-Layer Perceptron.* Perceptrón Multicapa.
PID	Proporcional-Integral-Derivativo.
MSE	*Mean Square Error.* Error Cuadrático Medio.
BP	*Back-Propagation.* Retro-Propagación hacia atrás.
$NARX$	*Nonlinear Autoregressive Exogenous.* Modelo Exógeno Auto-Regresivo no Lineal.
DWT	*Discrete Wavelet Transform.* Transformada Discreta *Wavelet*.
ISS	*Input-to-State Stability.* Entrada-Estado Estable.
BPM	*Back-Propagation with Meta-Learning.*Retro-Propagación con Meta-Aprendizaje.
BPS	*Back-Propagation Stable.* Retro-Propagación variable en el tiempo.

Capítulo 1

Introducción

En el área de investigación de la Teoría del Control Automático los esfuerzos están enfocados en dos directrices principales: Identificación y Control de sistemas. En muchas ocasiones son actividades inclusivas pero es posible estudiarlas individualmente. La identificación de sistemas es el proceso de estimar modelos o parámetros de sistemas dinámicos de acuerdo con los datos adquiridos de este [1]. Por otro lado, el control de sistemas se encarga de que el comportamiento de un proceso siga una referencia bajo la presencia de perturbaciones. Los modelos matemáticos utilizados van desde una ecuación lineal hasta modelos más complicados como lo son las redes neuronales. La principal característica de las redes neuronales artificiales radica en su naturaleza, donde a partir del entorno (entradas/salidas) obtienen su conocimiento a través de un proceso adaptativo denominado aprendizaje [2]. Se ha demostrado en innumerables trabajos que las redes neuronales pueden ser utilizadas tanto para la identificación de sistemas como para el control de los mismos [3]. El aprendizaje automático ha tomado diversos caminos para aumentar su capacidad de resolución de problemas complejos, sin embargo, en muchas ocasiones el entendimiento y la posterior implementación de los algoritmos resulta ser complicado para quienes no cuentan con un antecedente substancial a nivel teórico de los tópicos. Es común observar una evolución vertical en la teoría de los métodos, no obstante, aún cuando las diferentes topologías de una red neuronal son analizadas exhaustivamente, no se han resuelto las

problemáticas comunes de los métodos de una manera satisfactoria o, en su defecto, aún es posible realizar avances debido al interés general sobre los retos que ofrecen. Algunos de los métodos desarrollados como lo son el Meta-Aprendizaje [4] y la Transferencia-Aprendizaje [5] dotan al aprendizaje automático con propiedades y características para superar problemas en el kernel de los algoritmos académicamente utilizados o permiten potenciar las capacidades para la resolución de los mismos. En Control Automático escasamente se ha explorado como las propiedades de los métodos de ML y TL pueden ser utilizadas para mejorar o generar algoritmos, técnicas o métodos para identificación y control de sistemas.

En este trabajo se presenta un panorama general de los métodos de Meta-Aprendizaje y Transferencia-Aprendizaje. Además, se proponen modificaciones en los métodos ML y TL para mostrar una manera de resolver las dificultades que se presentan en el uso de perceptrón MLP como topología de red neuronal. Posteriormente, se combinan las características de los métodos propuestos para generar un método denominado Meta-Transferencia de Aprendizaje. Se analizan matemáticamente las propiedades de estabilidad y convergencia de los métodos propuestos. Finalmente, se muestran diferentes casos de estudio tales como el pronóstico de series de tiempo abordando problemas complejos como la predicción de sismos y el pronóstico de contaminantes ambientales; el interés en este tipo de problemáticas se debe a que tienen una alta sensibilidad a las condiciones iniciales así como las series de tiempo no cumplen con las características de periodicidad y tendencia regular. Además, se muestra también su uso para la identificación no paramétrica de sistemas no lineales y se hace una mejora al control PID basado en en redes neuronales, la cual permite mejorar el desempeño en el rechazo activo de perturbaciones.

1.1. Motivación

Las redes neuronales artificiales tienen alto uso en diversos campos de la investigación, sin embargo, en ocasiones se opta por una evolución hacia la innovación de técnicas o métodos a nivel teórico, pero con un costo de implementación superior tanto computacional como en complejidad para su manejo por personas ajenas a las áreas de ciencias de la computación.

Por esta razón, a nivel práctico existen muchos de estos algoritmos que no han demostrado su eficacia en la resolución de problemas del mundo real. Es importante destacar que los resultados reportados en tesis y artículos científicos sólo constituyen una ilustración de un método, técnica, algoritmos propuestos en estos trabajos y por ende no es posible saber con certeza los límites de las herramientas empleadas. Las redes neuronales *MLP* son un claro ejemplo de lo antes mencionado, se usan con frecuencia para comparar con las nuevas técnicas, pero en muchas ocasiones es injusta dicha comparación y no se sintonizan parámetros adecuados para la problemática en cuestión. Si bien los esfuerzos deben de ir en orden de depender cada vez menos de los hiper-parámetros presentes en métodos basados en redes neuronales y tener mejores desempeños, aún es posible hacer mejoras a este tipo de topologías haciendo uso de la combinación de nuevas técnicas dentro del aprendizaje automático y dotar de nuevas características a una estructura y modelo neuronal que tiene como principal propiedad una sencillez que permite su fácil implementación aún para quienes no tiene una formación académica fuerte en los tópicos afines.

1.2. Objetivos

General:

- Desarrollar un método basado en la combinación de los métodos Meta-Aprendizaje y Transferencia-Aprendizaje para el pronóstico de series de tiempo.

Particulares:

- Modificar el método de Meta-Aprendizaje para la identificación de sistemas no lineales ante la presencia de perturbaciones.
- Diseñar una método para el uso de Meta-Aprendizaje y Transferencia-Aprendizaje en la predicción de series de tiempo.
- Proponer un algoritmo de búsqueda de patrones para la especificación de la similitud entre bases de datos y su uso en el método de Transferencia-Aprendizaje.

- Diseñar un método de Meta-Transferencia de aprendizaje para la superación de las limitantes de un red neuronal multicapa.
- Demostrar matemáticamente las propiedades de estabilidad y convergencia del método de Meta-Transferencia de Aprendizaje.
- Utilizar el Meta-Aprendizaje modificado para la identificación de sistemas no lineales.
- Realizar pronósticos de corto y largo plazo para series de tiempo caóticas.
- Implementar el método de Transferencia-Aprendizaje para el problema de pérdida de información en bases de datos.
- Realizar simulaciones y comparaciones numéricas de los métodos propuestos y los métodos clásicos basados en redes neuronales.

1.3. Estructura

El trabajo de tesis está estructurado del a manera siguiente. El Capítulo 2 describe en el marco teórico de las redes neuronales para la identificación de sistemas dinámicos no lineales y la predicción de series de tiempo, y se presenta una revisión de las limitantes de los modelos neuronales multicapa. Además, incluye una breve historia de la evolución los métodos de Meta-Aprendizaje y de Transferencia-Aprendizaje. El Capítulo 3 expone la identificación de sistemas no lineales utilizando redes neuronales, se presentan las modificaciones a los métodos Meta-Aprendizaje y Transferencia-Aprendizaje; también se muestra el proceso para el algoritmo de búsqueda de características especiales basado en la TW para complementar el método de Transferencia-Aprendizaje. Por otra parte, se propone un método de Meta-Transferencia de aprendizaje, el cual permite combinar y utilizar las ventajas de lo métodos antes mencionados. Por otra parte, se realiza una demostración matemática de las propiedades importantes para la teoría de Control Automático como lo son la estabilidad y convergencia del método. El Capítulo 4 aborda diferentes casos de aplicación de los métodos ML, TL y MTA. Se aborda la predicción de series de tiempo caóticas como lo es el pronóstico de magnitud para terremotos para los modelos de corto

plazo, largo plazo, y de predicción en multi-horizonte. También, se describe la identificación de sistemas no lineales empleando el modelo de predicción paralelo. Se trata el problema de pérdida de información en la recolección de datos para las series de tiempo en un problema del mundo real como lo es el pronóstico de los contaminantes del aire. Finalmente, en el Capítulo 5 las conclusiones y el trabajo a futuro son presentados.

1.4. Contribuciones

1. El método de Meta-Aprendizaje se modifica para obtener una mejora en la precisión de la predicción y robustez en presencia de ruido. Las modificaciones consisten en agregar una tasa de aprendizaje variable en el tiempo y un término de momentum.

2. Se modifica el método de Transferencia-Aprendizaje para superar el problema en la pérdida de información en una serie de tiempo.

3. Se desarrolla un algoritmo de búsqueda de patrones para la especificación de la similitud entre bases de datos basado en la Transformada *Wavelet* y su posterior uso en el método de Transferencia-Aprendizaje

4. Se propone un método de Meta-Transferencia de Aprendizaje.

5. Se demuestra la propiedad de estabilidad del método propuesto en el entrenamiento del método de Meta-Transferencia de Aprendizaje.

6. Se demuestra la propiedad de convergencia débil y fuerte del método método de Meta-Transferencia de Aprendizaje.

7. Se muestran diferentes maneras de utilizar los métodos de ML, TL y MTA para resolver limitantes clásicas de las redes neuronales multi-capas.

1.5. Publicaciones

1.5.1. Congresos

- Mario Maya, Wen Yu, Short-term prediction of the earthquake through Neural Networks and Meta-Learning, 16th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE19), Mexico City, Mexico, 2019.
- Maya Mario, Yu Wen & Li Xiaoou, Time series forecasting with missing data using neural network and meta-transfer learning, 2021 IEEE Symposium Series on Computational Intelligence (SSCI 2021), Orlando, Florida, USA, December 4th - 7th 2021.

1.5.2. Revistas

- Mario Maya, Wen Yu & Luciano Telesca (2021) Multi-Step Forecasting of Earthquake Magnitude Using Meta-Learning Based Neural Networks, Cybernetics and Systems, DOI: 10.1080/01969722.2021.1989170
- Maya, Mario, Yu, Wen & Telesca, Luciano. (2021). Neural networks for long-term earthquake prediction using modified meta-learning. Journal of Intelligent & Fuzzy Systems. 1-14. 10.3233/JIFS-210173.
- Meta-Transfer Learning Using Wavelet Decomposition for Multi-Horizon Time Series Forecasting
  Mario Maya; Wen Yu
  IEEE ACCESS. Aceptado
- Meta-transfer learning for air pollution forecasting with missing data
  Mario Maya; Wen Yu
  ISA Transactions

Capítulo 2

Identificación no paramétrica de sistemas dinámicos no lineales a través aprendizaje automático

La predicción de series de tiempo es de especial interés para los investigadores, debido a que el pronóstico de largo plazo aún representa una problemática fuerte para el aprendizaje automático, la inteligencia artificial o la informática. Hoy en día, los mayores esfuerzos de los investigadores se centran en encontrar nuevas formas de obtener mejores resultados en el problema de la predicción. La predicción de series de tiempo permite determinar valores futuros en alguna tarea a partir de registros históricos pasados, considerando un margen de error de predicción. Se utilizan muchos métodos populares para resolver este problema, por ejemplo, el enfoque de Box y Jekins [6]. En [7] se hace una revisión de los métodos más habituales para solucionar este problema. Las RNA se han utilizado con éxito para lograr el pronóstico de las series temporales [8], [9], [10]. El MLP es el tipo de red neuronal más común, cuyas entradas y salidas están vinculadas por una o más capas ocultas. Sin embargo, el MLP tiene dos problemas principales: la convergencia lenta y el mínimo local, [11]. Para evitar estos problemas, se han desarrollado varias ramas del aprendizaje automático, por ejemplo, el aprendizaje profundo [12].

2.1. Series de tiempo y modelos de caja negra

El pronóstico de series de tiempo no sólo es uno de los campos más actuales, sino también uno de los más importantes en el aprendizaje automático, la inteligencia artificial o la informática, [13], [6]. Hoy en día, los investigadores realizan grandes esfuerzos para encontrar nuevas formas de obtener mejores resultados en los problemas de pronóstico. La predicción de series de tiempo permite obtener, considerando un margen de error, valores futuros a partir de registros históricos pasados. Muchos problemas de las series de tiempo no ofrecen suficiente información para usar las técnicas clásicas [14]. El enfoque de caja negra, sin embargo, permite resolver este problema, donde los modelos neuronales son las estructuras más comúnmente utilizadas [15].

2.1.1. Series de Tiempo

Todos los fenómenos de la naturaleza pueden ser interpretados a través de magnitudes físicas, tales cómo: temperatura, corriente, voltaje, velocidad, frecuencia, pH, presión, flujo, par, entre muchas otras. Estas variables dependientes del tiempo pueden ser llamadas “estados". En el área de Control Automático los sistemas de adquisición de datos son aquellos que permiten recolectar la información de los estados por medio del uso de sensores y transductores, los cuales convierten una señal analógica medible en información digital por cada tiempo de muestreo; cuando estas observaciones son generadas secuencialmente a través del tiempo entonces pueden ser llamadas series de tiempo en tiempo discreto.

Las series de tiempo pueden clasificarse en dos ramas: deterministas y estocásticas. La primera de ellas tendrá lugar cuando los datos adquiridos secuencialmente son dependientes, esto significa que los valores futuros pueden ser pronosticados de manera exacta a partir de los eventos pasados. Sin embargo, es común observar que el pronóstico no depende únicamente de valores pasados, por lo que la predicción suele no ser exacta, dando lugar a un factor de probabilidad el cual estará condicionado al conocimiento a priori de datos históricos. Una serie de tiempo estocástica se define en el momento que la varianza tiende a

infinito con el paso del tiempo, [13]. Cabe señalar, que las series de tiempo caóticas tienen un comportamiento parecido a la series de tiempo estocásticas pero su aleatoriedad es totalmente determinista, esto es gran interés porque los datos pueden ofrecer información útil sobre un fenómeno, como se menciona en [16].

Las Series de Tiempo tienen cuatro objetivos fundamentales, [13]:

- Descripción: Constituye un análisis simple basado en la obtención de la gráfica de los datos disponibles contra el tiempo, el gráfico es usado para obtener y determinar propiedades de la serie a través de inspección visual o métodos de uso sencillo.
- Correlación: Cuando se involucran dos o más variables en el análisis de un mismo fenómeno, es posible analizar la dependencia que existe entre ellas y la sensibilidad de una variable ante la variación de comportamiento de otro estado.
- Predicción: A partir de información previa se busca determinar los valores futuros inmediatos o a través de un multi-horizonte. Comúnmente los términos predicción y pronóstico son usados indiscriminadamente, sin embargo, en [17] se menciona que "pronóstico" debe ser usado para métodos objetivos, mientras que "predicción" debe ser usado para métodos subjetivos.
- Control: En este objetivo se pueden tener dos enfoques fundamentales. El primero tiene relación con el Control Predictivo, ya que la acción de control está sujeta a la predicción de perturbaciones, como de tendencia no deseada de la variable de control y con esto realizar una acción oportuna por parte del controlador. El segundo es más profundo y complicado, ya que con el uso de la información disponible, es posible modelarla y con ello trabajar con una estrategia/ley de control "óptima".

Toda serie de tiempo puede ser caracterizada a través de la presencia o ausencia de ciertas propiedades que pueden ayudar a determinar el método o técnica especial para alcanzar el objetivo que se tenga de la serie. Estas propiedades pueden ser: tendencia, estacionalidad, periodicidad o cualquier elemento de correlación con eventos pasados sistemáticos. Este tipo de series pueden usar modelos de pronóstico lineales, ya que su complejidad suele ser baja.

Por otro lado, cuando se tienen series de tiempo caóticas, es decir, que se tiene una alta sensibilidad a las condiciones iniciales, el pronóstico exacto es imposible ya que el error aumentara rápidamente con la más ligera inexactitud de la información disponible en las bases de datos, debido a fallas en la adquisición de datos, ruido, en otras. Algunos ejemplos de ello pueden ser comportamientos: financieros, climatológicos, minería de datos, entre otros [18].

2.1.2. Modelos de Caja Negra

La identificación no paramétrica se encuentra basada en el hecho de que no se tiene disponible un modelo matemático que definan el comportamiento de un sistema a través de ecuaciones diferenciales ordinarias o parciales, esto regularmente se debe a que el proceso presenta una gran complejidad en su naturaleza; por lo que el comportamiento estará descrito únicamente a través de un modelo experimental y este tipo de modelos son comúnmente llamados modelos de caja negra. Por otro lado, los modelos matemáticos teóricos llamados modelos de caja blanca, son aquellos que puedan proveer mayor información acerca del sistema en cuestión, explican la dinámica de un sistema a través del tiempo ya que se conoce tanto la estructura y los parámetros del modelo. Ambos modelos presentan una clara desventaja, los de caja blanca suponen conocer todo, y tal vez en ambientes controlados esto sea posible, sin embargo, en condiciones industriales esto es difícil de lograr para procesos en los que intervengan variables analíticas o se tenga movimiento de materia a través de un conducto; cabe señalar que existen sistemas electromecánicos, como los robots, que su modelo y parámetros han sido identificados con una alta exactitud, mientras que los modelos de caja negra vistos desde el punto de las redes neuronales artificiales tienen una gran desventaja: las ecuaciones resultantes no pueden ser interpretadas físicamente, ver Figura (2.1). Ahora bien, es preferible tener la combinación de ambos enfoques dando lugar a modelos de caja gris. Sin embargo, en las ultimas décadas, los modelos de caja negra son cada vez más utilizados debido que:

- El análisis físico y matemático, para la obtención de un modelo basado en ecuaciones

diferenciales puede volverse bastante complejo incluso para sistemas simples.

- En su mayoría, los coeficientes del modelo derivados de las consideraciones teóricas no son suficientemente precisos.
- No se conocen todas las acciones que tienen lugar dentro del sistema.
- Las acciones que tienen lugar no se pueden describir matemáticamente con los requisitos precisión.
- Algunos sistemas son muy complejos, lo que hace que el análisis teórico tenga un costo temporal muy alto, por lo que no es práctico fuera del campo de la investigación.
- Los modelos identificados se pueden obtener en un tiempo más corto y con menos esfuerzo en comparación que los modelos de caja blanca.

A través de la medición de la entrada y salida de un proceso, permite que el análisis experimental desarrolle modelos matemáticos. Sin embargo, sólo se obtienen modelos que gobiernan el comportamiento de entrada-salida del sistema, es decir, los modelos en general no describirán la estructura interna precisa del sistema. Estos modelos de entrada-salida son aproximaciones y aún así son suficientes para muchas áreas de aplicación. Si el sistema también permite la medición de estados, obviamente también se puede recopilar información sobre la estructura interna del sistema. Tradicionalmente las ecuaciones diferenciales (tiempo continuo) o las ecuaciones en diferencias (tiempo discreto) son usadas para describir el comportamiento en el tiempo de un sistema dinámico. Como se mencionó, en algunas ocasiones resulta complejo el camino analítico, en particular cuando el requisito de adaptación está presente para sistemas variantes en el tiempo.

2.2. Modelado de sistemas dinámicos no lineales a través de redes neuronales

Las redes neuronales son estructuras de modelos de caja negra no lineales que se utilizan como métodos convencionales de estimación de modelos. Tienen buenas capacidades de

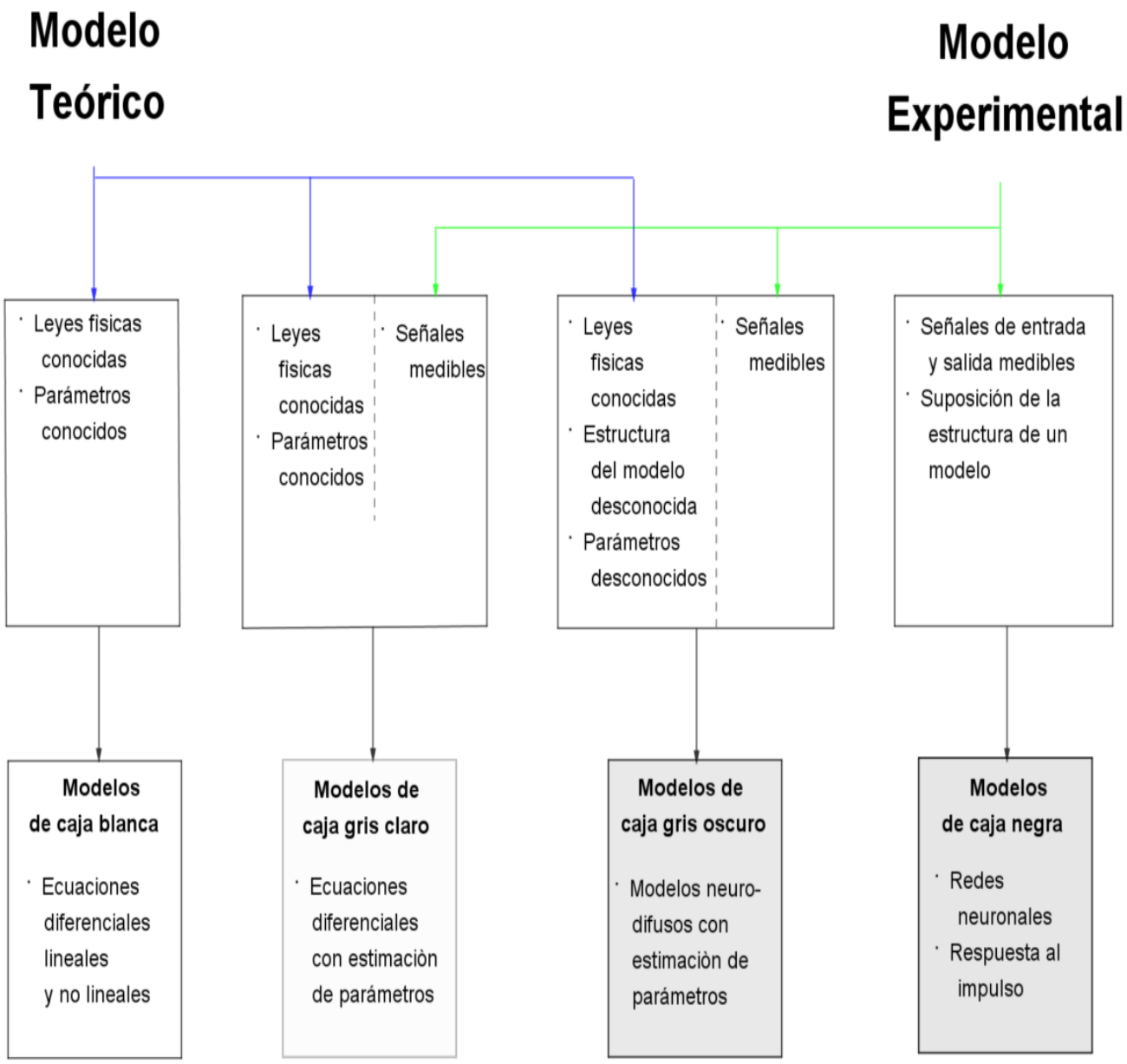


Figura 2.1: Taxonomía de los tipos de modelos.

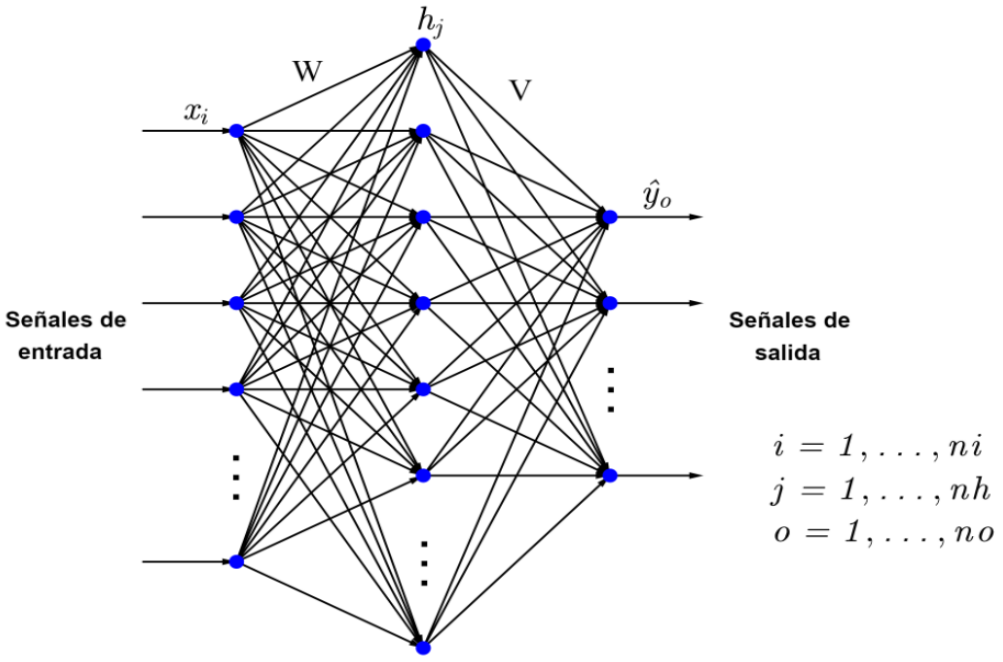


Figura 2.2: Arquitectura de red neuronal multicapa.

aproximación general para sistemas no lineales complejos. Al proponer los hiper-parámetros en estas estructuras, también existe una buena adaptabilidad para concentrarse en aquellos parámetros que tienen la mayor importancia para el conjunto de datos en particular.

2.2.1. Redes neuronales multicapa

Las redes neuronales artificiales tienen varias arquitecturas, un caso particular es cuando se estructuran por capas (ver Figura 2.2), comúnmente llamadas perceptrón multi-capas, siendo este tipo de las más utilizadas a lo largo de la historia del campo de investigación e incluso en la actualidad debido a la simplicidad para su aplicación e implementación [19]. Son usadas para realizar tareas tales como: regresión [20], pronóstico de series de tiempo [21], clasificación [22], comprensión de datos [23], control de sistemas [24]. La arquitectura MLP está compuesta por lo siguiente:

- $X(k)$ contiene los valores utilizados por la capa de pesos en la entrada:

$$X(k) = [x_1, x_2, ..., x_i], i = 3, 4, 5, ..., n_i,$$

  donde n_i es el número de entradas seleccionadas.

- $h(k)$ contiene la respuesta a la salida de los nodos de la capa oculta:

$$H(k) = [h_1, h_2, ..., h_j], j = 3, 4, 5, ..., n_h,$$

  donde n_h es el número de nodos seleccionados.

- $\hat{Y}(k)$ contiene la respuesta a la salidas de la red neuronal:

$$\hat{Y}(k) = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_o], o = 3, 4, 5, ..., n_o,$$

  donde n_o es el número de salidas seleccionadas.

La matriz de pesos comúnmente referida a las capas ocultas, i.e., ver la Figura (2.2), la cual es un MLP con dos capas, una capa oculta (W_{ij}) y una capa de salida (V_{jo}). La matriz $W_{ij} \in \Re^{(n_i \times n_h)}$ comunica a la capa de entrada con la capa oculta, mientras que la matriz $V_{jo} \in \Re^{(n_h \times n_o)}$ comunica a la capa los nodos de salida de la capa oculta con la salida de la red neuronal. Para la inicialización de las matrices de pesos se eligen valores aleatorios dentro de un intervalo definido $[a, b]$.

En la capa de entrada, el efecto de las entradas a una neurona se considera aditivo, por lo tanto la entrada neta que recibe un nodo es la suma del producto de cada señal individual por el peso sináptico que los une, esto es:

$$neth_j(k) = \sum_{i=1}^{ni} w_{ij}(k) x_i(k); j = 1, ..., n_h. \tag{2.1}$$

Cada nodo recibe y envía señales a las demás neuronas, por lo que después de calcular la

entrada neta, es necesario procesarla para obtener una respuesta. La función de activación $\Gamma(\cdot)$ procesa la información y determina si esta es propagada a la siguiente capa. Definiendo la salida de la capa oculta:

$$h_j = \Gamma(neth_j(k)). \tag{2.2}$$

Si la MLP tiene más capas, basta con agregar un superíndice l en la ecuación anterior y realizar la operación definida por la ecuación (2.1) cambiando el vector de entradas por la salida de la capa anterior y escogiendo la matriz de pesos comunicantes de la capa. La salida del número capas l es:

$$h_j^l = \Gamma(neth_j^l(k)). \tag{2.3}$$

Para la obtención de la salida de la red neuronal se ejecuta el mismo mecanismo mencionado en el paso anterior considerando:

$$neto_o(k) = \sum_{j=1}^{nh} v_{jo}(k)h_j(k); o = 1, ..., n_o; \tag{2.4}$$

$$\hat{y}_o = \Phi(neto_o(k)), \tag{2.5}$$

donde $\Phi(\cdot)$ es la función de activación.

El proceso de aplicación de una red neuronal consta de las etapas de aprendizaje y generalización. La primera, también denominada etapa de entrenamiento [25], se emplea un conjunto de elementos extraídos de bases de datos, estos son utilizados como patrones y con ello se calcula los pesos sinápticos de la red neuronal conforme es recorrido dicho conjunto los cuales fueron iniciados aleatoriamente. Al realizar el paso hacia adelante se obtiene una respuesta estimada por la red neuronal y al comparar esta con la salida real del sistema se determina el error de estimación y se elige una función de costo para el entrenamiento de los pesos sinápticos. La etapa de entrenamiento se realiza de forma iterativa donde se busca minimizar dicha función de costo a través del tiempo. En la etapa de generalización, la red neuronal opera de manera autónoma sin la necesidad de actualizar

sus hiper-parámetros, normalmente se utiliza un conjunto de datos distinto al usado en la etapa de entrenamiento para validar que la red neuronal realice su tarea correctamente y no solamente haya memorizado el conjunto de datos utilizados en la etapa de entrenamiento.

2.2.2. Leyes de Aprendizaje

Como se describió en la sección anterior, el entrenamiento de la red neuronal permite aproximar funciones lineales y no lineales, este proceso se encuentra basado en leyes o ecuaciones de aprendizaje para minimizar una función de costo y es denominado como paradigma de aprendizaje de manera supervisada, [26]. En este tipo de aprendizaje se tiene un conjunto de datos de las señales adquiridas de la entrada y salida del sistema dinámico no lineal y se le conoce como "maestro", con ello se pretende determinar los valores de los pesos sinápticos de la red neuronal a partir de la información disponible. Algunas tareas comunes para este esquema de aprendizaje son: reconocimiento de patrones, regresión, pronóstico de series de tiempo, entre otras. Normalmente la función de costo $J(k)$ a minimizar es el error cuadrático medio (MSE) y se define como:

$$J(k) = \sum_{i=0}^{n} \frac{1}{2} E_i^2. \tag{2.6}$$

La ley de aprendizaje más comúnmente usada para la actualización de los pesos sinápticos es el conocido Retro-Propagación hacia atrás (BP), o también conocido como *Backpropagation*, continuando con el ejemplo de la sub-sección anterior, se tiene que las ecuaciones que definen el aprendizaje son, [27]:

$$V_{jo}(k+1) = V_{jo}(k) - \eta \frac{\partial J(k)}{\partial V_{jo}(k)}; \tag{2.7}$$

$$W_{ij}(k+1) = W_{jo}(k) - \eta \frac{\partial J(k)}{\partial W_{ij}(k)}, \tag{2.8}$$

donde el gradiente de la capa de salida es:

$$\frac{\partial J(k)}{\partial V_{jo}(k)} = \frac{\partial J(k)}{\partial \hat{y}_o(k)} \frac{\partial \hat{y}_o(k)}{\partial netho_o(k)} \frac{\partial netho_o(k)}{\partial V_{jo}(k)}.$$

Cabe mencionar que del gradiente $\frac{\partial J(k)}{\partial V_{jo}(k)}$ se obtienen los gradiente locales, estos serán usados para actualizar los pesos de la oculta:

$$\delta_{o_o}(k) = \frac{\partial J(k)}{\partial \hat{y}_o(k)} \frac{\partial \hat{y}_o(k)}{\partial netho_o(k)}.$$

El gradiente de la capa oculta está definido por:

$$\frac{\partial J(k)}{\partial W_{ij}(k)} = \left(\sum_{o=1}^{no} \delta_{o_o}(k) V_{jo}(k) \right) \Gamma'(neth_j(k) \cdot x_i(k), \tag{2.9}$$

donde $\Gamma'(\cdot)$ es la derivada de la función de activación.

Si existieran más capas ocultas cuyo número es denotado por el superíndice l el gradiente local quedaría definido por:

$$\delta^l_{h_j}(k) = \left(\sum_{o=1}^{no} \delta_{o_o}(k) V_{jo}(k) \right) \Gamma'_l(neth^l_j(k)), \tag{2.10}$$

sustituyendo la ecuación (2.10) en (2.9) se obtiene que el gradiente de la función de costo J con respecto a los pesos sinápticos de la matriz W_{ij} es:

$$\frac{\partial J(k)}{\partial W_{ij}(k)} = \delta^l_{h_j}(k) x_i(k). \tag{2.11}$$

La ley de aprendizaje de *BP* hacia atrás tiene ciertas limitantes, como lo puede ser el tiempo de convergencia debido al parámetro de aprendizaje η, la cual es una constante positiva $0 < \eta < 1$. Se han hecho esfuerzos a través del tiempo para modificar y dotar de ciertas cualidades a este método. El término denominado momentum, [27] se agrega para incrementar la velocidad de convergencia y evitar caer en un mínimo local superficial,

y beneficiara que los pesos sinápticos tiendan hacia los mínimos locales más profundos obteniendo un mejor desempeño en las etapas de generalización del modelo neuronal. La ley de aprendizaje basada en *BP* con termino momentum es:

$$V_{k+1} = V_k - \eta \frac{\partial J}{\partial V_{jo}} + \alpha \Delta V_k;$$
$$W_{k+1} = W_k - \eta \frac{\partial J}{\partial W_{ij}} + \alpha \Delta W_k, \tag{2.12}$$

donde $\Delta V_k = V_k - V_{k-1}$, $\Delta W_k = W_k - W_{k-1}$, α es una constante positiva, $0 < \alpha < 1$.

En ocasiones, un coeficiente no adecuado del termino α puede generar oscilaciones e inestabilidad en la etapa de entrenamiento. Además, las modificaciones tratan de evitar los problemas del gradiente descendente, sin embargo, esto no está resuelto completamente o se obtienen bajos desempeños, la secciones siguientes se aborda a mayor detalle estás problemáticas.

2.2.3. Esquemas de identificación para el modelado de sistemas dinámicos no lineales por redes neuronales

Sea un sistema dinámico no lineal discreto del que se desconoce un modelo matemático, este puede ser descrito por el siguiente modelo:

$$\bar{x}(k+1) = f[\bar{x}(k), u(k)],\ y(k) = g[\bar{x}(k)], \tag{2.13}$$

donde la entrada entrada del sistema es el vector $u(k)$, el vector de estados internos es $\bar{x}(k)$ y la salida del sistema es el vector $y(k)$. $f(\cdot)$ y $g(\cdot)$ son funciones no lineales suaves desconocidas, $f, g \in C^{\infty}$.

La adquisición de eventos a través del tiempo de las señales de entrada y salida permiten definir los siguientes conjuntos de entrada y salida, si y sólo si la entrada aplicada al sistema no lineal es conocida $u(k)$ y la salida del sistema dinámico no lineal puede ser adquirida y

discretizada a través de un sistema de adquisición de datos:

$$U(k) = [u(k), u(k-1), \cdots, u(k-n)];$$
$$\bar{X}(k) = [\bar{x}(k), \bar{x}(k-1), \cdots, \bar{x}(k-n)];$$
$$Y(k) = [y(k), y(k-1), \cdots, y(k-n)].$$

Si el gradiente $\frac{\partial Y(k)}{\partial \bar{X}(k)}$ es no singular evaluado alrededor del punto $[\bar{X}, U(k)] = [0; 0]$, es posible definir el modelo exógeno auto-regresivo no lineal ($NARX$) [15]:

$$y(k) = \hbar(X(k)), \tag{2.14}$$

donde $\hbar(\cdot)$ denota la ecuación no lineal en diferencias que describe la dinámica del sistema dinámico no lineal en función de los eventos adquiridos de la entrada y la salida:

$$X(k) = [y(k-1), y(k-2, \cdots, y(k-m_y)), u(k), u(k-1), \cdots, u(k-m_u))]^T, \tag{2.15}$$

por lo que $X(k) = [X_1, \cdots, X_l]$, con $l = m_y + m_u + 1$, y el número de datos total adquiridos del sistema es denotado por N donde el índice de referencia por evento es $k = 1, 2, \cdots, N$. Dado que la función $\hbar$ es desconocida es posible sustituirla por una red neuronal que a través del vector de entrada $X(k)$ y los hiper-parámetros (número de capas, número de nodos, constantes de aprendizajes, entradas) seleccionados es posible aproximar con $\hat{y}$ la dinámica producida por el sistema dinámico no lineal $y(k)$ ante una entrada conocida $u(k)$:

$$\hat{y}(k) = NN(X(k)), \tag{2.16}$$

donde NN representa una red neuronal artificial.

Para la identificación del sistema descrito por el modelo (2.13), se han propuesto los siguientes esquemas:

- Modelo Serie-Paralelo (ver Fig. 2.3a).

Se encuentra definido por el siguiente modelo:

$$\hat{y}(k) = NN[y(k-1), y(k-2), \cdots, y(k-n_y), u(k), u(k-1), \cdots, u(k-n_u)], \quad (2.17)$$

donde el vector de entrada $X(k)$ de la red neuronal NN es un modelo $NARX$. Tal que, n_u y n_y son los ordenes de regresión para entrada y la salida respectivamente. $n_y \neq m_y$ y $n_u \neq m_u$, siendo el esquema de identificación más utilizado ya que en la mayoría de los casos se conoce tanto la entrada como la salida.

- Modelo Paralelo (ver Fig. 2.3b),
  Se encuentra definido por el siguiente modelo:

$$\hat{y}(k) = NN[\hat{y}(k-1), \hat{y}(k-2), \cdots, \hat{y}(k-n_y), u(k), u(k-1), \cdots, u(k-n_u)]; \quad (2.18)$$

ó

$$\hat{y}(k) = NN[u(k), u(k-1), \cdots, u(k-n_u)], \quad (2.19)$$

donde la principal diferencia entre el Modelo Serie-Paralelo, el cual usa la regresión de la salida del sistema dinámico no lineal $y(k)$, consiste en que el Modelo Paralelo no usa la salida del sistema, en cambio, puede o no usar la regresión de la salida estimada de la red neuronal $\hat{y}(k-1)$.

Ambos modelos de identificación pueden ser usados también para una tarea de predicción, sin embargo, en general las estimaciones que entregan suelen no ser buenas cuando el sistema dinámico no lineal presenta problemas consistentes como lo es ruido en la adquisición de datos, pérdida de información, bases de datos insuficientes o la complejidad asociada a la naturaleza del sistema. Cabe mencionar que el esquema de identificación en paralelo suele ser difícil de implementar ya que únicamente se conoce información sobre la entrada del sistema y las predicciones realizadas por el modelo neuronal tienden a divergir con respecto a la señal de prueba ocasionado que el error de estimación aumente, [3].

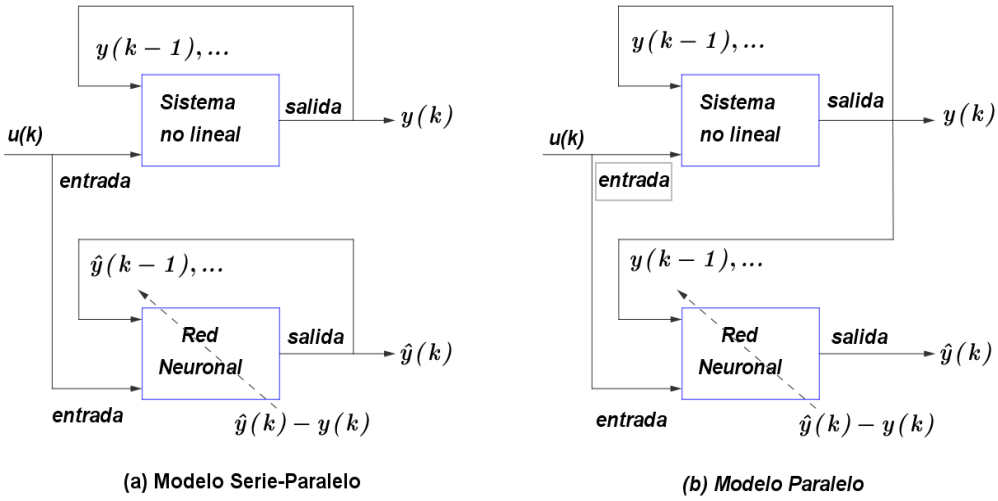


Figura 2.3: a) Modelo Serie-Paralelo. b) Modelo Paralelo.

2.3. Aspectos en la implementación de las redes neuronales

Con frecuencia el ámbito de investigación teórica crece de una manera desmesurada en comparación con la aplicación de los métodos, técnicas, herramientas y métodos propuestos, esto genera un desequilibrio evidente no sólo en la mayoría de las áreas del conocimiento. Este fenómeno es apreciable con facilidad en las redes neuronales artificiales, a través de la incesante proposición de nuevas topologías o estructuras cada vez más complicadas y con furor matemático extravagante dejan de lado algo esencialmente crítico, los medios de implementación y la clara definición de los problemas que se busca resolver. Se debería de considerar con mayor fuerza evaluar los límites de los métodos y no dejar de lado que la plasticidad que puede llegar a tener una RNA. Estos análisis permitirán proponer soluciones que mejoren las características de un modelo neuronal tales como una MLP utilizando una ley de aprendizaje como el BP.

2.3.1. Aprendizaje: Problemas y limitantes

El aprendizaje de una RNA, comúnmente se encuentra basado en el minimización de una función de costo, dicha acción se realiza a través de algoritmos matemáticos para el entrenamiento de los pesos sinápticos, a menudo se usan métodos como la rBP el cual usa el gradiente de la función de costo con respecto a los pesos sinápticos y los ajusta a lo largo de la dirección contraria del gradiente para reducir el error de aprendizaje. Los paradigmas de aprendizaje se clasifican en función de la información que adquieren a partir de la experiencia, estos pueden ser:

- Aprendizaje supervisado: Es aquél que contiene un algoritmo capaz de estudiar un conjunto de datos adquiridos de un fenómeno físico partir de un maestro y puede llegar a tener objetivos de clasificación y regresión. Involucra procesar diferentes valores de un vector de entrada dado $X(k)$, la RNA extrae ciertas características de la serie de tiempo y arroja un valor de salida $\hat{Y}(k)$ el cual es comparado con un valor deseado conocido $Y(k)$ para producir un error y posteriormente realizar la acción de entrenamiento; buscando también minimizar la función de costo del error entre la respuesta de la red y del sistema, [28].
- Aprendizaje no supervisado: En este aprendizaje se aprenden patrones de un conjunto de datos de entrada que no son conocidos (datos no etiquetados) y sin ningún lazo de realimentación explícito. Por ello, este algoritmo necesita aprender del conjunto de datos sin ninguna guía y sólo haciendo uso de las propiedades y características que contiene la serie de tiempo. Se aplican cuando se requiere aprender la distribución de probabilidad que el conjunto de datos genera, [27].

El teorema de aproximación universal [29], menciona que es posible aproximar cualquier función no lineal a partir de una sola capa oculta, sin embargo, puede no lograrse. Lo anterior se debe a problemas en los algoritmos de aprendizaje, uno de los algoritmos más usados y conocidos es BP el cual presenta dos problemáticas en particular: la falta de datos y el mínimo local. El segundo puede ser ocasionado no sólo por la falta de información, sino por otros aspectos relacionados a las condiciones iniciales del entrenamiento y los

hiper-parámetros seleccionados para topología de una RNA tales como: número de capas ocultas, número de entradas, nodos por capa oculta y constantes de aprendizaje. El hecho de que los pesos sinápticos no puedan converger a un mínimo global garantiza que exista en todo momento un error de estimación. El problema con el mínimo global es que existe, pero no es probable conocerlo. En [30], se aborda el problema de los mínimos locales en el aprendizaje a través de BP, y se hacen una serie de recomendaciones que permiten determinar las condiciones que se deben de cumplir para acercarse al mínimo global.

La RNA denotada por $\mathcal{NN}$, en su arquitectura MLP las neuronas o pesos sinápticos son agrupados en conjuntos llamados capas, mientras que el entorno de aprendizaje denominado $\mathcal{L}_e$ es un conjunto de datos para el aprendizaje supervisado y es una colección de pares de entrada/salida , los cuales serán usados por la $\mathcal{NN}$, $\mathcal{L}_e = ((X_o(t), D(t)), X_o \in \Re^{n(o)}, D(t) \in \Re^{n(l)}, t = 1, ..., T)$ donde, $(X_o(t)$ son los datos de entrada y $D(t)$ son los datos objetivo, cuyas dimensiones $n(o)$ y $n(l)$ son definidas por el usuario al realizar la etapa de adquisición de datos. La búsqueda del mínimo local se trata que a partir de un $\mathcal{L}_e$ se proponga un arquitectura de la $\mathcal{NN}$ que a través del algoritmo de aprendizaje se encuentre un conjunto de pesos tales que la función de costo E_T, como lo puede ser el (2.6), sea minimizada. Para intentar que el gradiente descendente converja al mínimo global ($E_T \longrightarrow 0$) se enlistan algunas recomendaciones a continuación [30]:

- $\mathcal{NN}$ sea piramidal, $n(l+1) \leq n(l), l = 1, ..., L-1$.
- Las matrices de pesos por cada capa, $W_l, l = 1, ..., L-1$ sea de rango completo por filas.
- La relación del kernel de matriz X_o' y el kernel de la salida del sistema denotado por S_l^Y:

$$Ker[X_o^{'}] \cap S_l^Y = \{\oslash\}\,.$$

Lo anterior ha sido probado de forma matemática, sin embargo, en las situaciones del mundo real, es muy difícil cumplir con la tercera proposición, ya que eso significaría tener el conocimiento de S_l^Y, i.e. el conjunto de todas las salidas $\hat{Y}(t)$ que genera la variación de los

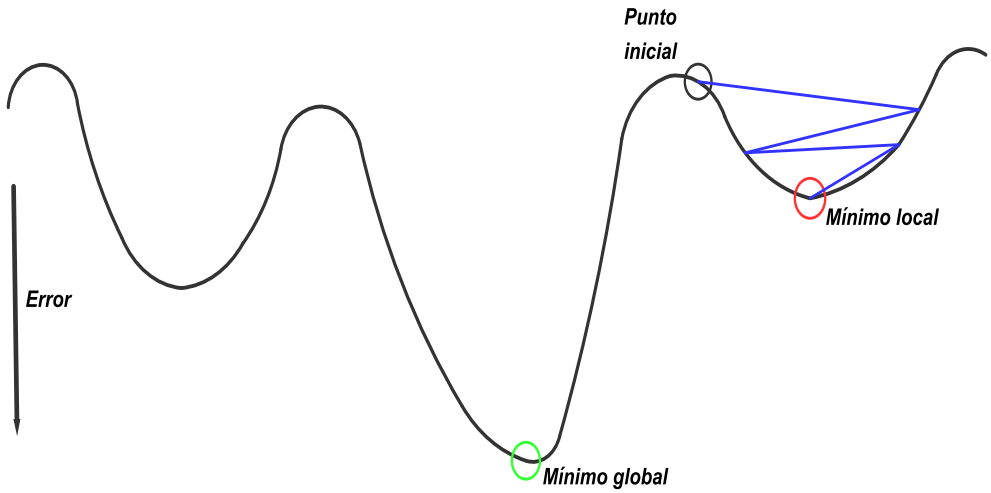


Figura 2.4: Gradiente Descendente.

pesos W_i en el compacto Ω. Esto significa que el rango la matriz de los datos de entrada sea de rango completo tal que $Rank[X_o] = T$ y por tanto que sean linealmente independientes por fila o columna. Si bien la formalidad matemática permite definir los caminos y propiedades que se deben de cumplir para lograr encontrar $E_T \longrightarrow 0$, en la práctica esto no se cumple con facilidad debido, por ejemplo, a la información disponible o el grado de dependencia de las variables a tratar en un sistema. La generación de un mínimo local es casi inevitable en función de los hiper-parámetros seleccionados, ver Figura (2.4).

En trabajos recientes también se aborda el problema de la selección de las condiciones iniciales para los pesos se ha vuelto un tema popular para las áreas de optimización, ya que en efecto la convergencia también depende del azar para la selección de estos. En [31], se expone un método para calcular las ponderaciones iniciales del MLP se basa en la calidad de la medida de similitud propuesta en el marco de la teoría ampliada de conjuntos aproximados. Por otro lado, la información disponible para tratar un problema de predicción o control a través de un RNA, en muchas maneras puede ser interpretado desde la dependencia que existe entre la entrada y la salida, hasta la falta de valores en el conjunto de datos debido a

una mala adquisición de datos. Es común que las series de tiempo sufran pérdida de datos, de una manera no muy elegante los operadores de redes neuronales borran los elementos desconocidos y usan elementos disponibles de eventos pasados asociados a los históricos de las variables de estado, esto suele funcionar cuando la serie de tiempo tiene características de estacionalidad o periodicidad, pero cuando se tratan series de tiempo caóticas entonces las propiedades y características cambian sin un patrón aparente y los métodos tradicionales dejan de surtir efecto y lograr los objetivos propuestos se vuelve cada vez más complicado. Un caso de estudio es el mostrado en [32], donde se busca mitigar la consecuencia de los gases de efecto invernadero, para ello se requieren modelos neuronales que permitan la predicción de forma precisa de la cantidad y tiempo de recuperación del gas metano en los vertederos. Cuando el estudio se asocia a problemas de emisiones de gases de efecto invernadero, los modelos de predicción de series de tiempo son de considerable interés, en los que se requieren registros pasados significativos en bases de datos para cada uno de gases en cuestión. Se realizó un modelo de series de tiempo que consta de dos etapas: la primera la primera se realiza mediante la imputación de datos faltantes en la serie de tiempo, seguido de una aproximación basada en un modelo $NARX$. Esto es un claro ejemplo de que no hay estudios o análisis que permitan determinar, si tomar información del pasado y utilizarla para completar la serie de tiempo es correcto, ya que las características espacio-temporales del fenómeno pueden generar un sesgo en el entrenamiento de la RNA y por tanto en la etapa de validación el error de estimación no sea aceptable.

2.4. Enfoques modernos en el aprendizaje automático

Los métodos de aprendizaje automático adaptan los parámetros de un modelo, esto se realiza para una determinada tarea y el re-entrenamiento es necesario cuando cambian las condiciones del sistema. El error de estimación se debe a la falta de datos y tiempo de recolección de los mismos debido a características de espacio-tiempo. Sin embargo, generalmente es difícil extraer la experiencia y convertirlo en un sesgo inductivo, esto ocurre de manera particular con ciertas clases de modelos de caja negra como las redes neuronales.

El ML proporciona una forma de automatizar la selección de un sesgo inductivo para una tarea principal, de tal manera que aprovecha las observaciones hechas a los datos provenientes de tareas secundarias que guarden una relación de similitud. Con sesgo un inductivo entrenado de esta manera el modelo de aprendizaje automático puede llevarse a cabo de manera potencial con una menor cantidad de datos de entrenamiento y/o una mayor complejidad [33].

Las tecnologías basadas en minería de datos y aprendizaje automático estocásticos han tenido un éxito significativo en conocimientos en áreas de ingeniería que incluyen clasificación, regresión y agrupamiento. Sin embargo, los métodos de aprendizaje funcionan bien sólo bajo una suposición común: los datos de entrenamiento y prueba se extraen del mismo conjunto que cuentan con una misma distribución probabilística. Cuando lo anterior cambia, la mayoría de los modelos estadísticos deben reconstruirse utilizando datos de entrenamiento recién recopilados. En muchos casos reales de aplicaciones, es caro o imposible de recopilar los datos de entrenamiento necesarios. Por lo tanto, un interés de investigación es buscar reducir la dependencia de los datos disponibles. En tales casos, la TL es deseable, [5].

Finalmente, es imperante manifestar que no existe un consenso en el uso de los términos de Meta-Aprendizaje y Transferencia-Aprendizaje, si bien son conceptos que aparecen de manera simultánea o con escasos años de separación entre los primeros trabajos, no es posible definir o establecer diferencias entre ambos. Sin embargo, para fines de este trabajo el Meta-Aprendizaje hace uso de los datos dentro de su propio dominio, mientras que el Transferencia-Aprendizaje puede usar conocimiento de un dominio distinto.

2.4.1. Meta-Aprendizaje

El concepto de Meta-Aprendizaje (ML), data de hace más de 35 años en la década de los 80's para aplicaciones en el campo del aprendizaje automático. Su raíz proviene de trabajos dedicados al campo de la psicología, ya que se afirma que la dinámica o comportamiento humano deben entenderse a través de los cambios efectuados debido al enfrentamiento a múltiples problemas del aprendizaje, los cuales tienen un cierto grado de similitud, [34]. En

otra dirección dentro del mismo campo, se encuentra el enfoque educativo [35], donde se menciona que los estudiantes deben ser conscientes no sólo de lo que una tarea o actividad ofrece en primer plano como lo puede ser la resolución de la misma, sino en un sentido profundo sobre el entendimiento del control de sus propios recursos cognitivos; dentro del aprendizaje automático estos pueden interpretarse como un modelo que gana experiencia a través de múltiples escenarios de aprendizaje con una cierta distribución probabilidad de los datos asociados a cada una tareas, que a su vez utiliza esa información o conocimiento disponible para maximizar el desempeño de aprendizaje en tareas futuras. Un término asociado de forma recurrente es "aprender a aprender"mencionado en [36], donde los beneficios obtenidos tales como la eficiencia y discriminación de la información relevante en los datos y su análisis en escalas de tiempo están mejor alineadas hacia el aprendizaje humano ya que esto se da de manera evolutiva a lo largo de la vida, [37].

Los conceptos antes mencionados sobre ML aplicados a métodos, algoritmos dentro del aprendizaje automático aparecen por primera vez en [38], donde se exponen una serie de métodos para que las redes neuronales realicen un aprendizaje auto-referencial al recibir como entradas sus propios pesos sinápticos y con ello predecir actualizaciones para los mismos, esto puede denominarse como el uso de algoritmos evolutivos. Posteriormente, se han observado diferentes áreas cómo lo son algoritmos bio-inspirados. En [39],[40], propusieron reglas de aprendizaje basados en ML biológico, avances en sistemas auto-referenciales [41],[42], hasta que finalmente se comenzó a ocupar el ML con el gradiente descendente en 1991 [43], con diversos esfuerzos por coordinar ambos conceptos a lo largo de trabajos como [44], [45], mientras que en [46] se dio una descripción general y condensada de la literatura propuesta hasta ese momento.

En [4], se realiza un increíble esfuerzo por unificar el concepto en nuestros días, ya que a pesar de que el concepto es relativamente novedoso no ha sido sencillo de lograr una convención del concepto de ML en las áreas del conocimiento. En el trabajo muestran una taxonomía del concepto que parten de las siguientes preguntas:

- Meta-Representación ("¿Qué?") Se basa en la elección del Meta-Conocimiento para entonces Meta-Aprender. Esto podría ser desde los parámetros iniciales del sistema

basado en ML [47], hasta la optimización de código basado en el aprendizaje por refuerzo [48].

- Meta-Optimizador ("¿Cómo?") Tiene que ver con el tipo de aprendizaje y ley de aprendizaje a utilizar, cómo lo puede ser el gradiente descendente [47], aprendizaje por refuerzo [48] y algoritmos evolutivos [49].
- Meta-Objetivo ("¿Por qué?") Se determina hacia donde se orienta las propiedades y ventajas de las técnicas dentro del campo, como aprendizaje en pocas iteraciones [50], optimización rápida [51], robustez dentro del cambio de dominio entre tareas [52] y ruido en los datos [53].

A continuación se muestra un diagrama en la Figura (2.5) que permite ubicar a este trabajo dentro de la taxonomía propuesta como un Meta-Optimizador, además de los campos de aplicación estudiados hasta el momento, donde es visible observar que la mayoría de los esfuerzos han sido encaminados hacia las ciencias de la computación y dentro del área del control automático no existen indicios o trabajos donde se pueda apreciar el uso de esta técnica.

Dentro del área del Meta-Optimizador existen diversas subáreas en las que se clasifica el aprendizaje: gradiente descendente, aprendizaje por refuerzo y algoritmos evolutivos. Dada la naturaleza del trabajo, el enfoque es hacia el primero de los antes mencionados.

Existen variantes del gradiente descendente con la línea de ML, [54], [55], [56]. En la literatura se menciona que son algoritmos potencialmente más eficientes ya que están basados en cuestiones analíticas/deterministas [4]. Por otra parte, tienen sus propios retos a superar cómo lo son:

- Depurar los pasos de optimización internamente a través de algoritmos de diferenciación tales como [57], [58], o algoritmos implícitos de diferenciación [59], [60]; así cómo los términos de segundo orden debidos al gradiente [61].
- Reducir los inevitables problemas de degradación del gradiente descendente, cuya efecto aumenta con el número de pasos de optimización del ciclo interno.
- Cálculo de gradientes, cuando la tarea incluye operaciones discretas y no diferenciables.

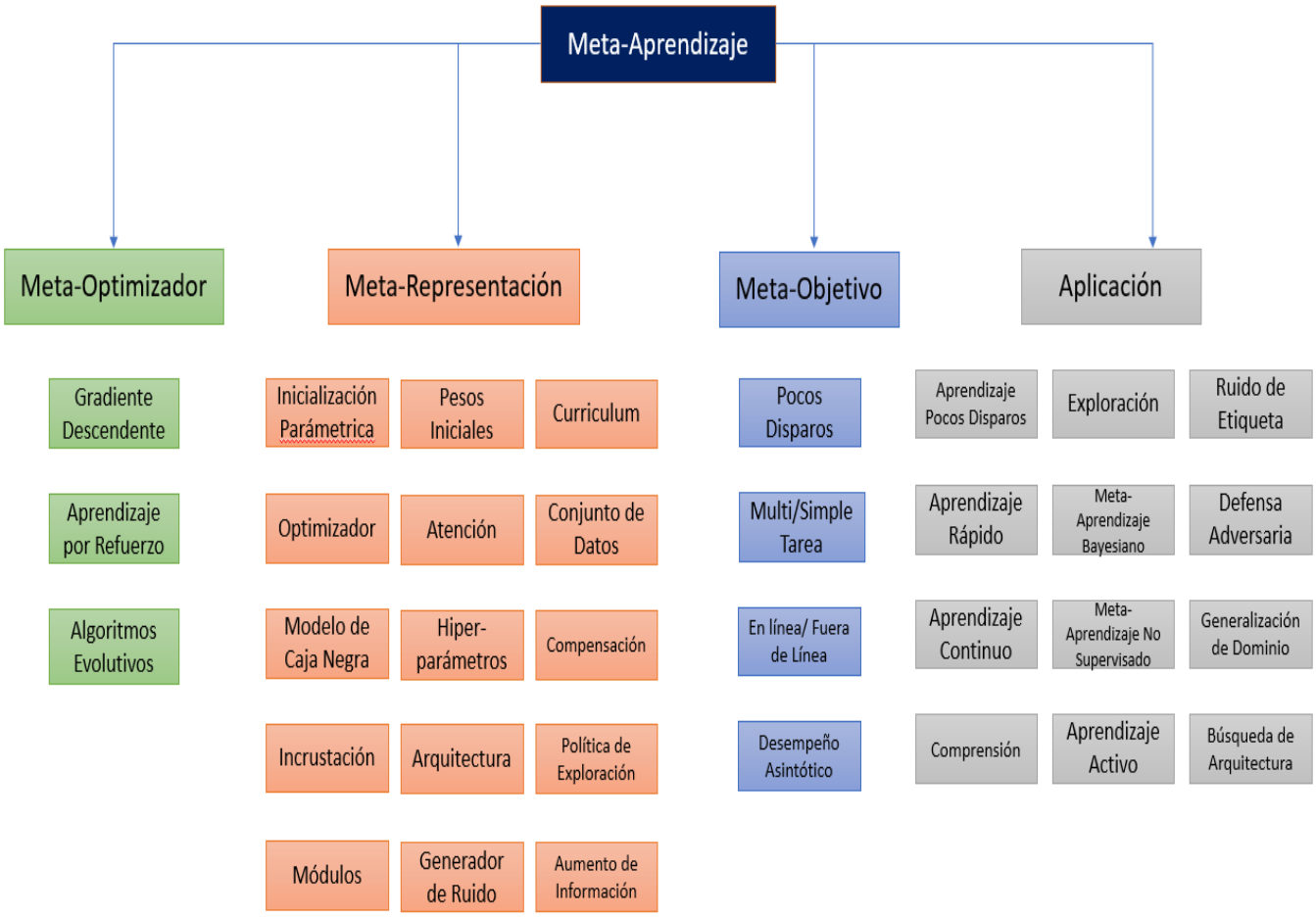


Figura 2.5: Taxonomía del Meta-Aprendizaje.

2.4.2. Transferencia-Aprendizaje

Cuando se habla de problemas o aplicaciones del mundo real las propiedades, características o elementos que conforman dicha situación de ninguna manera son ideales, además es caro o imposible de recabar y almacenar todos los datos para entrenamiento de una RNA necesarios para reconstruir un modelo de caja negra. Es por ello, que ante situaciones en los que la cantidad de información es insuficiente para evadir problemas de mínimos locales a través del uso del método de ML; es necesario recurrir a nuevas ideas que permitan resolver el problema, para tales casos la transferencia de conocimientos o el Aprendizaje por Transferencia (TL) entre dominios de tareas sería deseable, [5]. Un ejemplo de como se ha utilizado la técnica de TL es el presentado en [62], donde un modelo neuronal es entrenado por un periodo de tiempo, el cual aprende de la ubicación espacial de un dispositivo, en ocasiones se necesita recalibrar debido a los cambios de ubicación determinado por el objetivo de la tarea, es posible adaptar el modelo de localización en un período de tiempo (la fuente dominio) para un nuevo período de tiempo (el dominio de destino), o para adaptar el modelo de localización entrenado en un dispositivo móvil (el dominio de origen) para un nuevo dispositivo móvil (el destino dominio).

La Figura (2.6) muestra la diferencia entre los procesos de aprendizaje de técnicas de aprendizaje tradicionales y del método de TL. Como se sabe, las técnicas tradicionales de aprendizaje automático intentan aprender cada tarea desde cero, mientras que las técnicas de aprendizaje por transferencia intentan transferir el conocimiento de algunas tareas anteriores a un objetivo tarea cuando este último tiene menos datos de entrenamiento de alta calidad considerando que el dominio de entrenamiento puede ser similar o no.

Cuando se habla de la etapa de entrenamiento tradicional los datos utilizados, en la mayoría de los casos se sabe que la distribución es la misma para los datos etiquetados y no etiquetados. La transferencia de aprendizaje, por el contrario, permite que los dominios de las tareas y sus distribuciones utilizadas en el entrenamiento y las pruebas puedan ser diferentes. En el mundo real, se pueden encontrar ejemplos de TL. Por ejemplo, es probable que aprender a reconocer naranjas puede ayudar a reconocer tomates. Similar, es el aprender a sintonizar un controlador PID en tiempo continuo y con esto se puede facilitar aprender

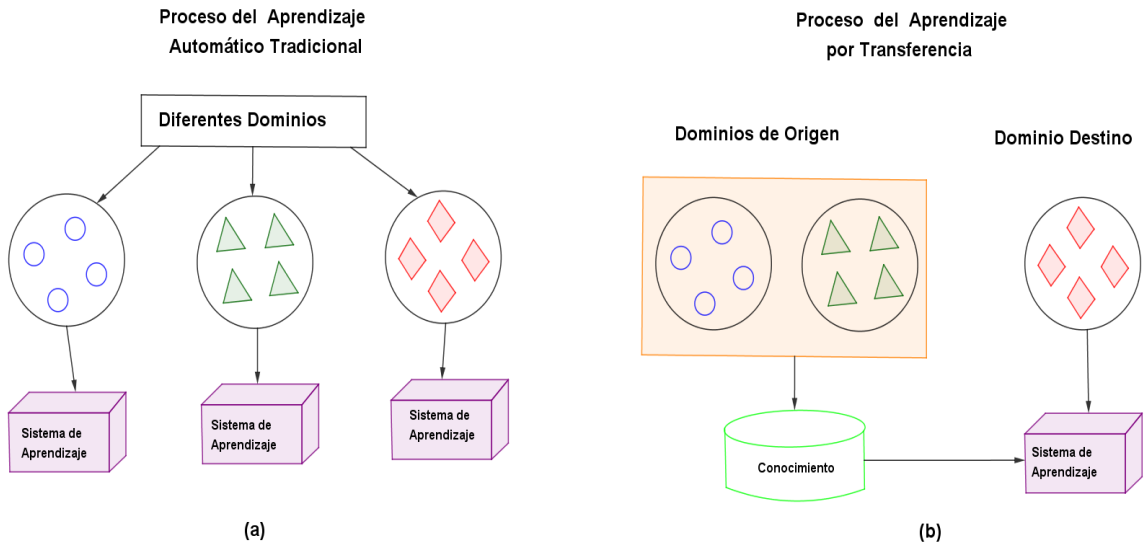


Figura 2.6: Aprendizaje por Transferencia.

a sintonizar un PID en tiempo discreto. El estudio del aprendizaje por transferencia está motivado por el hecho de que las personas pueden aplicar inteligentemente el conocimiento aprendido previamente a resolver nuevos problemas más rápido o con mejores soluciones.

La motivación fundamental para el método TL, tuvo lugar en el congreso de NIPS-95 sobre "Aprender a Aprender", el cual se centró sobre en la necesidad de métodos de aprendizaje automático de por vida que retengan y reutilicen el conocimiento aprendido previamente. Al igual que el ML, el TL responde a las siguientes preguntas:

- "¿Qué transferir?": Pregunta qué parte de asumir que conocimiento es posible transferir a través de dominios o tareas. La información puede fluir entre dominios específicos o tareas individuales, y/o puede ser entre diferentes dominios de modo que pueden ayudar a mejorar el rendimiento del dominio de destino o tarea.
- "¿Cómo transferir?": Después de descubrir qué conocimientos se pueden transferir, es necesario desarrollar algoritmos de aprendizaje para transferir el conocimiento traducido como experiencia.

- '¿Cuándo transferir?": Esto debe ser contestado para identificar en que situaciones es conveniente hacer la transferencia de conocimiento y cuando no lo es. Ya que, cuando el dominio de origen y el dominio de destino no están relacionados entre sí, fuerza bruta es posible que la transferencia no se realice correctamente. En el peor de los casos se perjudica el rendimiento del aprendizaje en el objetivo de la tarea, esta situación a menudo se le denomina transferencia-negativa.

Los diferentes enfoques utilizados para el TL se pueden clasificar como:

- Transferencia de Instancia: Para escribir algunos datos etiquetados en el dominio de origen y usarlos en el dominio de destino [63], [64], [65], [66].
- Encontrar una buena representación de características que reduzca la diferencia entre los dominios de origen y el error de los modelos de clasificación y regresión [67], [68], [69], [70].
- Descubrir parámetros compartidos entre el dominio de origen y el dominio de destino puede beneficiarse de la transferencia de aprendizaje [71], [72], [73].
- Construcción de un mapeo de conocimiento relacional entre el dominio de origen y el dominio de destino [74], [75], [76].

En el entorno de aprendizaje por transferencia inductiva, la tarea objetivo es diferente de la tarea de origen sin importar cuándo los dominios de origen y destino son iguales o no. En este caso, algunos datos etiquetados del dominio de destino son necesarios para inducir una predicción objetiva. Además, según diferentes situaciones de etiquetado y datos sin etiquetar en el dominio de origen, se puede categorizar el entorno de aprendizaje de transferencia inductiva en dos casos:

- Cuando los datos etiquetados en el dominio de origen están disponibles. Se establece que sólo se tiene como objetivo lograr un alto rendimiento en la tarea principal mediante la transferencia de conocimientos de la tarea de origen, mientras que el aprendizaje multitarea intenta aprender la tarea de origen y de destino simultáneamente.

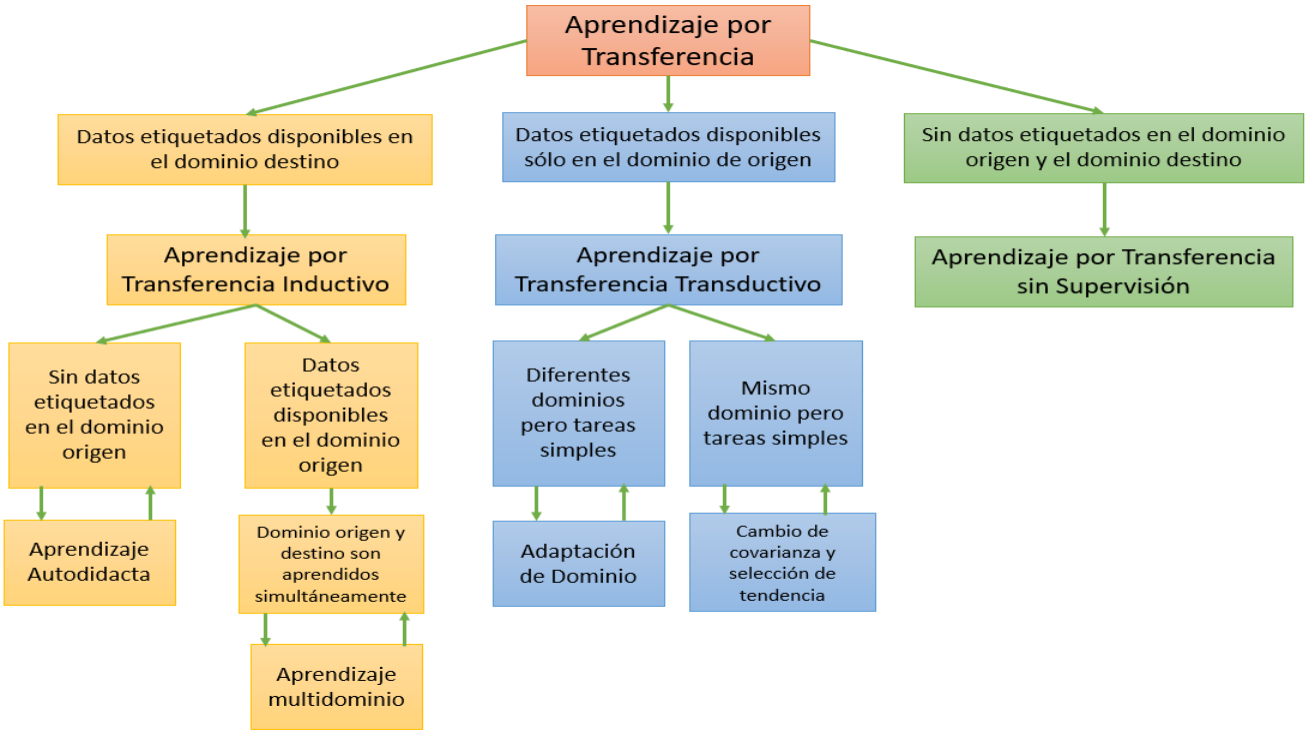


Figura 2.7: Taxonomía del Aprendizaje por Transferencia.

- No hay datos etiquetados en el dominio de origen disponible. En este caso, la transferencia inductiva tiene un entorno de aprendizaje que es similar al autodidacta. En [77], se menciona que el entorno de aprendizaje autodidacta puede tener etiquetas entre el origen y el destino, por lo que los dominios pueden ser diferentes, lo que implica que la información complementaria del dominio de origen no se puede utilizar directamente. La transferencia inductiva se da cuando los datos etiquetados del dominio de origen no están disponibles.

En la Figura (2.7), se muestra un mapa para comprender la taxonomía del TL.

Capítulo 3

Meta-Transferencia de Aprendizaje para redes neuronales

3.1. Identificación y predicción de sistemas dinámicos no lineales

La identificación y predicción de sistemas dinámicos no lineales a través de las *RNA* basado en un modelo *NARX*, depende de la disponibilidad el vector de la entrada del sistema $U(k) \in \mathbb{R}^n$ y el vector de la salida del sistema $Y(k) \in \mathbb{R}^n$ tal que:

$$U(k) = [u(k), u(k-1), \cdots, u(k-n)]; \tag{3.1}$$

$$Y(k) = [y(k), y(k-1), \cdots, y(k-n)], \tag{3.2}$$

donde el número de elementos disponibles debido a la adquisición de datos esta definido por el coeficiente n.

3.1.1. Predicción de corto plazo

El modelo de predicción neuronal a corto plazo para un sistema dinámico no lineal queda descrito por un modelo NARX:

$$\begin{aligned} y(k+1) = F[&y(k), y(k-1), \cdots, y(k-n), \\ &u(k), u(k-1), \cdots, u(k-n)], \end{aligned} \tag{3.3}$$

donde $F(\cdot) \in C^{\infty}$ es una función no lineal desconocida. Considérese el modelo neuronal serie-paralelo y para una única capa oculta:

$$\hat{y}(k+1) = \Gamma\left[W_k X(k)\right], \tag{3.4}$$

donde $\hat{y}(k) \in \mathbb{R}$, $W_k \in \mathbb{R}^n$ es la matriz de pesos de capa oculta, $\Gamma(\cdot)$ es la función de activación y el vector de entrada está definido por:

$$\begin{aligned} X(k) = [&y(k), y(k-1), \cdots, y(k-n_y^*) \\ &u(k), u(k-1), \cdots, u(k-n_u^*)] \in R^{n\times 1}, \end{aligned} \tag{3.5}$$

donde n_y^* y n_u^*, son el número de eventos adquiridos seleccionados aleatoriamente para la representación del sistema. Para una red neuronal multicapa, el modelo de predicción está definido cómo:

$$\hat{y}(k+1) = V_k \Gamma\left[W_k X(k)\right], \tag{3.6}$$

donde $W_k \in R^{m\times n}$ es la matriz de pesos de la capa oculta y $V_k \in R^{1\times m}$ es la matriz de pesos de la capa de salida.

Finalmente, para un modelo de aprendizaje profundo se tiene que:

$$\hat{y}(k+1) = V_k \Gamma\left\{W_l \Gamma_l\left[\cdots W_1 X(k)\right]\right\}, \tag{3.7}$$

donde l es el número de capas ocultas, W_l son las matrices de pesos sinápticos asociados a cada capa oculta l y $\Gamma(\cdot)$ es la función de activación.

3.1.2. Predicción de largo plazo

La predicción en su forma de multi-horizonte se encuentra definida por la siguiente ecuación:

$$y(k+1+\xi_\sigma) = F[y(k-\xi_{\alpha y}), u(k-\xi_{\alpha u})], \tag{3.8}$$

donde ξ_σ es el coeficiente de predicción, $\xi_{\alpha y}$ es el coeficiente de atraso de la salida del sistema y $\xi_{\alpha u}$ es el coeficiente de atraso de la salida del sistema. Entonces:

$$\xi_\sigma = \{0, 1, 2, ..., n_\sigma\}, \xi_{\alpha y} = \{1, 2, ..., n_{\alpha y}\}, \xi_{\alpha u} = \{1, 2, ..., n_{\alpha u}\}, \tag{3.9}$$

donde n_σ es el número máximo de predicciones, $\eta_{\alpha y}$ es el número máximo de eventos considerados de la salida del sistema y $\eta_{\alpha u}$ es el número máximo de eventos considerados de la entrada del sistema no lineal. De tal manera que:

$$[y(k+n_\sigma), ..., y(k+1)] = F[y(k-\eta_{\alpha y}), u(k-\eta_{\alpha u})]. \tag{3.10}$$

Se define el vector de entrada como:

$$X_\sigma(k) = [y(k-\eta_{\alpha y}), u(k-\eta_{\alpha u})], \tag{3.11}$$

por lo que el sistema dinámico no lineal descrito por $y(k+1+\xi_\sigma)$ es:

$$y(k+1+\xi_\sigma) = F[X_\sigma(k)]. \tag{3.12}$$

Aplicando el modelo neuronal NN serie-paralelo:

$$\hat{y}(k+1+\xi_\sigma) = NN[y(k-\eta_{\alpha y}), u(k-\eta_{\alpha u})], \tag{3.13}$$

o el modelo neuronal NN en paralelo dado por:

$$\hat{y}(k+1+\xi_\sigma) = NN[\hat{y}(k-\eta_{\alpha y}), u(k-\eta_{\alpha u})]. \tag{3.14}$$

Para los modelos neuronales multi-capa y aprendizaje profundo se sigue el mismo camino que la sección anterior.

3.2. Meta-Aprendizaje y redes neuronales multicapa

Considérese las problemáticas expuestas en la sección (2.3). Un método alternativo para superar las limitaciones de un algoritmo de aprendizaje BP aplicado en un MLP, es el método de Meta-Aprendizaje, el cual aprovecha la experiencia adquirida en su etapa de entrenamiento con hiper-parámetros definidos para una MLP y con ello mejorar el proceso de formación de conocimiento en un nuevo entrenamiento a partir de condiciones iniciales distintas. En la Figura (3.1), se muestra un ejemplo del funcionamiento del ML. Se requiere hacer una pronóstico semanalmente de una variable, el modelo neuronal extrae y usa el mejor conocimiento posible de cada semana disponible; esto representa un conjunto aleatorio de dimensiones arbitrarias con diferentes condiciones iniciales. El ML utiliza esta experiencia para mejorar el proceso de formación de conocimiento a partir de condiciones iniciales distintas para una misma tarea. Esto corresponde dentro del marco general a un meta-optimizador, ver sección (2.4), ya que se centra en el método para aprender a través de la modificación de la ley de aprendizaje.

Para mejorar la precisión del modelado de un sistema dinámico lineal a través de una MLP es necesario mejorar su etapa de entrenamiento, para este fin se propone el siguiente método de Meta-Aprendizaje. Esto es una modificación a la ley de aprendizaje BP descrita por la ecuación (2.12):

$$W_{k+1} = W_k - \eta \frac{\partial J}{\partial W} + \alpha \Delta W_k + \beta_{w,k} \hat{X}_{W,k} \tag{3.15}$$

$$V_{k+1} = V_k - \eta \frac{\partial J}{\partial V} + \alpha \Delta V_k + \beta_{v,k} \hat{X}_{V,k} \tag{3.16}$$

donde $\beta_{W,k}$, $\beta_{V,k} \in \mathbb{R}$ mientras que $\hat{X}_{W,k} \in \mathbb{R}^n$ y $\hat{X}_{V,k} \in \mathbb{R}^m$, estos términos pertenecen al ML, para determinarlos se propone el algoritmo (1).

La aplicación del método de ML se describe a continuación:

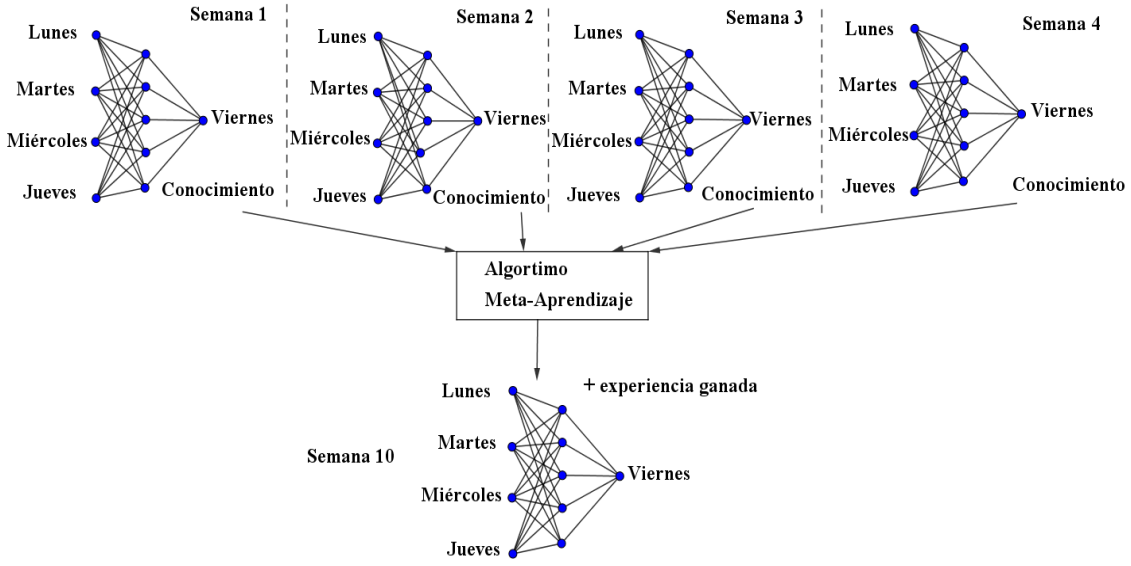


Figura 3.1: Ilustración del funcionamiento de ML.

Algoritmo 1 Búsqueda del vector $\hat{X}_{W,k}$

Require: Elegir una condición de ángulo $cos\theta_{W,i}$, como $0 \leq Ac \leq 1$

Seleccionar una condición inicial aleatoria para $\hat{X}_{W,0}, i = 1$

while do$(cos\theta_{W,i} \geq Ac)$

Seleccionar aleatoriamente un vector $\hat{X}_{W,i}$ tal que $\hat{X}_{W,i} \leq \hat{X}_{W,i-1} \leq 1$

Calcular $\left\| \hat{X}_{W,i}\left(k\right) \right\|$

Estimar $cos\theta_{W,i} = \frac{W^* - W_i}{\|W^* - W_i\|} \cdot \frac{\hat{X}_{W,i}}{\left\|\hat{X}_{W,i}\right\|}$ $i = i+1$

end while

1. Si se desea realizar pronósticos a corto o lago plazo, se entrena un modelo neuronal descrito previamente, i.e, (3.6) ó (3.13) considerando una capa oculta; con el algoritmo del gradiente descendente dado por la ecuación (2.12). Esto se hace un número de veces definido por el usuario utilizando diferentes pesos iniciales para, i.e., V_0 y W_0 de forma aleatoria en un intervalo de [-1,1].

2. A partir del paso anterior se seleccionan los pesos sub-óptimos, V^* y W^*, para esto se identifica aquel experimento en el que se logró un mejor desempeño al minimizar la función de costo (2.6). Está idea es consistente con los conceptos básicos de ML al extraer como experiencia propiedades o características de conocimientos previos adquiridos por medio de tareas alternas en forma de pesos sinápticos.

3. Para extraer el conocimiento se re-entrena el modelo neuronal seleccionado utilizando la ley de aprendizaje descrita por la ecuación (3.16). Donde el vector, i.e., $\hat{X}_{W,k} \in R^n$, hace que los pesos W_k converjan a W^*, de tal manera que la distancia entre ellos $\|W^* - W_i\|$ se minimice. Para cada iteración se tiene que cumplir la condición angular $cos\theta_{W,k}$, seleccionada de tal manera que $0 \leq Ac \leq 1$. El cálculo de los términos $\hat{X}_{W,k}$ y $\hat{X}_{V,k}$ es:

$$\begin{aligned} \hat{X}_{W,k} &= \max_i \left[\theta_{W,i}\right], \cos\left(\theta_{W,i}\right); \\ &= \frac{W^* - W_i}{\|W^* - W_i\|} \frac{\hat{X}_{W,k}}{\left\|\hat{X}_{W,k}\right\|}, \end{aligned} \tag{3.17}$$

$$\begin{aligned} \hat{X}_{V,k} &= \max_i \left[\theta_{V,i}\right], cos\left(\theta_{V,k}\right) \\ &= \frac{V^* - V_k}{\|V^* - V_k\|} \frac{\hat{X}_{V,k}}{\left\|\hat{X}_{V,k}\right\|}. \end{aligned} \tag{3.18}$$

El paso de aprendizaje $\beta_{W,k}$ reduce $\|W^* - W_i\|$:

$$\beta_{W,k} = (W^* - W_k) \frac{\hat{X}_{W,k}}{\left\|\hat{X}_{W,k}\right\|}. \tag{3.19}$$

Este término variable en el tiempo asegura que la condición angular (3.17) sea cumplida en cada paso. El termino $\beta_{V,k}$ reduce $\|V^* - V_i\|$, tal que:

$$\beta_{V,k} = (V^* - V_k) \frac{\hat{X}_{V,k}}{\left\|\hat{X}_{V,k}\right\|}.$$

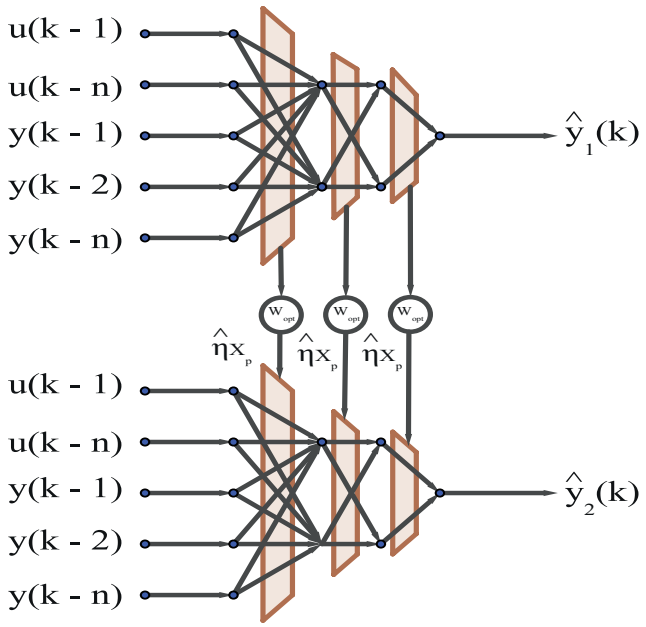


Figura 3.2: Extracción y transferencia de conocimiento entre MLP.

El término de ML ($\beta_{W,k}X_{W,k}$) reduce el error de modelado en cada paso k para el modelo neuronal propuesto y produce una convergencia rápida entre los pares (W_k, W^*) y (V_k, V^*), porque en los pesos W_k y V_k se proyectan a los pesos sub-óptimos W^* y V^*, es decir, los

pesos actuales W_k convergen al vector de pesos deseados W^* con la dirección $\hat{X}_{W,k}$ y debido al paso $\beta_{W,k}$, ver la Figura (3.2).

En la ecuación (3.16) se agrega el término llamado η_k ya que el modelo presentado en la ecuaciones (3.7) y (3.14) es débil en su uso en el pronóstico de series de tiempo caóticas. Debido a la naturaleza que presenta, podría haber problemas de inestabilidad en el proceso de entrenamiento, por lo que se muestra una prueba de estabilidad para garantizar un proceso de entrenamiento estable. El modelo en la ecuación (3.16) y su posterior modificación en la ecuación (3.36) podría acceder a mínimos locales más profundos debido al uso del conocimiento desde el término ($\beta_{W,k}$, $\hat{X}_{W,k}$) ya que es una proyección de los pesos del sistema en los pesos sinápticos sub-óptimos obtenidos anteriormente.

$$\begin{aligned} W_{k+1} &= W_k - \eta_k \tfrac{\partial J}{\partial W} + \alpha \Delta W_k + \beta_{w,k} \hat{X}_{W,k}; \\ V_{k+1} &= V_k - \eta_k \tfrac{\partial J}{\partial V} + \alpha \Delta V_k + \beta_{w,k} \hat{X}_{V,k}; \\ \eta_k &= \tfrac{\eta}{1+\|\phi' X^T(k)\|^2}, 0 < \eta_k < \eta \leq 1, \end{aligned} \tag{3.20}$$

donde η es la tasa de aprendizaje positiva $0 < \eta < 1$, $\frac{\partial J}{\partial W} = \phi' \left[W_k \hat{X}(k) \right] E(k)$ y $\Delta W_k = W_k - W_{k-1}$. Además, $\Delta V_k = V_k - V_{k-1}$, α es una constante $0 < \alpha < 1$ y $\beta_{w,k}$, $\hat{X}_{W,k}$ son los nuevos parámetros del Meta-Aprendizaje.

La modificación del método ML para el modelado neuronal se muestra en la Figura (3.3), donde se ilustra como el conocimiento es extraído por medio de los términos $\beta_{w,k}$, $\hat{X}_{W,k}$ de una MLP, para después ser utilizados por una MLP con las mismas características excepto las condiciones iniciales las cuales deberán de ser distintas.

Es posible utilizar un modelo serie-paralelo ó paralelo. Sin embargo resulta de mayor interés el impacto que puede llegar a tener para el último mencionado ya que aún en la actualidad está representación sigue planteando dificultades a los respectivos campos de investigación.

La principal limitante que se tiene que considerar en el uso del método de ML yace en la constante Ac perteneciente a la condición angular, ya que está ponderada puede generar un bucle infinito, debido a que el algoritmo no puede encontrar el vector $\beta_{W,k}$ que satisfaga la condición. La cantidad previa de información de la serie de tiempo debe ser suficiente para

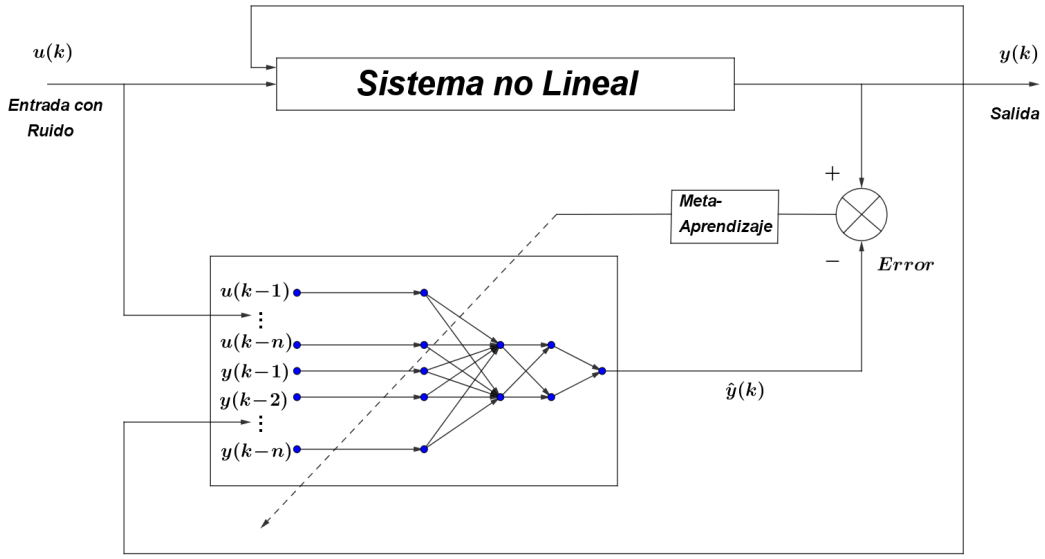


Figura 3.3: Meta-Aprendizaje para la identificación de sistemas en el esquema de simulación.

que el método funcione.

3.3. Transferencia-Aprendizaje y redes neuronales multicapa

Otro problemática importante a resolver es cuando los datos de entrenamiento no están completos. Es posible tomar datos de otros conjuntos de datos similares o afines a los de una tarea en cuestión. La propiedad fundamental del aprendizaje por transferencia es que puede retener y reutilizar el conocimiento aprendido previamente. Sin embargo, el TL inductivo clásico consiste en una red neuronal simple pre-entrenada, donde el entrenamiento secuencial utiliza los pesos finales de la tarea auxiliar como condiciones iniciales de la tarea principal. Esto representa un método común para intentar resolver el problema antes mencionado para un modelo neuronal. En el contexto del pronóstico de series de tiempo, esta técnica es utilizada a través del uso de información previamente adquirida en el pasado, esto puede ofrecer buenos resultados cuando se cuentan con ciertas propiedades de estacionalidad o

de patrones repetitivos. Sin embargo, cuando la serie de tiempo es caótica las propiedades estacionarias se pierden, por lo tanto, la precisión del pronóstico es menor y el coeficiente de correlación es pobre. A continuación se propone una modificación al método TL inductivo basado en un modelo neuronal dependiente del uso del método ML.

Se supone que existe alguna relación explícita o implícita entre los dos dominios Λ_a y Λ_b, correspondientes a dos tareas τ_a y τ_b, se tiene que:

$$\begin{aligned} \Lambda_a = X_a(k) = [x_a(k-1), x_a(k-2), ..., x_a(k-n)]; \\ \Lambda_b = X_b(k) = [x_b(k-1), x_b(k-2), ..., x_b(k-n)], \end{aligned} \tag{3.21}$$

donde Λ_a y Λ_b están relacionadas, sin embargo $\tau_a \neq \tau_b$.

El clásico método de TL se define cuando al aplicarse un modelo neuronal, cómo lo puede ser (3.4) a la tarea τ_a, se utilizan los pesos W_{f,τ_a} de la $k-esima$ iteración de $k_e-esima$ época de entrenamiento, de modo que $W_{\tau_a} = W_{0,\tau_b}$, esto puede ser no efectivo para sistemas complejos o con falta de información. En comparación con el método de aprendizaje por transferencia modificado, en el que se usará un modelo neuronal con la misma estructura al de la tarea (τ_a) para la tarea τ_b, para aprovechar el conocimiento del domino Λ_b y compensar efectivamente la complejidad o falta de información en el dominio Λ_a.

El método de TL puede apoyarse en un conjunto Λ_σ de tareas auxiliares tal que $\Lambda_{a_i} \in \Lambda_\sigma$, donde cada una de las tareas auxiliares puede ser representada por el modelo (3.4-3.7). Se aplica la ley de aprendizaje (3.16) para cada modelo Λ_a para cada etapa de entrenamiento de modo que:

$$\begin{aligned} \hat{y}_{a1}(k) = NN[(\hat{y}_{a1}(k-1), \hat{y}_{a1}(k-2), ..., \hat{y}_{a1}(k-n)]; \\ \hat{y}_{a2}(k) = NN[(\hat{y}_{a2}(k-1), \hat{y}_{a2}(k-2), ..., \hat{y}_{a2}(k-n)]; \\ \vdots \\ \hat{y}_{ai}(k) = NN[(\hat{y}_{ai}(k-1), \hat{y}_{ai}(k-2), ..., \hat{y}_{ai}(k-n)]. \end{aligned} \tag{3.22}$$

Por cada tarea auxiliar se produce un W^*_{ai} que representa pesos de vectores sub-óptimos

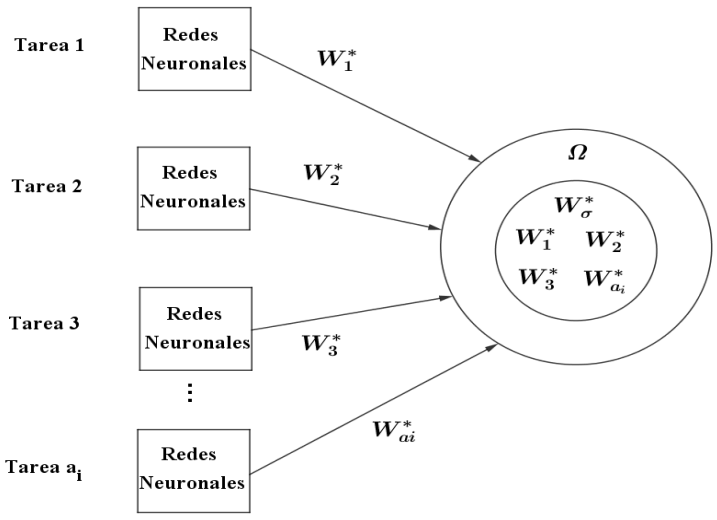


Figura 3.4: Transferencia-Aprendizaje.

del pronóstico de las diferentes series de tiempo o tareas τ_σ. Existe una región cerrada Ω tal que $W_\sigma^* \subset \Omega$, donde un W_σ^* es la matriz de ponderaciones sub-óptimas; entonces $W_{ai}^* \subset W_\sigma^*$.

La tarea principal Λ_a, puede tener varias tareas auxiliares contenidas en Λ_σ, sin embargo, el algoritmo (1) y el BP modificado (3.16) van a utilizar el conocimiento adquirido del TL, mediante la elección de un elemento de W_σ^*. De acuerdo con:

$$W^* = \arg\min_{W_\sigma^*} \left[\hat{y}_{\Lambda_\sigma}(k) - x_{\Lambda_b}(k)\right]^2, \tag{3.23}$$

donde el error se calcula en la etapa de prueba. La Figura (3.4) muestra cómo el aprendizaje de transferencia se usa para encontrar W^*.

En resumen, el ML clásico puede ayudar a evitar un mínimo local. Tiene rápida convergencia y produce un mejor rendimiento que el basado en el algoritmo del gradiente descendente. Cuando los datos de entrenamiento no son suficientes, el ML suele no trabajar con efectividad. Para resolver esto, es necesario el TL modificado para superar este problema.

La idea clave es que el método de ML utiliza los conocimientos adquiridos en el etapa de TL, ver Figura (3.5), siendo esto posible gracias a que las propiedades de convergencia

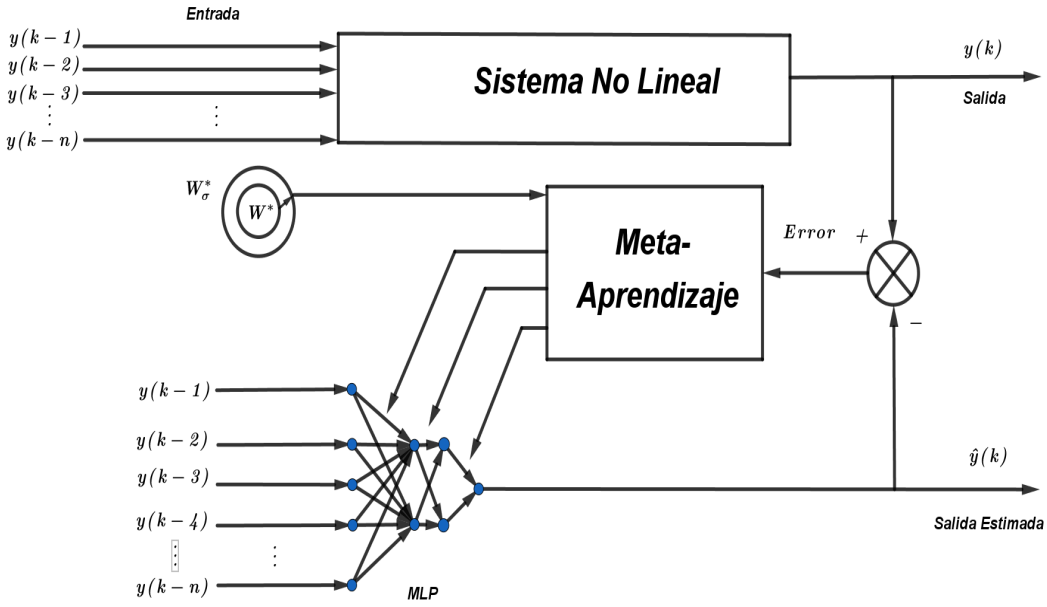


Figura 3.5: Transferencia de Aprendizaje para el pronóstico de series de tiempo

están garantizadas, las cuales serán analizadas más adelante.

Es posible aceptar que haya pérdida de información en los conjuntos de datos Λ_a y Λ_b, teniendo la condición que el porcentaje de información de pérdida en general sea $\Lambda_a >> \Lambda_b$. Se usará el conjunto de datos Λ_a para el pronóstico de series de tiempo. Pero primero se usaran los datos Λ_b para obtener los pesos W_b^* y V_b^*. Para aplicar el método de TL se propone el siguiente algoritmo (2) complementando con el algoritmo (1),

3.4. Algoritmo de Meta-Transferencia de Aprendizaje

Los métodos de ML y TL tienen condiciones necesarias que deben ser cumplidas para operar adecuadamente, sin embargo, al enfocarse en el TL se observa que en los métodos en tendencia han optado por usar técnicas basadas en la explotación de las bases de datos a través de la búsqueda masiva de patrones por parte de los ordenadores, en los problemas del

Algoritmo 2 Aprendizaje por Transferencia.

Usar el modelo (3.4),(3.6) o (3.7)
Aplicar el modelo (3.21) a la tarea principal Λ_p
if el índice J no puede ser minimizado **then**
    Proponer varias tareas auxiliares Λ_{ai} con un dominio relacionado a Λ_p
    Aplicar el modelo (3.22) para cada Λ_{ai}
    Tomar W^*_{ai} y formar el conjunto W^*_σ
    Usar el modelo (3.23) para seleccionar W^*
    Aplicar el método de ML de acuerdo con el algoritmo (1)
end if
Regresa la minimización de J

mundo real esto es muy complicado ya que puede ser no viable económicamente obtener esas cantidades de información, además de su complejo almacenamiento. Por lo que representa una oportunidad asegurar que el conocimiento de utilizado por el método TL modificado sea adecuado de acuerdo a las propiedades cualitativas de la serie temporal caótica.

De acuerdo con la propiedad fundamental de la transferencia de conocimiento, que establece que es posible utilizar los conocimientos adquiridos previamente en una tarea auxiliar Λ_a y así ayudar en el desempeño de la tarea principal Λ_p.

Definición 3.1 *Sean dos conjuntos tales que un dominio auxiliar Ψ_a y una tarea de aprendizaje auxiliar Λ_a, un dominio principal Ψ_p y una tarea de aprendizaje principal Λ_p. El aprendizaje de transferencia tiene como objetivo ayudar a mejorar la predicción de series de tiempo Λ_p por medio de la red neuronal (3.4) en Ψ_p utilizando el conocimiento de Ψ_a y Λ_a.*

Se denota a los datos perteneciente de un dominio auxiliar Ψ_a y Ψ_p:

$$\Psi_a = X(k+1+\xi_\sigma)_a = [y_a(k-1), y_a(k-2), ..., y_a(k-n)]; \tag{3.24}$$

$$\Psi_p = X(k+1+\xi_\sigma)_p = [y_p(k-1), y_p(k-2), ..., y_p(k-n)]. \tag{3.25}$$

Existe un problema fundamental dentro de la técnica de aprendizaje por transferencia: la selección de conocimientos previos adquiridos por una tarea auxiliar para la mejora en el desempeño de una tarea principal definida.

En este trabajo se propone el siguiente método para encontrar la información adecuada a través de la Transformada *Wavelet* (*TW*), esto permite encontrar la correlación entre los dos dominios Ψ_a y Ψ_p. En general, pueden existir n dominios Ψ_n entre los cuales se pueden hacer comparaciones para encontrar el mejor conjunto que garantice la mejora de los resultados obtenidos por la tarea principal.

La aplicación de métodos deterministas o estocásticos no es suficiente para predicción de horizontes múltiples. En el dominio del tiempo continuo o tiempo discreto, no es fácil observar características o propiedades de una serie de tiempo que permitan establecer una correlación entre dos bases de datos. Una opción es la Transformada de Fourier, sin embargo está depende un tiempo de muestreo definido que no todo fenómeno de la naturaleza tiene de manera estática, i.e., el pronóstico de la magnitud de terremotos no tiene un definido un tiempo de ocurrencia, por lo tanto significaría que el tiempo de muestreo es aleatorio. La Transformada *Wavelet* permite analizar señales sin la necesidad de un tiempo de muestreo, esto permite un análisis de series de tiempo con ocurrencia aleatoria. Una ventaja del análisis basado en *TW*: es la capacidad de realizar análisis espacio-temporales locales de series temporales. *TW* permite revelar aspectos de la señal que otras técnicas de análisis pasan por alto, como tendencias, puntos de corte, discontinuidades, etc. *TW* analiza series de tiempo de la punto de vista del espacio-tiempo y permite un análisis de múltiples sentidos [78].

Esta resolución múltiple también se puede obtener usando *TW*, llamada Transformada Discreta *Wavelet* (*DWT*) , utiliza bancos de filtros [79], mientras que *TW* utiliza versiones discretizadas de los ejes de escala y expansión. *DWT* es una transformación que descompone una señal dada en varios conjuntos. *DWT* se ha implementado con éxito en [80], [81].

Se supone lo siguiente: Sean los dominios Ψ_a y Ψ_p, estos guardan una correlación y esta puede ser determinada a través de la descomposición *Wavelet*.

La descomposición *Wavelet* es en realidad la aplicación de la *DWT*, pero para diferentes

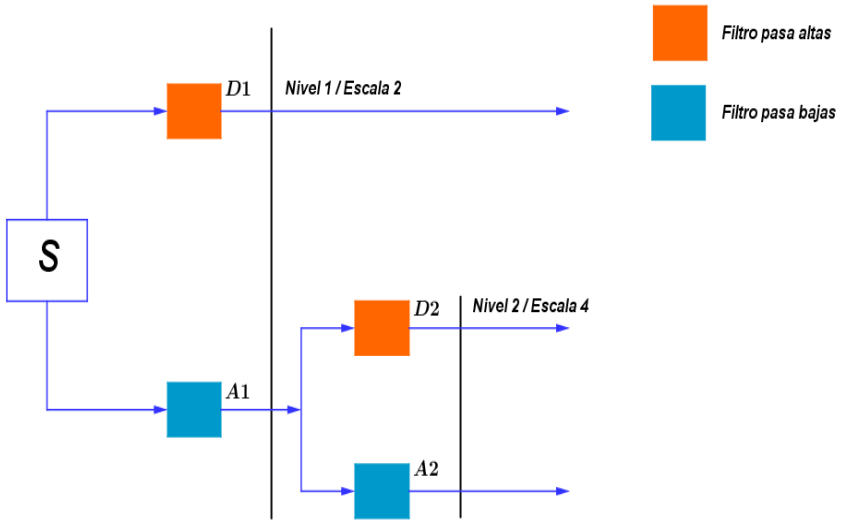


Figura 3.6: Descomposición Wavelet usando filtros pasa bajas y paso altas.

factores de escala [82]. La *DWT* se puede representar como en la ecuación (3.26):

$$W_{m.n}^{wav} = 2^{-\frac{m}{2}} \sum_{i=1}^{L} f_i \ \Psi[2^{-m}i - n], \tag{3.26}$$

donde m representa el índice de escala, n es la variable de traducción, Ψ es la *Wavelet* madre y L es la longitud de la serie o función $f(\cdot)$.

Aunque se aplica el concepto matemático de la Transformada *Wavelet*, computacionalmente consiste en un conjunto de filtros de paso altos y pasa bajas. Un ejemplo de esto se puede ver en la Figura (3.6) donde se muestra cómo se realiza una descomposición de escala 4 o nivel 2 para una señal dada. Como resultado, se obtienen coeficientes de aproximación (cA) y de detalle (cD). El primero representa la frecuencia más baja de la señal y el segundo la frecuencia más alta de la misma señal. La descomposición *Wavelet* se aplica a cada dominio Ψ_n. La Transformada *Wavelet* utiliza una amplia gama

de *Wavelet* de análisis de soporte ortonormales compactas. La ortogonalidad en *DWT* hace que la información deducida a una cierta escala m sea disjunta de la información a otras escalas:

$$\sigma_{wav}(m) = \left[\frac{1}{N-1} \sum_{n=1}^{N} (W_{m.n}^{wav} - \langle W_{m.n}^{wav} \rangle)^2 \right]^{\frac{1}{2}}, \tag{3.27}$$

donde N representa el número de coeficientes *Wavelet* en una escala dada m y $W_{m.n}$ es el promedio entre los coeficientes. A continuación se describe cómo calcular el estándar de desviación utilizando los coeficientes de ondícula para un par de dominios.:

$$\sigma_{cA} = \sqrt{\frac{1}{N-1} \sum_{i=0}^{N} (cA_i - W_{m_1})^2}; \tag{3.28}$$

$$\sigma_{cD} = \sqrt{\frac{1}{N-1} \sum_{i=0}^{N} (cD_i - W_{m_2})^2}, \tag{3.29}$$

donde $W_{m1} = mean(cA)$ and $W_{m2} = mean(cD)$.

Puede haber dos formas de interpretar la desviación estándar: como una correlación fuerte y como una correlación débil. Dependiendo de la naturaleza de la serie de tiempo caótica, se puede seleccionar uno u otro. Los dos coeficientes de desviación elegidos deben tener la desviación mayor, para que la información tomada sea completamente diferente y la red neuronal tenga un buen entrenamiento

Crear un dominio Ψ_{TF} permite generar una base de datos híbrida entre Ψ_a y Ψ_b, la cual será usada para encontrar los pesos óptimos W^*, estos serán usados por el Meta-Aprendizaje técnica y así, se evitarán los mínimos locales que podrían tener la tarea principal Λ_p. Para generar el conjunto asociado a Ψ_{TF}, se proponen los siguientes algoritmos, asumiendo que no toda serie de tiempo caótica tiene un tiempo definido Ts, es necesario considerar otro parámetro que nos permita entender la serie de tiempo en función del tiempo entre eventos, esto se denominará tiempo entre eventos. La siguiente función permite generar el dominio

Ψ_{TF}, en el cual tiene información de la serie temporal proveniente de las tareas Λ_p y Λ_a. La función que mezcla los datos de ambos conjuntos se describe mediante la siguiente ecuación,

$$f(\tau_{i_{\Lambda_p}}, \tau_{i_{\Lambda_a}}) = \begin{cases} \tau_{i_{\Lambda_p}} & if \geqslant \gamma; \\ \tau_{i_{\Lambda_p}} \oplus \tau_{i+\beta_{\Lambda_a}} \oplus \tau_{i+\beta+1_{\Lambda_p}} if < \gamma. \end{cases} \tag{3.30}$$

El tiempo entre eventos es necesario porque el algoritmo de búsqueda de características especiales de series de tiempo caóticas basado en el algoritmo de Transformación *Wavelet* de resolución múltiple busca la relación que existe entre las bases de datos de Λ_p y Λ_a considerando otra característica en lugar de la magnitud y así determinar el mejor conjunto de datos de Λ_a para usar el ML.

Cuando el tiempo entre eventos de Λ_p es menor que un tiempo entre eventos llamado γ, la información de datos de Λ_a se guarda en una matriz, que estará llena no sólo de los datos de tiempo series pero también ceros. Luego, se detecta la posición donde se guarda la información de los datos en el último arreglo para que este parámetro nos permita saber dónde agregar la información de la serie temporal que pertenece al dominio Λ_a en la información de Λ_p. Después de eso, un parámetro β indica la cantidad de datos de Λ_a agregados al conjunto de datos de Λ_p. Finalmente, la nueva señal para entrenar el Meta-Aprendizaje está lista.

La Figura (3.7) muestra cómo se usa el aprendizaje por transferencia para encontrar W^*, consulte el algoritmo (0). Además, se aplica el método ML para garantizar el uso de los conocimientos adquiridos en la etapa de TL. Este procedimiento tiene como objetivo resolver los problemas que se pueden presentar en la aplicación delBP.

Una vez aplicado el aprendizaje por transferencia modificado, esta información se utiliza para ejecutar el Meta-Aprendizaje, la combinación de ambas técnicas con el algoritmo de búsqueda se denominará Meta-Transferencia de Aprendizaje descrita en el esquema (3.7).

Para asegurar que la tarea principal Λ_p usa el conocimiento de pesos de W^*, el siguiente algoritmo modificado de retro-propagación puede minimizar el índice J en (2.6). Sin embargo,

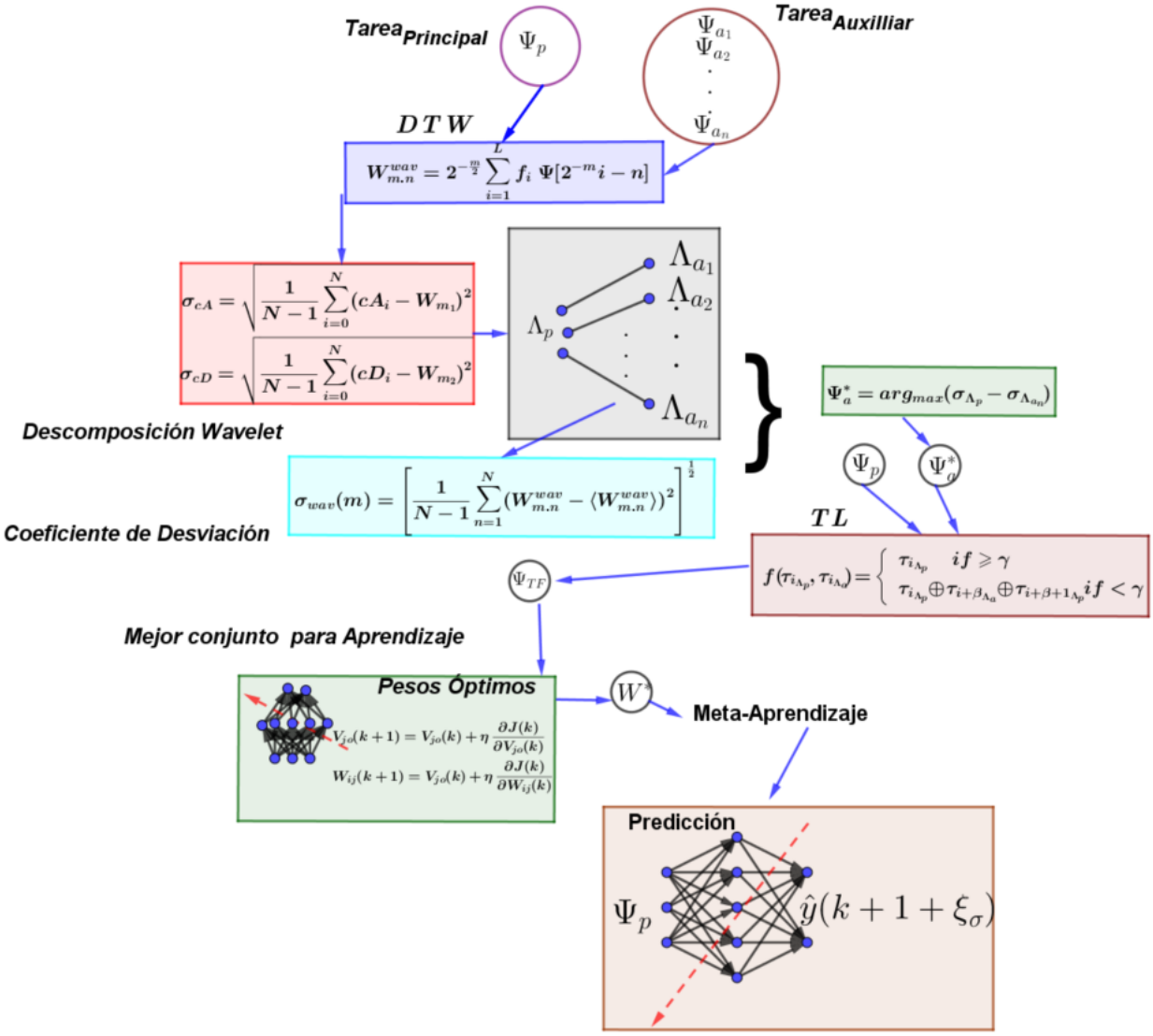


Figura 3.7: Metodología de Meta-Transferencia aprendizaje.

Algoritmo 3 Aprendizaje por Transferencia con método de búsqueda.

1: Elegir una tarea definida por Λ_p
2: Proponer varias tareas Λ_p. Este conjunto puede ser una correlación o no con Λ_p
3: Aplicar el modelo (3.26) para cada Λ_p y Λ_{ai}
4: Aplicar la ecuación (3.28) para obtener los coeficientes cA y cD
5: Compare la correlación de factores con el modelo (3.27)
6: Seleccionar un par Λ_p y Λ_p con más o menos correlación. Este criterio se elige mediante prueba.
7: Aplicar (mix) para combinar los conjuntos de datos de Λ_p y Λ_p
8: Usar el modelo (3.7)
9: Regresa los pesos sub-óptimos W^*

para la obtención de $\beta_{i,k}$ se puede presentar un problema ya que depende una condición angular Ac, al trabajar con un modelo neuronal profundo el tamaño de los vectores $\hat{X}_{i,k}$ aumentan en dimensión debido al número de pesos de la capa esto permite mejorar la precisión del pronóstico, esto propone condiciones necesarias para lograr que los pesos, i.e., W_k y V_k converjan a W^* y V^* respectivamente. Por lo tanto, es posible que se puede ciclar el algoritmo propuesto o provocar largos tiempo en la ejecución del método propuesto, es por ello que proponemos la siguiente modificación al ML para evitar el problema mencionado, se propone lo siguiente, ver algoritmo (4) para asegurar una solución por parte del método antes mencionado,

Como resultado del uso de esta nueva forma, el método de ML se modifica y es necesario formular el siguiente Teorema para este trabajo:

Teorema 3.1 *Sea el vector $\hat{X}_{V,k}$ que realiza la convergencia de la matriz de pesos V_k en la matriz de pesos óptimos V^*. Entonces el vector $\hat{X}^{s*}_{V,k}$ realiza la convergencia de la matriz de pesos V_k en la matriz de pesos sub-óptimos V^{s*}. La diferencia de las matrices de pesos $V^{s*} - V^* = 0$, si el número de iteraciones de acuerdo con el algoritmo (4 es suficiente, tal que $V^{s*} = V^*$.*

Es posible obtener el siguiente Lema a partir del Teorema (3.1).

Lemma 3.1 *Si $V^{s*} \neq V^*$, la matriz de pesos sub-óptimos $V^{s*} \in \Delta$, donde Δ es un compacto*

Algoritmo 4 Cálculo de $\hat{X}_{V,k}^{s*}$.

1: Seleccionar una condición inicial $\hat{X}_{V,0} = 0$
2: Seleccionar un coeficiente r (número de iteraciones)
3: **for** r veces **do**
4: Elegir un vector aleatorio $\hat{X}_{V,k}$ $0 \leq \hat{X}_{V,k}(i) \leq 1$
5: $i = 1, 2, ..., m$
6: Calcular $\left\| \hat{X}_{V,k}(k) \right\|$
7: Obtención de $cos\Theta_{V,i} = \frac{V^* - V_k}{\|V^* - V_k\|} \cdot \frac{\hat{X}_{V,k}(k)}{\left\|\hat{X}_{V,k}(k)\right\|}$
8: **end for**
9: Seleccionar el máx$(cos\Theta_{V,i})$
10: Regresa $\hat{X}_{V,k}^{s*}$

cerrado de radio δ *con centro en* V^{s*}*, tal que* $0 < \delta < \infty$*. Entonces,* $V^* = V^{s*} + \delta$*.*

De acuerdo con lo anterior, los modelos (3.16) y (3.19) se pueden reescribir como,

$$\beta_{W,k}^{s*} = (W^* - W_k) \frac{\hat{X}_{W,k}^{s*}}{\left\| \hat{X}_{W,k}^{s*} \right\|}, \tag{3.31}$$

y el ML modificado está dado por la siguiente ecuación:

$$\begin{aligned} W_{k+1} &= W_k - \eta \frac{\partial J}{\partial W} + \alpha \Delta W_k + \beta_{w,k}^{s*} \hat{X}_{W,k}^{s*}; \\ \eta_k &= \frac{\eta}{1 + \|\phi' X^T(k)\|^2}, 0 < \eta_k < \eta \leq 1, \end{aligned} \tag{3.32}$$

donde η es la tasa de aprendizaje positiva $0 < \eta < 1$ y $\frac{\partial J}{\partial W} = \phi' \left[W_k \hat{X}(k) \right] E(k)$. Además, $\Delta W_k = W_k - W_{k-1}$, α es una constante $0 < \alpha < 1$ y $\beta_{w,k}^{s*}$, $\hat{X}_{W,k}^{s*}$ son los nuevos parámetros del Meta-Aprendizaje modificado.

3.5. Propiedades importantes del Meta-Transferencia de Aprendizaje

En esta sección, se discuten algunas de las propiedades matemáticas, tales como la estabilidad de ley de aprendizaje basada en el Meta-Aprendizaje y la convergencia hacia pesos óptimos determinados por la Transferencia-Aprendizaje. Las cuales son necesarias para un entendimiento profundo del método propuesto de Meta-Transferencia de Aprendizaje descrito en la secciones anteriores así como la demostraciones de las aseveraciones antes mencionadas.

3.5.1. Estabilidad

A continuación, se discuten dos casos generales: una sola capa en la red neuronal, ver modelo (3.4), y una red neuronal multicapa, ver modelo (3.6). Si la red neuronal por construcción estuviera formada por más de dos capas, ver modelo (3.7). Considérense las siguientes definiciones, [83]:

Definición 3.2 *Una función continua $h(\cdot)$ es clase $\mathcal{K}$, si $h|\ [0,a) \to [0,\infty)$. Esto es:*

- *Que sea estrictamente creciente.*
- $h(0) = 0$.

Definición 3.3 *Una función continua $p(\cdot)$ es clase $\mathcal{KL}$, si $p|\ [0,a) \times [0,\infty) \to [0,\infty)$. Esto es:*

- *Para cada punto fijo s se tiene que la función $p(r,s)$, pertenece a la clase $\mathcal{K}$.*
- *Para cada punto fijo r se tiene que la función $p(r,s)$ es decreciente con respecto s y por lo tanto $lim_{s_k \to \infty} p\,(s_k) = 0$.*

Definición 3.4 *Una función continua $g(\cdot)$ es clase $\mathcal{K}_\infty$, si $g|\ [0,a) \to [0,\infty)$. Esto es:*

- *la función $g(\cdot)$ es clase $\mathcal{K}$.*

- $a = \infty$.
- $\lim_{r\to\infty} g(r) = \infty$.

Definición 3.5 *Un sistema (3.3) es globalmente Entrada-Estado Estable (ISS), si existe una función $\gamma(\cdot)$ de clase $\mathcal{K}$ y existe una función $\beta(\cdot)$ de clase $\mathcal{KL}$; donde para cada $u \in L_\infty$ $(\sup\{\|u(k)\|\} < \infty)$ y para cada condición inicial de los estados, $x^0 \in R^n$, se mantiene que:*

$$\left\|x\left(k, x^0, u\left(k\right)\right)\right\| \leq \beta\left(\left\|x^0\right\|, k\right) + \gamma\left(\left\|u\left(k\right)\right\|\right).$$

Definición 3.6 *El sistema descrito en (3.3) tiene una función suave ISS $V : \Re^n \to \Re^+$, si las siguientes condiciones se cumplen:*

(a) Existen dos funciones clase $\mathcal{K}_\infty$, $\alpha_1(\cdot)$ y $\alpha_2(\cdot)$ de tal manera que:

$$\alpha_1(s) \leq V\left(s\right) \leq \alpha_2(s), \quad \forall s \in \Re^n.$$

(b) Si existe una función $\alpha_3(\cdot)$ clase $\mathcal{K}_\infty$ y una función $\alpha_4(\cdot)$ clase $\mathcal{K}$ tal que:

$$V_{k+1} - V_k \leq -\alpha_3(\|x\left(k\right)\|) + \alpha_4(\|u\left(k\right)\|),$$

para todo $x\left(k\right) \in \Re^n, u\left(k\right) \in \Re^m$.

Lemma 3.2 *[84], para un sistema dinámico no lineal descrito en tiempo-discreto, las siguientes sentencias son equivalentes:*

- *es Entrada-Estado Estable;*
- *es robustamente estable;*
- *admite una función suave ISS-Lyapunov.*

Propiedad 1. Un sistema dinámico no lineal es *ISS*, si el comportamiento del sistema permanece acotado, cuando su entrada esta acotada.

De acuerdo con el teorema de Stone-Weierstrass, [85]. En general, un sistema dinámico no lineal (3.3) se puede describir a través de un modelo neuronal (3.4), con la adición de un error de modelado:

$$y\left(k\right)=\phi\left[W^{*}x\left(k\right)\right]+\mu\left(k\right), \tag{3.33}$$

donde W^{*} es la matriz de pesos sinápticos óptimos, $\mu\left(k\right)$ es el error de modelado.

Debido a que la función $\phi(\cdot)$ está acotada y la salida del sistema $y\left(k\right)$, se asume que el error de modelado $\mu\left(k\right)$ está acotado:

$$\left\|\mu\left(k\right)\right\|^{2}<\bar{\mu},$$

donde $\overline{\mu}$ es una constante positiva. El error de identificación se encuentra definido por:

$$e(k)=\hat{y}(k)-y(k). \tag{3.34}$$

Expandiendo en Series de Taylor alrededor del punto $[W_{k}X\left(k\right)]$, se obtiene que el error de identificación puede representarse como:

$$\begin{aligned} e\left(k\right) &= \phi\left[W_{k}X\left(k\right)\right]-\phi\left[W^{*}X\left(k\right)\right]+\mu\left(k\right) \\ &= \phi^{'}\left[W_{k}X\left(k\right)\right]\widetilde{W}_{k}X\left(k\right)+\zeta\left(k\right), \end{aligned} \tag{3.35}$$

donde $\widetilde{W_{k}}=W_{k}-W^{*}$, el termino de segundo orden de la aproximación del error esta definido por $\varepsilon\left(k\right)$ de tal modo se establece que el error de modelado es: $\zeta\left(k\right)=\varepsilon\left(k\right)+\mu\left(k\right)$. Además, $\phi^{'}$ es la derivada de la función de activación no lineal $\phi\left(\cdot\right)$ en el punto $[W_{k}X\left(k\right)]$ respecto de su argumento debido al gradiente descendente.

Ya que $\mu\left(k\right)$ está acotada, $\varepsilon\left(k\right)$ se encuentra acotado de la siguiente forma:

$$\left\|\varepsilon\left(k\right)\right\|^{2}\leq\overline{\varepsilon},$$

donde $\overline{\varepsilon}$ es una constante positiva.

La estabilidad del algoritmo de Meta-Aprendizaje para una red neuronal con una sola capa oculta se enuncia en el Teorema (3.2).

Teorema 3.2 *Considérese un sistema no lineal discreto (3.3), que se identifica a través de una red neuronal con una sola capa oculta (3.4). El error de identificación $e(k)$ permanece acotado en el sentido L_∞, si la ley de Meta-Aprendizaje (3.36) es Entrada-Estado Estable.*

$$W_{k+1} = W_k - \eta_k e\left(k\right)\varphi' X^T\left(k\right) + \alpha\Delta W_k + \beta_{w,k}\hat{X}_{W,k}, \tag{3.36}$$

donde $\eta_k = \frac{\eta}{1+\|\varphi' X^T(k)\|^2}$, $0 < \eta_k < \eta \leq 1$, $0 < \alpha \leq 1$.

Demostración. De las ecuaciones (3.17) y (3.19), el termino de Meta-Aprendizaje se puede reescribir cómo:

$$\beta_{w,k}\hat{X}_{W,k} = -\tilde{W}_k \frac{X_p^T X_p}{\|X_p\|} = -\gamma\tilde{W}_k. \tag{3.37}$$

Se propone la siguiente función candidata de Lyapunov:

$$L_k = 2\left\|\tilde{W}_k\right\|^2 + 3\left\|\alpha\tilde{W}_{k-1}\right\|^2 + 7\left\|\alpha\tilde{W}_k\right\|^2 + \left\|\gamma\alpha\tilde{W}_{k-1}\right\|^2 + \left\|\gamma\tilde{W}_k\right\|^2 + \left\|\tau\tilde{W}_k\right\|^2, \tag{3.38}$$

donde L_k es una función cuadrática, por definición es creciente, es $L_k(0) = 0$ e incluye al error de identificación, por lo tanto es una función candidata de Lyapunov. De la ley de aprendizaje (3.36), se tiene que:

$$\tilde{W}_{k+1} = \tilde{W}_k - \eta_k e\left(k\right)\varphi' X^T\left(k\right) - \alpha\Delta\tilde{W}_k - \gamma\tilde{W}_k.$$

El incremento $\Delta L_k = L_{k+1} - L_k$, se define de la siguiente forma:

$$\begin{aligned}
\Delta L_k &\leq \left(1+\alpha\right)\left\|\eta_k e\left(k\right)\varphi' X\left(k\right)\right\|^2 - \eta_k\left(1+\alpha-\gamma\right)e^2\left(k\right) + \eta_k\left(1+\alpha-\gamma\right)\zeta^2\left(k\right) \\
&= -\eta_k\left[\left(1+\alpha\right) - \left(1+\alpha-\gamma\right)\eta\frac{\left\|\varphi' X^T(k)\right\|^2}{1+\left\|\varphi' X^T(k)\right\|^2}\right]e^2\left(k\right) + \eta_k\left(1+\alpha-\gamma\right)\zeta^2\left(k\right) \\
&\leq -\pi e^2\left(k\right) + \eta_k\left(1+\alpha-\gamma\right)\zeta^2\left(k\right),
\end{aligned} \tag{3.39}$$

donde:

$$\pi = (1+\alpha) - (1+\alpha-\gamma)\,\eta\frac{K}{1+K} > 0; \tag{3.40}$$

$$K = \sup_{k} \left\|\varphi' X^T(k)\right\|^2. \tag{3.41}$$

Con base en el error de los pesos denotado por $\widetilde{w}_i = w_i - w^*$, se proponen las siguientes cotas para función L_k, tal que:

$$n\,min\left(\tilde{w}_i^2\right) \leq L_k \leq n\,max\left(\tilde{w}_i^2\right),$$

donde $n\,min\left(\tilde{w}_i^2\right)$ y $n\,max\left(\tilde{w}_i^2\right)$ son funciones clase $\mathcal{K}_\infty$, mientras que $\pi e^2(k)$ y $\eta_k \zeta^2(k)$ son funciones clase $\mathcal{K}$. De la ecuación (3.35), se sabe que L_k es función de $e(k)$ y $\zeta(k)$, entonces L_k es una función suave ISS-Lyapunov de acuerdo con la Definición (3.6). A partir del Teorema (3.2) mostrado en [84], la dinámica del error de identificación es Entrada-Estado Estable, ya que la "entrada"$\zeta(k)$ está acotada y la dinámica del "estado"(error de identificación) es ISS, por lo que $e(k)$ está acotado de acuerdo con la Definición (3.5). ■

La ecuación (3.6), puede ser simplificada de la siguiente manera:

$$\begin{aligned} W_{k+1} = W_k &- \eta_k e(k)\,\varphi' X^T(k) \\ &- \alpha\Delta W_k - \gamma W_k. \end{aligned} \tag{3.42}$$

Para el caso MLP, el modelo (3.6) puede escribirse cómo:

$$y(k) = V^*\phi\left[W^* X(k)\right] + \mu(k),$$

donde W^* y V^* son matrices de pesos desconocidas, que pueden minimizar el error de modelo $\mu(k)$.

El sistema dinámico no lineal (3.3) puede ser expresado de la siguiente forma:

$$y(k) = V^0\phi\left[W^* X(k)\right] + \delta(k), \tag{3.43}$$

donde V^0 es una matriz de pesos desconocida.

Generalmente, $\|\delta(k)\| \geq \|\mu(k)\|$. Utilizando las Series de Taylor alrededor del punto $W_k X(k)$, el error de identificación puede ser reescrito cómo:

$$\begin{aligned} e(k) &= V_k\phi[W_k X(k)] - V^0\phi[W^* X(k)] + \delta(k) \\ &= V_k\phi[W_k X(k)] - V^0\phi[W_k X(k)] + V^0\phi[W_k X(k)] - V^0\phi[W^* X(k)] + \delta(k) \\ &= \widetilde{V}_k\phi[W_k X(k)] + V^0\phi'\widetilde{W}_k X(k) + \zeta(k), \end{aligned} \tag{3.44}$$

donde ϕ' denota la derivada de la función de activación $\phi(\cdot)$ alrededor del punto $W_k X(k)$, $\widetilde{W}_k = W_k - W^*$, $\widetilde{V}_k = V_k - V^0$, $\zeta(k) = V^0\varepsilon(k) + \delta(k)$.

$\varepsilon(k)$ representa la aproximación de segundo orden del error, mediante la expansión de la Series de Taylor. La identificación se lleva acabó en lazo abierto, se asume que (3.3) es entrada-acotada y salida-acotada, *i.e.*, la salida $y(k)$ y entrada $u(k)$ del sistema dinámico no lineal (3.3) es acotada. La función de activación ϕ es acotada, por lo que se asume que $\delta(k)$ en (3.43), $\varepsilon(k)$, y $\zeta(k)$ en (3.44) son acotadas.

En el siguiente teorema se enuncia el análisis de estabilidad para el Meta-Aprendizaje en un *MLP* .

Teorema 3.3 *Considérese un sistema no lineal discreto (3.3), que se identifica a través de una red neuronal MLP (3.6) ó (3.7). El error de identificación $e(k)$ permanece acotado en el sentido L_∞, si la ley de Meta-Aprendizaje (3.45) es Entrada-Estado Estable.*

$$\begin{aligned} W_{k+1} &= W_k - \eta_k e(k)\,\varphi' V^{0T} X^T(k) - \alpha\Delta W_k - \gamma W_k; \\ V_{k+1} &= V_k - \eta_k e(k)\,\varphi' X^T(k) - \alpha\Delta V_k - \gamma V_k, \end{aligned} \tag{3.45}$$

donde $\eta_k = \frac{\eta}{1+\|\varphi' P^{0T} X^T(k)\|^2 + \|\varphi\|^2}$, $0 < \eta \leq 1$.

Demostración.

Se propone la siguiente función candidata de Lyapunov L_{M_k}, la cuál es un función cuadrática. Por definición es creciente y $L_k(0) = 0$:

$$L_{M_k} = \|L_{W_k}\|^2 + \|L_{V_k}\|^2 . \tag{3.46}$$

Es posible escribir L_{W_k} y L_{V_k}, con base en la ecuación (3.38), ya que la ley de aprendizaje propaga el error a través de cada capa oculta al estar basada en el gradiente descendente, de modo que:

$$\begin{aligned}\tilde{W}_{k+1} &= \tilde{W}_k - \eta_k e\left(k\right)\varphi' V^{0T} X^T(k) - \alpha\Delta W_k - \gamma W_k;\\ \tilde{V}_{k+1} &= \tilde{V}_k - \eta_k e\left(k\right)\varphi' X^T\left(k\right) - \alpha\Delta V_k - \gamma V_k.\end{aligned}$$

De acuerdo con Teorema (3.2), el incremento de L_{V_k} se establece de la misma manera la ecuación (3.39), mientras que el incremento ΔL_{W_k} es:

$$\begin{aligned}\Delta L_{W_k} &\leq \eta_k \left[(1+\alpha) - (1+\alpha-\gamma)\eta \frac{\left\|\varphi' V^{0T} X^T(k)\right\|^2}{1+\left\|\varphi' V^{0T} X^T(k)\right\|^2}\right] e^2(k)\\ &+\eta_k(1+\alpha-\gamma)\zeta^2(k) \leq -\pi e^2(k) + \eta_k\left(1+\alpha-\gamma\right)\zeta^2(k)\end{aligned}.$$

Siguiendo el desarrollo similar al Teorema (3.2), las cotas para L_{M_k} se establecen cómo:

$$n[min(\tilde{w_i}^2) + min(\tilde{v_i}^2)] \leq L_{M_k} \leq n[max(\tilde{w_i}^2) + max(\tilde{v_i}^2)],$$

donde $n[min(\tilde{w_i}^2) + min(\tilde{v_i}^2)]$ y $n\left[\text{máx}\left(\tilde{w_i}^2\right) + \text{máx}\left(\tilde{v_i}^2\right)\right]$ son funciones clase $\mathcal{K}_\infty$, mientras que $\pi e^2(k)$ es una función clase $\mathcal{K}_\infty$, y el termino $\eta_k\left(1+\alpha-\gamma\right)\zeta^2\left(k\right)$ es una función clase $\mathcal{K}$.

A partir de la ecuación (3.44) y la Definición (3.6), L_{M_k} esta en función $e(k)$ y $\zeta\left(k\right)$ admite un función suave ISS-Lyapunov que satisface la Definición (3.6).A partir del Teorema (3.3), si la "entrada"$\zeta\left(k\right)$ es acotada, la dinámica del "estado" $e(k)$ es acotado, [84]. ■

Para la estabilidad de Meta-Aprendizaje, es valido mencionar que se encuentra comprendido en las demostraciones anteriores, sin embargo, deben hacerse las pruebas de convergencia

débil y fuerte para demostrar el Teorema (3.1).

3.5.2. Convergencia

En esta sección se presenta el análisis de la convergencia. Para mostrar la efectividad del método propuesto *MTA* en este trabajo, se demuestra la propiedad de convergencia fuerte, la cual garantiza que los pesos sinápticos W_k convergen a los pesos sub-óptimos W^{s*}. Para determinar la convergencia fuerte es fácil saber que primero se necesita probar la convergencia débil, es decir, que el incremento de $\Delta W_k \to 0$. Para ayudar a probar estas propiedades, se utiliza el siguiente teorema,

Teorema 3.4 *El proceso de entrenamiento de una red neuronal basado en el método de Meta-Transferencia Aprendizaje esta acotado en el sentido L_∞, si el error de identificación permanece acotado:*

$$|e(k)| < \infty. \tag{3.47}$$

Demostración. Del Teorema (3.4)] El análisis de estabilidad *ISS* de la sección anterior, concluye que el error de identificación $e(k)$ está acotado en el sentido L_∞. Sin embargo, el método de *MTA* utiliza los pesos sub-óptimos W^{s*}, para completar la prueba que establezca (3.47), es necesario hacer la demostración del Teorema (3.1), por lo que:
Demostración.[Del Teorema (3.1)] Para una sola capa oculta (3.4), el Meta-Aprendizaje de acuerdo con el Teorema (3.1):

$$\beta_{w,k}^{s*}\hat{X}_{W,k}^{s*} = -\tilde{W}_k^s \frac{X_p^T X_p}{\|X_p\|} = -\gamma \tilde{W}_k^s, \tag{3.48}$$

donde $\widetilde{W}_k^s = W_k - W^{s*}$. Siguiendo el desarrollo del Teorema (3.2) se tiene que:

$$\begin{aligned} L_k^s = 2\left\|\tilde{W}_k^s\right\|^2 + 3\left\|\alpha\tilde{W}_{k-1}^s\right\|^2 + 7\left\|\alpha\tilde{W}_k^s\right\|^2 \\ + \left\|\gamma\alpha\tilde{W}_{k-1}^s\right\|^2 + \left\|\gamma\tilde{W}_k^s\right\|^2 + \left\|\tau\tilde{W}_k^s\right\|^2. \end{aligned}$$

De la ley de aprendizaje (3.32):

$$\tilde{W}_{k+1}^{s} = \tilde{W}_{k}^{s} - \eta_k e\left(k\right)\varphi' X^{T}\left(k\right) - \alpha\Delta\tilde{W}_{k}^{s} - \gamma\tilde{W}_{k}^{s}.$$

El incremento $\Delta L_k^s = L_{k+1}^s - L_k^s$ es:

$$\begin{aligned}
&\Delta L_k^s \leq \left(1+\alpha\right)\left\|\eta_k e\left(k\right)\varphi' X\left(k\right)\right\|^2 \\
&-\eta_k\left(1+\alpha-\gamma\right)e^2\left(k\right) + \eta_k\left(1+\alpha-\gamma\right)\zeta^2\left(k\right) \\
&= -\eta_k\left[\left(1+\alpha\right) - \left(1+\alpha-\gamma\right)\eta\frac{\left\|\varphi' X^T(k)\right\|^2}{1+\left\|\varphi' X^T(k)\right\|^2}\right]e^2\left(k\right) \\
&+\eta_k\left(1+\alpha-\gamma\right)\zeta^2\left(k\right) \\
&\leq -\pi e^2\left(k\right) + \eta_k\left(1+\alpha-\gamma\right)\zeta^2\left(k\right)
\end{aligned}$$

, donde:

$$\pi = \left(1+\alpha\right) - \left(1+\alpha-\gamma\right)\eta\frac{K}{1+K} > 0;$$
$$K = \sup_{K}\left\|\varphi' X^T(k)\right\|^2.$$

Por lo que se tiene que:

$$n\,min\left(\tilde{w}_i^2\right) \leq L_k^s \leq n\,max\left(\tilde{w}_i^2\right),$$

donde $n\,min\left(\tilde{w}_i^2\right)$, mientras que $n\,max\left(\tilde{w}_i^2\right)$ son funciones clase $\mathcal{K}_\infty$, y $\pi e^2\left(k\right)$ y $\eta_k\zeta^2\left(k\right)$ son funciones clase $\mathcal{K}$. De la ecuación (3.35), se sabe que L_k^s es función de $e\left(k\right)$ y $\zeta\left(k\right)$, entonces L_k es una función suave ISS-Lyapunov de acuerdo con la Definición (3.6). A partir del Teorema (3.2) mostrado en [84], la dinámica del error de identificación es entrada-estado estable, ya que la "entrada" $\zeta\left(k\right)$ está acotada y la dinámica del "estado" (error de identificación) es ISS, por lo que $e(k)$ está acotado de acuerdo con la Definición (3.5).

Para redes neuronales MLP, se utiliza la siguiente función definida positiva $L_{M_k}^s$:

$$L_{M_k}^s = \left\|\tilde{W}_k^s\right\|^2 + \left\|\tilde{V}_k^s\right\|^2, \tag{3.49}$$

de ley de aprendizaje modificada (3.32):

$$\tilde{W}_{k+1} = \tilde{W}_k - \eta_k e(k)\,\varphi' V^{0T} X^T(k) - \alpha \Delta W_k - \gamma W_k;$$
$$\tilde{V}_{k+1} = V_k - \eta_k e(k)\,\varphi' X^T(k) - \alpha \Delta V_k - \gamma V_k.$$

Desarrollo similar con el Teorema (3.3), se tiene que:

$$\begin{aligned}\Delta L^s_{M_k} &\leq \eta_k \left[(1+\alpha) - (1+\alpha-\gamma)\,\eta \frac{\left\|\varphi' V^{0T} X^T(k)\right\|^2}{1+\left\|\varphi' V^{0T} X^T(k)\right\|^2}\right] e^2(k) \\ &+\eta_k (1+\alpha-\gamma)\,\zeta^2(k) \\ &\leq -\pi e^2(k) + \eta_k (1+\alpha-\gamma)\,\zeta^2(k).\end{aligned}$$

Además:

$$n\left[\text{mín}\left(\tilde{w}_i^2\right) + \text{mín}\left(\tilde{v}_i^2\right)\right] \leq L^s_{M_k} \leq n\left[\text{máx}\left(\tilde{w}_i^2\right) + \text{máx}\left(\tilde{v}_i^2\right)\right],$$

donde $n\left[\text{mín}\left(\tilde{w}_i^2\right) + \text{mín}\left(\tilde{v}_i^2\right)\right]$ y $n\left[\text{máx}\left(\tilde{w}_i^2\right) + \text{máx}\left(\tilde{v}_i^2\right)\right]$ son funciones clase $\mathcal{K}_\infty$, $\pi e^2(k)$ es una función clase $\mathcal{K}_\infty$, $\eta_k(1+\alpha-\gamma)\,\zeta^2(k)$ es una función clase $\mathcal{K}$. A partir de la Definición (3.5), $L^s_{M_k}$ admite una función suave ISS-Lyapunov [86], y esta descrita en función $e(k)$ y $\zeta(k)$. Del Teorema (3.2), sí la "entrada" $\zeta(k)$ es acotada, entonces la dinámica del "estado" $e(k)$ es acotada. ■

Debido a lo anterior el error de identificación esta acotado en el sentido L_∞, por lo que la ecuación (3.47) se establece. ■

La convergencia débil para la Meta-Transferencia de Aprendizaje se enuncia por el siguiente teorema:

Teorema 3.5 *El método de Meta-Transferencia de Aprendizaje (3.32)-(3.31) tiene convergencia débil si, i.e.:*

$$lim_{k\to\infty}\Delta W_k^2 = 0, \quad lim_{k\to\infty}\Delta V_k^2 = 0 \tag{3.50}$$

, donde el incremento esta definido cómo:

$$\Delta W = W_{k+1} - W_k, \quad \Delta V = V_{k+1} - V_k.$$

Demostración. Dado que el error de identificación $e(k)$ está acotado, ver Teoremas (3.1), (3.2) y (3.3), esto significa que el término de gradiente asociado con cada una de las capas debido a la ley de aprendizaje de MTA es aproximadamente cero, cuando el tiempo tiende a infinito. Por tanto, el incremento definido por ΔW y ΔV tiende a cero y se establece (3.50). ■

El siguiente teorema prueba la convergencia fuerte de la propuesta de Meta-Transferencia de Aprendizaje.

Teorema 3.6 *Existe un conjunto $W_{ai}^* \subset \Omega$ tal que $W^{s*} \subset W_\sigma$. El método de Meta-Transferencia de Aprendizaje (3.32) conduce a una convergencia fuerte si:*

$$\lim_{k\to\infty} W_k = W^{s*}. \tag{3.51}$$

Demostración. Se define W^{s*} cómo una matriz sub-óptima de W_k en el instante k. El ángulo θ de la proyección de W_k sobre W^{s*} es:

$$cos\theta = \frac{W_k \cdot W^{s*}}{\parallel W_k \parallel \parallel W^{s*} \parallel}. \tag{3.52}$$

Se define l, ver Figura (3.8):

$$l = \parallel W^{s*} \parallel cos\theta = l_x + l_x = \frac{W^{s*} + W_k}{\parallel W_k \parallel} + \frac{W^{s*} + W_k}{\parallel W_k \parallel} = \frac{W_k \cdot W^{s*}}{\parallel W_k \parallel}.$$

Usando la desigualdad triangular:

$$\frac{W^{s*}W_k}{\|W^{s*}\|\|W_k\|} \leq \frac{W^{s*}}{\|W^{s*}\|\|W_k\|} - \frac{W_k}{\|W^{s*}\|\|W_k\|}. \tag{3.53}$$

Usando la ley de aprendizaje (3.32), el incremento es:

$$\Delta W = \frac{W^{s*} \cdot W_{k+1}}{\parallel W^{s*} \parallel \parallel W_{k+1} \parallel} - \frac{W^{s*} \cdot W_k}{\parallel W^{s*} \parallel \parallel W_k \parallel} \geq 0,$$

donde $k = 1, 2, ...$ y usando la desigualdad Cauchy-Bunyakovsky-Schwarz, el incremento ΔW junto con el teorema 3.4, el incremento a lo largo de la secuencia es:

$$0 \leq \left\| \frac{W^{s*} W_k}{\|W^{s*}\| \|W_k\|} \right\|^2 \leq \sum_{i=k+1}^{\infty} (\Delta W_i^{s*})^2 \leq \sum_{i=k}^{k+1} \Delta W_i^2 \leq \sum_{i=0}^{k} \Delta W_i^2.$$

Debido a:

$$0 \leq \|W^{s*} - W_{k+1}\| \leq \|W_{k+1} - W_k\| \leq \|W_k - W_0\| \leq \|W_0\|, \tag{3.54}$$

y que el error de modelado esta acotado cómo (3.47). Al aplicarle la función límite a (3.54) entonces es posible establecer (3.51), dado que el método MTA reduce en cada iteración la norma $\|W^{s*} - W_k\|$ a través del termino $\beta_{w,k}^{s*} \hat{X}_{W,k}^{s*}$. ■

Teorema 3.7 *Existe a $W_{ai}^* \subset \Omega$ tal que $W^{s*} \subset W_\sigma$. El método de Meta-Transferencia de Aprendizaje (3.32) conduce a una convergencia fuerte tal que:*

$$\lim_{k \to \infty} W_k^{s*} = W^*. \tag{3.55}$$

Demostración. Al cumplirse la condición de convergencia fuerte del Teorema (3.6) :

$$lim_{k \to \infty} W_k = W^{s*}.$$

Entonces, es posible determinar que a través de un camino similar se obtiene que:

$$0 \leq \|W^* - W_{k+1}^{s*}\| \leq \|W_k^{s*} - W_{k+1}\| \leq \|W_{k+1} - W_k\|, \tag{3.56}$$

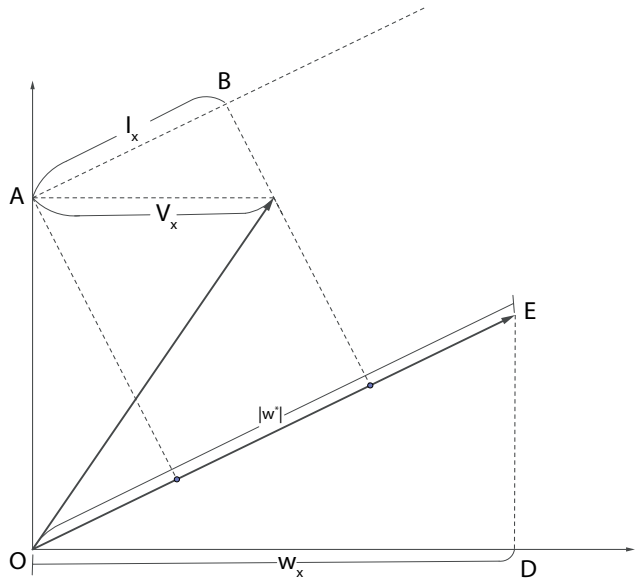


Figura 3.8: Ángulo entre dos vectores

Por lo tanto se puede establecer que:

$$lim_{k\to\infty} W_k^{s*} = W^*$$

Por lo anterior queda demostrado que la convergencia fuerte se sigue cumpliendo siempre que exista un número de iteraciones suficientes del algoritmo (1) para proyectar los pesos sinápticos tal que $W^{s*} = W^*$. ■

Con el resultado anterior es posible enunciar la prueba del Lema (1), de modo que: *Demostración.* La desigualdad triangular para el ángulo entre dos vectores:

$$\frac{W^* W_k^{s*}}{\|W^*\|\|W_k^{s*}\|} \leq \frac{W^*}{\|W^*\|\|W_k^{s*}\|} - \frac{W_k^{s*}}{\|W^*\|\|W_k^{s*}\|}. \tag{3.57}$$

Es posible definir la diferencia del lado derecho de la desigualdad anterior cómo:

$$\delta_\sigma(k) = \frac{W^*}{\|W^*\|\|W_k^{s*}\|} - \frac{W_k^{s*}}{\|W^*\|\|W_k^{s*}\|}. \tag{3.58}$$

Aplicando el límite al modelo anterior se tiene que:

$$lim_{k\to\infty}\delta_\sigma(k) = \delta,$$

entonces δ es un conjunto finito, siendo este un espacio compacto cerrado donde las sub-sucesiones contenidas son convergentes a W^*. ■

3.6. Meta-Aprendizaje para el controlador PID

De acuerdo con [87], un sistema de control se puede expresar en términos de sus señales de entrada y salida. En el caso de controlador PID es posible estimar las ganancias K_p, K_i y K_d a través :

$$[K_p, K_i, K_d] = NN[y(k-1), \ldots, y(k-n), u(k-1), \ldots, u(k-n), \\ r(k-1), \ldots, r(k-n), e(k), \ldots, e(k-n)], \tag{3.59}$$

donde y es la salida del sistema de control, u es la señal de control, r es el punto de referencia, e es la señal de error y $n = 1, 2, \ldots$ es el retardo en el tiempo. El mapeo de entrada-salida de un MLP de una sola capa oculta se parametriza cómo:

$$[K_p, K_i, K_d] = \Gamma\left[V_{jo} \cdot \Phi\left[W_{ij} \cdot X(k)\right]\right]. \tag{3.60}$$

Para un MLP, Γ y Φ son funciones de activación no lineales, W y V son las matrices de pesos sinápticos y $X(k)$ es el vector de entradas, para $i = 1, \ldots, n_i$, $j = 1, \ldots, n_h$ y

$o = 1, \ldots, n_o$, y el vector de entrada se define cómo:

$$X(k) = [y(k), y(k-1), u(k), u(k-1), \\ r(k), r(k-1), e(k), e(k-1), e(k-2)], \quad (3.61)$$

donde y representa el comportamiento de la variable de proceso en el tiempo y r es la referencia para la variable de proceso, en la siguiente figura se muestra un esquema de lo descrito anteriormente.

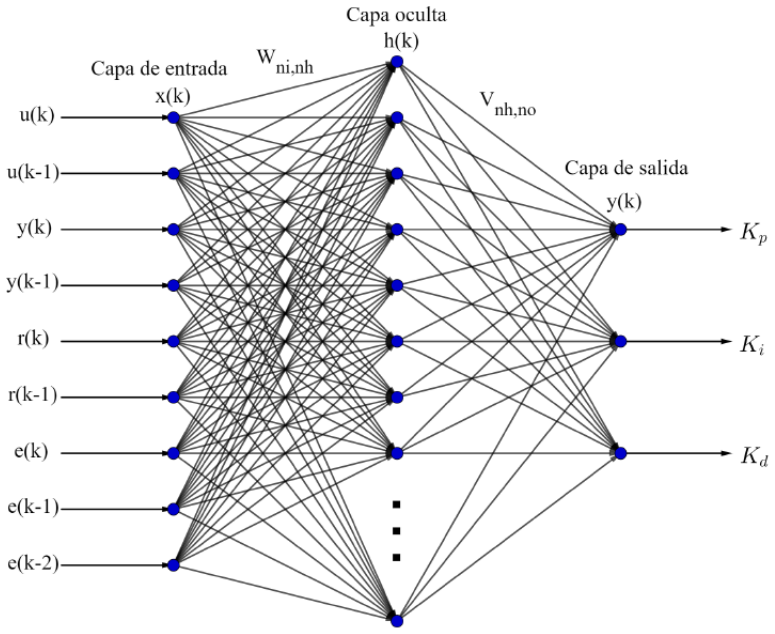


Figura 3.9: Esquema de la *RNA* para la obtención de las ganancias del controlador *PID*.

Un sistema no lineal cuya linealización exacta es fase mínima puede controlarse con un *PID* sintonizado por redes neuronales mediante el esquema de la Figura 4.23, i.e., para el

controlador PID en su forma de velocidad se tiene que :

$$u(k) = u(k-1) + K_P(e(k) - e(k-1)) + K_I e(k) + K_D(e(k) - 2e(k-1) + e(k-2)), \quad (3.62)$$

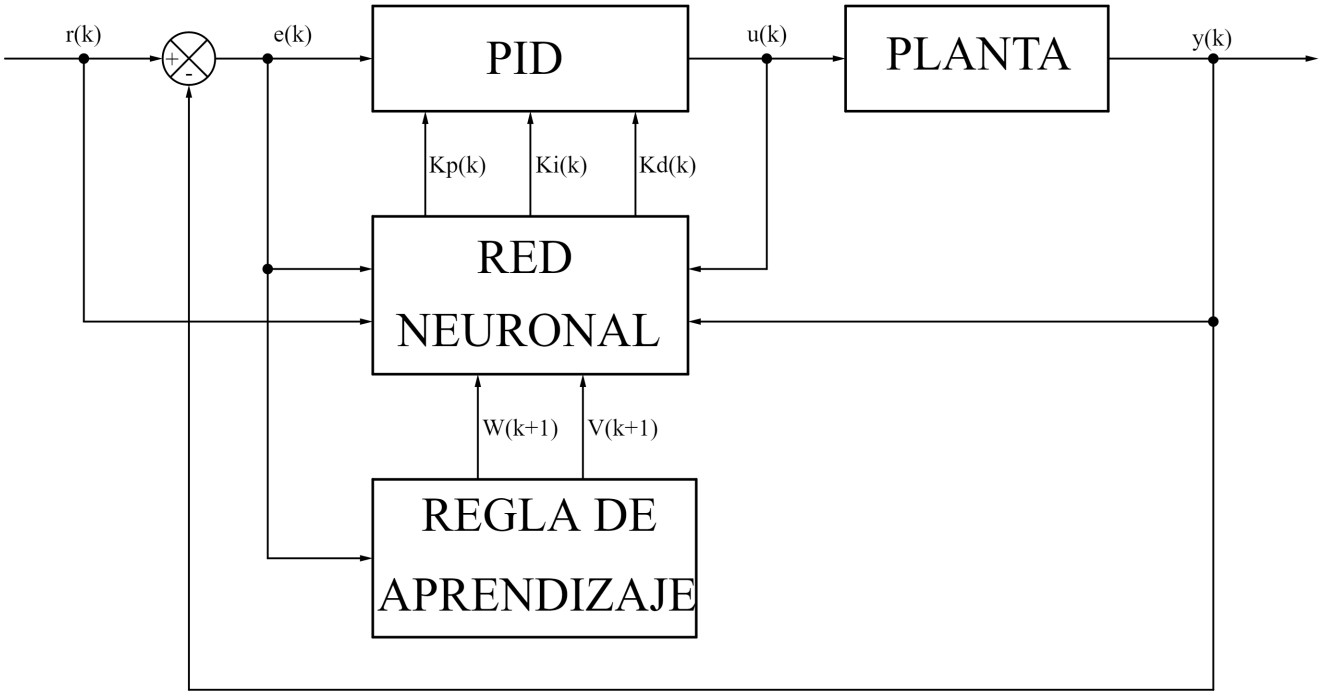


Figura 3.10: Esquema de control del sintonizador neuronal.

Este esquema utiliza un MLP que propaga la información desde la capa de entrada a la capa oculta como:

$$\begin{aligned} neth_j(k) &= \sum_{i=1}^{ni} W_{ij} \cdot x_i; \\ h_j(k) &= \Phi(neth_j(k)). \end{aligned} \quad (3.63)$$

En donde ni es la cantidad de entradas de la red, x son los datos de la capa de entrada y W es la matriz de pesos sinápticos que une a la capa de entrada con la capa oculta. De la misma manera, la información se propaga de la capa oculta a la capa de salida según:

$$net_o(k) = \sum_{j=1}^{nh} V_{jo} \cdot h_j. \tag{3.64}$$

Donde nh es la cantidad de nodos de la capa oculta, h es la respuesta de la capa oculta y V es la matriz de pesos sinápticos que une a la capa oculta con la capa de salida. Las salidas del MLP utilizado son las ganancias proporcional (K_P), integral (K_I) y derivativa (K_D), que están dadas por:

$$O_o(k) = \Gamma(net_o(k)). \tag{3.65}$$

Las funciones de activación Γ y Φ deben ser una función Sigmoide para asegurar que las ganancias del controlador PID sean positivas [88]. Además, se utiliza la siguiente función de costo:

$$J(k) = \frac{1}{2}e^2(k),$$

donde $e(k)$ es la señal de error, que representa la diferencia entre el punto de referencia r y la salida del sistema y en el instante k. La sintonización dinámica de las ganancias depende de una modificación del algoritmo BP para evitar el problema de caer en un mínimo local [88], la cual se describe por:

$$W(k+1) = W(k) - \eta\frac{\partial J(k)}{\partial W(k)} + \alpha\Delta W(k-1) + \beta W(k-2). \tag{3.66}$$

El gradiente del error está dado por $\frac{\partial J(k)}{\partial W(k)}$. Usando la regla de la cadena para descomponerlo con respecto a la capa de salida se tiene:

$$\frac{\partial J(k)}{\partial V_{jo}(k)} = \frac{\partial J(k)}{\partial y(k)} \cdot \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial O_o(k)} \cdot \frac{\partial O_o(k)}{\partial net_o(k)} \cdot \frac{\partial net_o(k)}{\partial V_{jo}(k)}, \tag{3.67}$$

donde el gradiente local de la capa de salida es:

$$\delta_o = \frac{\partial J(k)}{\partial y(k)} \cdot \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial O_o(k)} \cdot \frac{\partial O_o(k)}{\partial net_o(k)}, \tag{3.68}$$

con:

$$\begin{aligned} \frac{\partial J(k)}{\partial y(k)} &= -(r(k) - y(k)); \\ \frac{\partial O_o(k)}{\partial net_o(k)} &= \Gamma'(net_o). \end{aligned} \tag{3.69}$$

En el gradiente local de la capa de salida (3.67), $\frac{\partial y(k)}{\partial u(k)}$ representa el Jacobiano del sistema, y no es posible encontrarlo si los parámetros del sistema son desconocidos, por lo que es compensado con la constante de aprendizaje η. Además, $\frac{\partial u(k)}{\partial O_o(k)}$ depende del controlador PID digital (3.62), así:

$$\begin{aligned} \frac{\partial u(k)}{\partial O_1(k)} &= e(k) - e(k-1); \\ \frac{\partial u(k)}{\partial O_2(k)} &= e(k); \\ \frac{\partial u(k)}{\partial O_3(k)} &= e(k) - 2e(k-1) + e(k-2), \end{aligned} \tag{3.70}$$

donde las salidas O_1, O_2, O_3 representan a las ganancias K_p, K_i y K_d respectivamente. Así el gradiente local de la capa de salida se reescribe como:

$$\delta_o = -(r(k) - y(k)) \cdot \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial O_o(k)} \cdot \Gamma'(net_o), \tag{3.71}$$

La corrección de los pesos para la capa de salida se define como [27]:

$$\Delta V_{jo}(k) = \eta \delta_o(k) h_j. \tag{3.72}$$

La actualización de los pesos en la capa de salida para el sintonizador neuronal viene

dada por la siguiente ecuación:

$$V_{jo}(k+1) = V_{jo}(k) - \eta\delta_o h_j + \alpha\Delta V_{jo}(k-1) + \beta\Delta V_{jo}(k-2), \tag{3.73}$$

donde α y β se utilizan para evitar el problema de un mínimo local Mediante la regla de la cadena se descompone el gradiente de la capa oculta teniéndose:

$$\frac{\partial J(k)}{\partial W_{ij}(k)} = \frac{\partial J(k)}{\partial y(k)} \cdot \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial O_o(k)} \cdot \frac{\partial O_o(k)}{\partial net_o(k)} \cdot \frac{\partial net_o(k)}{\partial h_j(k)} \cdot \frac{\partial h_j(k)}{\partial neth_j(k)} \cdot \frac{\partial neth_j(k)}{\partial W_{ij}(k)}, \tag{3.74}$$

donde el gradiente local para la capa oculta se expresa como:

$$\delta_j = \left(\sum_{o=1}^{no} \delta_o \cdot V_{jo} \right) \cdot \Gamma'(neth_j). \tag{3.75}$$

La corrección en la capa oculta está dada por:

$$\Delta W_{ij}(k) = \eta\delta_j(k) x_i, \tag{3.76}$$

y los pesos se actualizan por [88]:

$$W_{ij}(k+1) = W_{ij}(k) - \eta\delta_j x_i + \alpha\Delta W_{ij}(k-1) + \beta W_{ij}(k-2). \tag{3.77}$$

La necesidad de conocer el Jacobiano del sistema o de compensarlo mediante la constante de aprendizaje η genera problemas en el comportamiento del esquema de autosintonización. Además, cómo se ha mencionado una de los características a considerar es el rechazo de perturbaciones, para ello se propone que la regla de aprendizaje anterior sea modificada con

base en el método de *MTA* para utilizar las ventajas del mismo, de tal manera que,

$$\begin{aligned} V(k+1) &= V(k) - \eta_k \delta_o h_j + \alpha \Delta V(k-1) + \beta \Delta V(k-2) + \beta_{w,k} \hat{X}_{W,k}; \\ W(k+1) &= W(k) - \eta_k \delta_j x_i + \alpha \Delta W(k-1) + \beta \Delta W(k-2) + \beta_{v,k} \hat{X}_{V,k}, \end{aligned} \tag{3.78}$$

donde η_k se determina con:

$$\eta_k = \frac{\eta}{1 + \|\Phi' x^T(k)\|^2}, 0 < \eta \leq 1,$$

donde Φ' representa la derivada de la función de activación. El uso de *MTA* permite la iteración entre controladores dotando de la capacidad de hacer uso del conocimiento previamente adquirido en forma de experiencia, acelerando el rechazo de la perturbación, mostrando un mejor desempeño en la minimización de la función de costo y con un menor gasto energético. Cabe mencionar que en este esquema la eficacia de la ley de aprendizaje modificada propuesta estará en función del desempeño de la tareas subyacentes de donde se pretende extraer el conocimiento, por lo que es conveniente hacer uso del algoritmo de búsqueda basado en la Transformada *Wavelet*.

Capítulo 4

Aplicaciones y simulaciones

En esta sección se compara el método de Meta-Aprendizaje BPM contra el método clásico entrenamiento BP y la retro-propagación variable en el tiempo denominado método (BPS, [89]. Los periodos de entrenamiento parten de seleccionar los hiper-parámetros de los modelos neuronales a través de prueba y error; en la comparación se observa que los algoritmos de BP y BPS tienen complicaciones en la obtención de resultados aceptables para minimizar la función de costo MSE, en la identificación de sistemas no lineales utilizando el modelo neuronal paralelo. Experimentalmente, se muestra que el método ML propuesto depende de una cantidad menor de elementos en el conjunto de datos para la etapa de entrenamiento. La eficacia del método ML se muestra al hacer pronósticos sobre series de tiempo caóticas como lo son los terremotos a corto plazo para la variable de Magnitud de un sismo. Además, se muestran las ventajas del uso del algoritmo de Meta-Transferencia de Aprendizaje, para la predicción de series de tiempo ante valores perdidos en la adquisición de datos y se hacen comparaciones con topologías de Aprendizaje Profundo con ML y TL. Para abordar el pronóstico de largo plazo es necesario hacer uso del algoritmo de Meta-Transferencia de Aprendizaje con el algoritmo de búsqueda de basado en la TW; se hacen comparaciones de estructuras de Aprendizaje Profundo, ML y TL.

4.1. Meta-Aprendizaje para la identificación de sistemas no lineales

Dado que el método propuesto ML se basa en el gradiente descendente, se considera una mejora a los resultados alcanzados por el clásico BP [87], ya que este algoritmo en ocasiones necesita una gran cantidad de información en la etapa de entrenamiento y una selección de los hiper-parámetros de aprendizaje como épocas, capas ocultas, nodos y condiciones iniciales para lograr un buen resultado en la identificación de sistemas no lineales dinámicos, considerando el modelo de simulación, estos parámetros se encuentran a través de prueba y error. Dado el número de parámetros a seleccionar, es preferible tener algoritmos que, por construcción, permitan una menor dependencia de la selección de los mismos para llevar acabo el aprendizaje en el modelo neuronal. Por ejemplo, el método de retro-propagación variable en el tiempo [89], donde el tamaño del paso η_k esta función del tiempo, permite una característica en la adaptabilidad del aprendizaje y elimina las épocas de entrenamiento en el proceso de aprendizaje. Además, la propiedad de estabilidad se obtiene mediante un análisis ISS. La importancia de contar con técnicas para la identificación de sistemas robustos a perturbaciones, facilita la implementación en aplicaciones del mundo real. También, respecto a los experimentos realizados, la disminución en la dependencia de los pesos iniciales es notable, en la Figura (4.1) se muestran 10 experimentos con diferentes condiciones iniciales para un MLP con tres capas ocultas, donde se observa cómo la diferencia de la norma de las matrices de los pesos sinápticos W_k y W^*, disminuye en función del término agregado por ML ($\beta_{W,k}X_{W,k}$), mostrando así la convergencia en una región ε a lo largo de la secuencia de datos de una serie de tiempo. La condición angular $Ac = 0.90$ para el método ML para los experimentos propuestos.

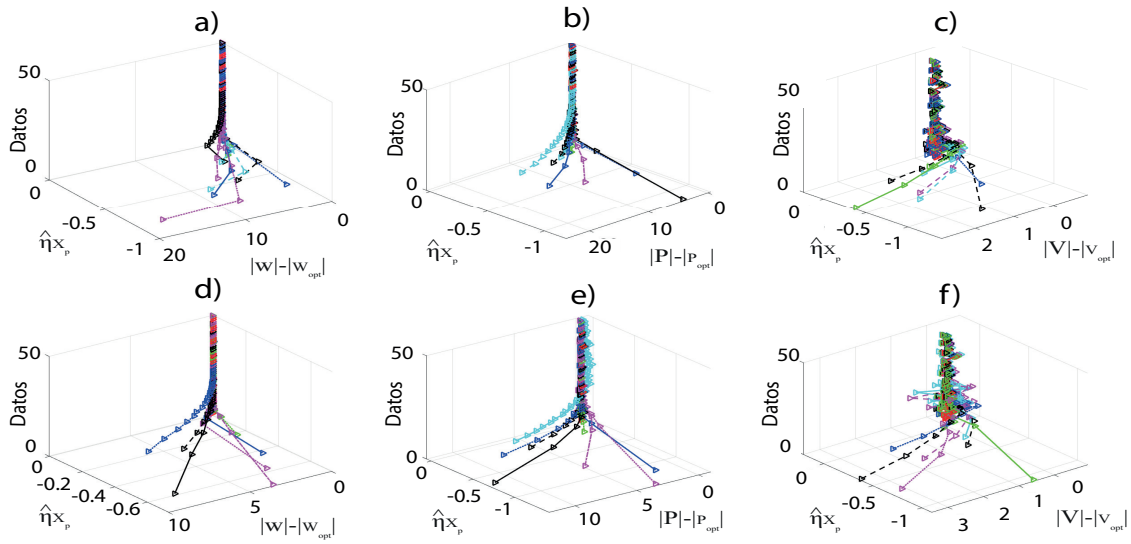


Figura 4.1: Ilustración de la convergencia con diferentes pesos iniciales.

4.1.1. Horno de Gas

El conjunto de datos está compuesto por 296 parejas de elementos [90], el tiempo de muestreo utilizado es de 9 segundos. El flujo del gas metano es la entrada $u(k)$, la concentración de CO_2 en la mezcla de gases bajo un suministro de aire constante es la salida $y(k)$.

Se usa la siguiente estructura para definir al sistema:

$$y\left(k\right) = \Phi_1\left[X(k)\right];$$
$$X(k) = [y(k-1), \cdots, y(k-4), u(k), \cdots, u(k-7)],$$

donde $\Phi_1(\cdot)$ es una función desconocida; se utiliza el "modelo de simulación" para construir la salida estimada de la red neuronal:

$$\hat{y}\left(k\right) = NN_1\left[\hat{X}\left(k\right)\right];$$
$$\hat{X}\left(k\right) = [\hat{y}(k-1), \cdots, \hat{y}(k-4), u(k), \cdots, u(k-7)],$$

la salida de la red neurona es denotada por $\hat{y}(k)$ y NN_1 es el modelo neuronal utilizado para identificar al sistema compuesto por el horno de gas.

Los valores de los vectores $X(k)$, $\hat{X}(k)$ y $u(k)$ son normalizados a través de la siguiente ecuación:

$$x(k) = \frac{x(k) - \text{mín}_k\{x(k)\}}{\text{máx}\{x(k)\} - \text{mín}_k\{x(k)\}}. \tag{4.1}$$

Se usan 200 datos para realizar la etapa de entrenamiento. Donde el modelo neuronal MLP usa 3 capas ocultas. Cada una de las capas tiene respectivamente 180 nodos, 120 nodos y 40 nodos, tal que: $W_k \in \mathbb{R}^{180x14}, V_k^1 \in \mathbb{R}^{120x180}, V_k^2 \in \mathbb{R}^{40x120}, V_k^3 \in \mathbb{R}^{1x40}$. La función de activación es $\phi(\cdot) = \tanh(\cdot)$ y $\phi'(\cdot) = \sec h^2(\cdot)$. Las constantes de aprendizaje son $\eta = 0.001$ y $\alpha = 0.0005$. Las condiciones iniciales para las matrices de pesos W_k, V_k^1, V_k^2 y V_k^3 son números aleatorios del intervalo $[-0.1, 0.1]$. Se añade una distribución normal para corromper las bases de datos en las etapas de entrenamiento y validación con un varianza de 0.05. Sólo fueron necesarios 120 datos para el entrenamiento con el algoritmo ML. Es importante señalar que los métodos BPM, BPS y BP, tienen la misma topología, condiciones iniciales y parámetros de aprendizaje.

Los resultados de la identificación son mostrados en la Figura (4.2) y ver Tabla (4.1), donde es posible observar que la respuesta obtenida por BPM tiene un mejor desempeño que los algoritmos utilizados para la comparación.

4.1.2. Sistema no lineal de primer orden

El siguiente ejemplo forma parte de los denominados problemas Benchmark, el cual es un sistema de primer orden no lineal, [90]:

$$y(k+1) = \frac{y(k)}{1 + y^2(k)} + u^3(k), \tag{4.2}$$

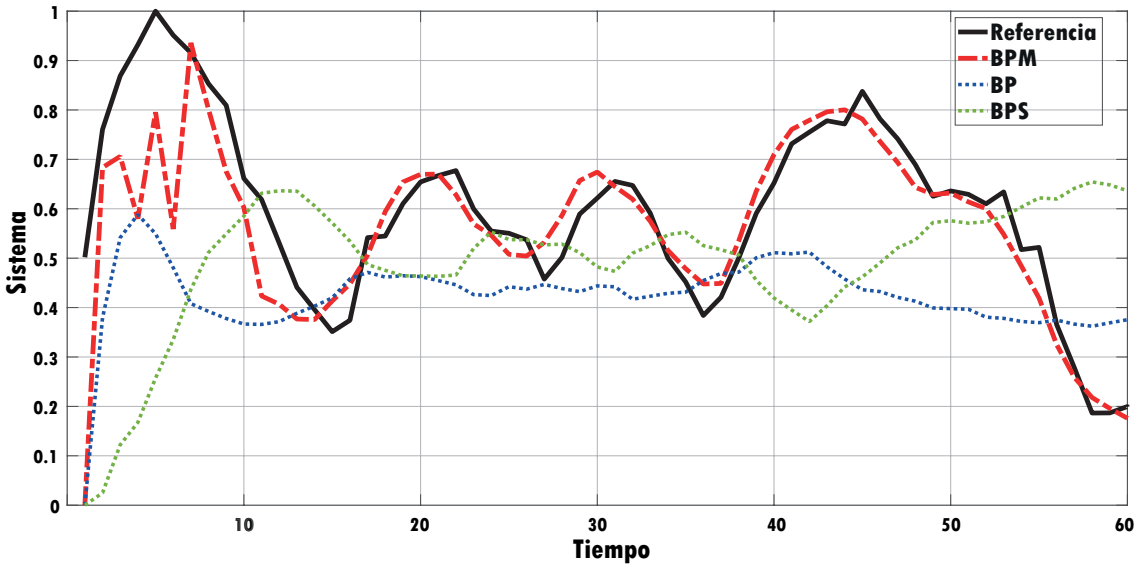


Figura 4.2: Comparación gráfica de los métodos propuestos para la identificación no paramétrica para el Horno de Gas.

donde $u(k)$ es una entrada periódica, se tiene una diferente A y B, para las etapas de entrenamiento y validación, tal que la entrada esta definida por:

$$u(k) = A \sin\left(\frac{\pi k}{50}\right) + B \sin\left(\frac{\pi k}{20}\right). \tag{4.3}$$

El sistema dinámico no lineal (4.2), puede ser reescrito como:

$$\begin{aligned} y\left(k\right) &= \Phi_2\left[x(k)\right]; \\ x(k) &= [y(k-1), \cdots, y(k-4), u(k-1), \cdots u(k-9)], \end{aligned}$$

donde Φ_2 es desconocida. El modelo paralelo esta definido por:

$$\begin{aligned} \hat{y}\left(k\right) &= NN_2\left[\hat{x}\left(k\right)\right]; \\ \hat{x}\left(k\right) &= [\hat{y}(k-1), \cdots, \hat{y}(k-4), u(k-1), \cdots u(k-9)]. \end{aligned}$$

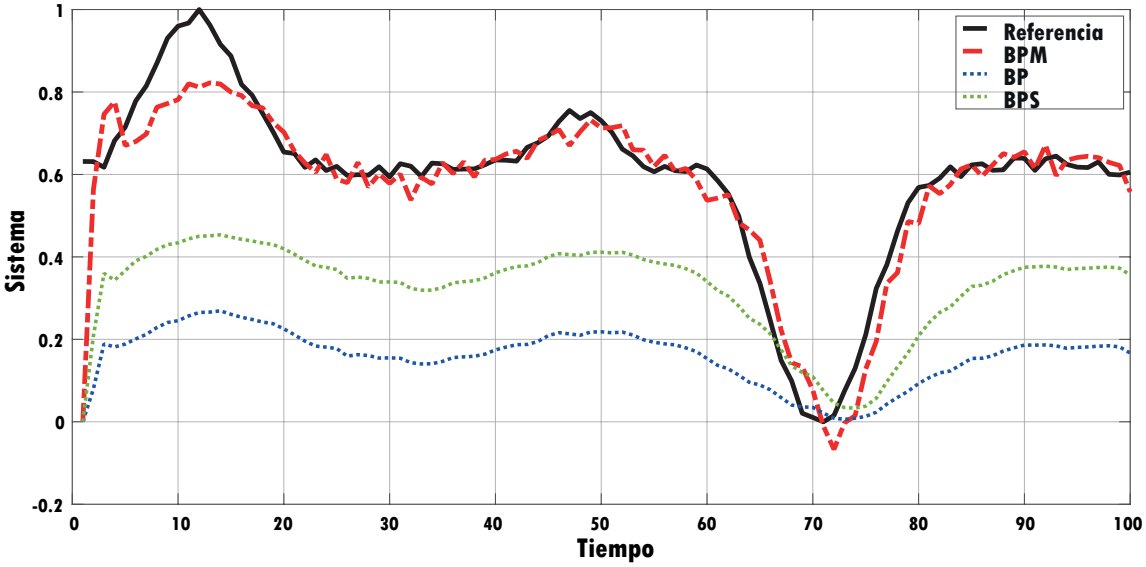


Figura 4.3: Comparación gráfica de la identificación no paramétrica para el Sistema de Primer Orden no lineal.

Se usan 80 datos para entrenar el modelo neuronal NN_2, este cuenta con 3 capas ocultas. Cada una de las capas tiene 100 nodos, 60 nodos, 20 nodos. Las constantes de aprendizaje son $\eta = 0.065$ y $\alpha = 0.02$. Los pesos iniciales son seleccionados aleatoriamente de acuerdo al intervalo $[-0.1, 0.1]$. La Figura (4.3) y la Tabla (4.1), muestran la comparación de los diferentes métodos BP, BPS y BPM, siendo el ultimo de estos el que demuestra una mejor aproximación al modelo real.

4.1.3. Sistema Wiener-Hammerstein

El conjunto de datos del problema Wiener-Hammerstein benchmark [90], es generado por un circuito eléctrico que se encuentra estructurado por tres bloques en cascada. Cada uno de los bloques tiene una particularidad especial: el primero corresponde a un sistema lineal, el segundo con una no linealidad estática y el tercero a otro sistema lineal. No hay medición directa a la no linealidad estática, porque se encuentra entre dos sistemas dinámicos internos

desconocidos. El conjunto de datos de referencia consta de pares de entrada / salida de $188,000$ elementos. Se usa el siguiente vector de entrada recursiva para describir al sistema:

$$y(k) = \Phi_3[x(k)];$$
$$x(k) = [y(k-1), \cdots, y(k-6), u(k), \cdots, u(k-19)].$$

El modelo paralelo utilizado para este problema está descrito por:

$$\hat{y}(k) = NN_3[\hat{x}(k)];$$
$$\hat{x}(k) = [\hat{y}(k-1), \cdots, \hat{y}(k-6), u(k), \cdots, u(k-19)].$$

Se usaron $10,000$ datos para entrenar el modelo neuronal y 100 datos para la etapa de prueba. Los parámetros de estructura del modelo neuronal son los mismos que NN_2. Las tasas de aprendizaje son $\eta = 0.065$ y $\alpha = 0.05$. La matriz de los pesos iniciales también son números aleatorios en $[-0.1, 0.1]$.

Los resultados de la prueba se muestran en la Figura (4.4). Entonces, el rendimiento de BPM es mejor que los algoritmos BP y BPS, ver Tabla (4.1).

Los datos utilizados son suficientes para que funcione el método ML. De tal forma que la cantidad de información requerida sea menor a la requerida por un método de aprendizaje de BP y BPS. Por lo tanto, esto es una ventaja para propósitos reales de implementación o resolución de problemas.

Finalmente, se muestra cómo el Meta-Aprendizaje afecta la dinámica de los pesos en el proceso de formación. La Figura (4.5) muestra el proceso de entrenamiento del modelo de Wiener-Hammerstein. Se observa que $\beta_{w,k}\hat{X}_{W,k}$ reduce la norma del error de los pesos en el instante k y el óptimo de los pesos en cada iteración, incluso si tienen ruido. Por lo tanto, el ML tiene una propiedad para evitar mínimos locales.

Al utilizar el error cuadrático medio para comparar los resultados se tiene que el método ML propuesto, se entiende que es posible generar una modelo de identificación no lineal a través de modelos de simulación. Como se muestra en Figuras (4.2)-(4.4), el modelo propuesto ofrece buenos resultados de modelado para las serie de tiempo y el sistema no lineal. También, tiene mejor robustez y adaptabilidad. Con lo expuesto en esta sección se ha

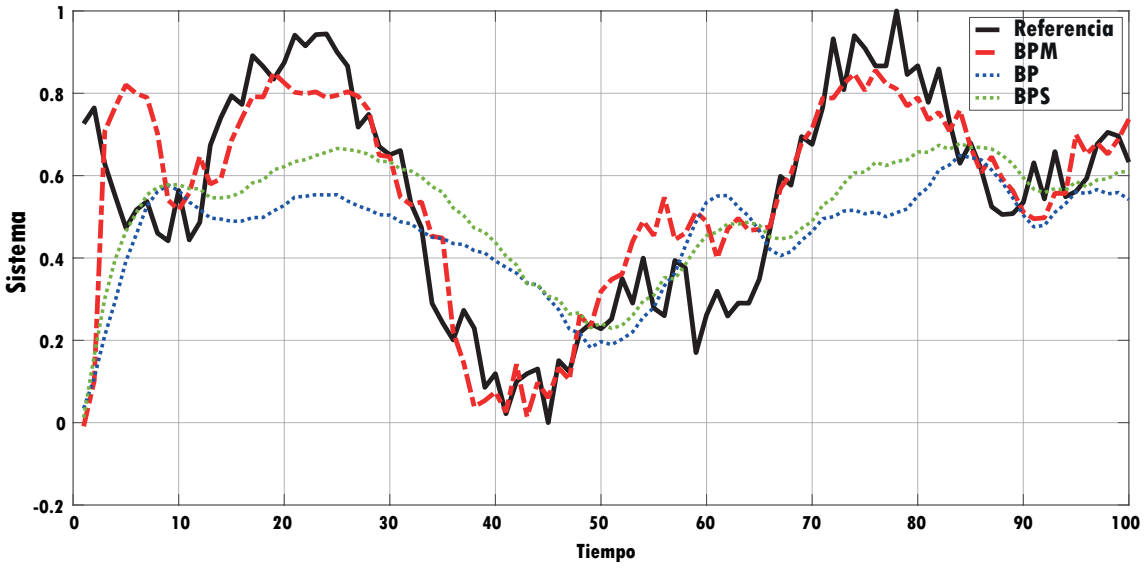


Figura 4.4: Validación de la identificación no paramétrica para el sistema Wiener-Hammerstein.

ilustrado como el método propuesto MTA es eficaz en la identificación de sistemas a través del modelo paralelo.

Tabla 4.1: Errores de Validación MSE (x10^{-3})

	Horno de Gas	Sistema No Lineal	Wiener-Hammerstein
BP	78.94	92.6	83.19
BPS	57.91	85.2	63.78
BPM	4.18	5.20	11.22

4.2. Meta-Aprendizaje para la predicción de magnitud de terremotos en corto plazo

La predicción de terremotos ha captado la atención de los científicos durante décadas. En particular, la predicción de terremotos fuertes siempre ha representado una desafío, no

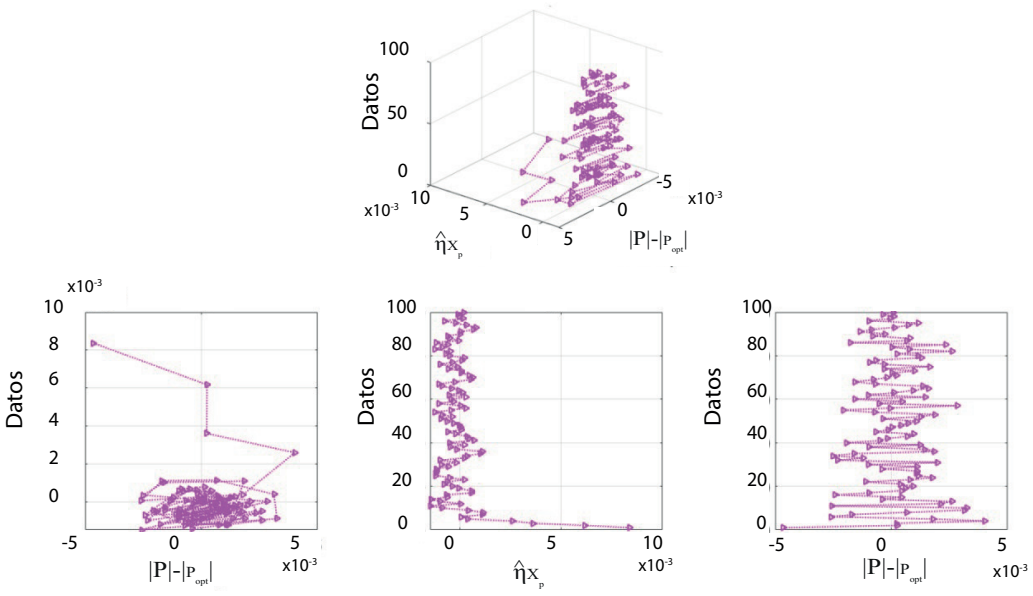


Figura 4.5: Ilustración de la convergencia de pesos ante la presencia de ruido en los datos.

sólo porque la investigación de grandes eventos, en general, proporciona los medios para una mejor comprensión de los mecanismos dinámicos y procesos sísmicos subyacentes, sino también porque fuertes terremotos pueden causar grandes daños estructurales y un elevado número de víctimas, especialmente en aquellas áreas con alto riesgo sísmico. En las últimas décadas, la previsión de terremotos se realizó principalmente considerando dos enfoques. Un enfoque consiste en estudiar la variación espacio-temporal de algunas variables geofísicas. Vinculado con los terremotos, como la variación del nivel de agua en los pozos [91]; [92] o los cambios anómalos geoeléctricos y magnéticos de señales medidas en áreas sísmicas [93]; [94]; [95]; [96], o una variación anómala del campo térmico medido por sensores satelitales [97], o las emisiones de radón [98]. El otro enfoque es investigar las secuencias del terremoto y centrarse en algunas propiedades dinámicas y estadísticas que caracterizan su evolución espacio-temporal, como el comportamiento fractal y multifractal [99], o las propiedades no extensivas basadas en Tsallis [100], o en características topológicas de las redes de terremotos [101], el uso del valor b de Gutenberg-Richter [102] o el modelo tipo epidémico que describe

la secuencia de réplicas de un sismo [103]. Recientemente, las RNA se han aplicado con éxito en sismología. En [104] desarrollaron una detección de terremotos, el método esta basado en un problema de clasificación supervisado que propone una red neuronal convolucional para detección de terremotos y su ubicación a partir de sismogramas, construido a partir de los avances en el aprendizaje profundo. Las redes neuronales artificiales fueron utilizadas para construir un sistema de clasificación para la detección de eventos sísmicos; su algoritmo era más preciso que los algoritmos convencionales basados únicamente en el umbral STA / LTA [105], [106]. También se han empleado redes neuronales para la predicción de terremotos, no sólo para pronosticar terremotos de laboratorio [107], sino también para eventos reales. En [108], desarrollaron un enfoque de red neuronal probabilística para pronosticar la magnitud del evento más fuerte en un dominio espacio-tiempo predefinido en una región sísmica. K. M. Asim [109] modeló la relación entre parámetros sísmicos y ocurrencias de terremotos aplicando cuatro diferentes enfoques de redes neuronales (red neuronal de reconocimiento de patrones, conjunto de impulso de red neuronal recurrente, bosque aleatorio y de clasificador con programación lineal clasificador). Se llevaron a cabo varios otros estudios para predecir parámetros a partir de fenómenos sísmicos mediante redes neuronales artificiales [110], [111], [112], [113], y mediante el uso de redes neuronales profundas [114], [115], [116], [117]. En estos estudios, la magnitud se predijo sólo para el próximo evento.

La predicción de terremotos es todavía un tema muy debatido entre la comunidad del campo sismológico. Este tema es aún más desafiante, si se considera que la secuencia de magnitudes es dinámicamente bastante intermitente. Este estudio se centra en la predicción de terremotos en multi-horizonte de magnitud, lo que significa predecir la magnitud de un terremoto que ocurre después de j terremotos anteriores, utilizando el modelo (3.4). Aplicando el enfoque de ML para pronosticar las magnitudes de los terremotos ocurridos en Italia de mayo a agosto de 2018 los datos utilizados, son extraídos de la base de datos disponible públicamente en: cnt.rm.ingv.it/en/iside. El catálogo usado está completo para eventos con una magnitud mayor a 1.5 [118]. Se usa una red neuronal de dos capas para pronosticar la magnitud del terremoto.

Los pasos para realizar el experimento se muestra a continuación:

1. Se selecciona un conjunto de eventos de magnitud como conjunto de datos de entrenamiento.
2. Los parámetros η y α se eligen al azar.
3. Los pesos iniciales se eligen aleatoriamente dentro del rango [-1,1].
4. Se ejecuta el modelo neuronal con un cierto número de épocas.
5. El índice J se obtiene en la etapa de prueba.
6. Los pasos 4 y 5 se repiten hasta que se minimiza el índice J en la etapa de prueba.
7. De la iteración que minimiza el índice J, los pesos iniciales se mantienen W_i y los pesos finales W^*.
8. La RNA se ejecuta usando el algoritmo (1) y en cada iteración los parámetros ($\beta_{W,k} X_{W,k}$.
9. El índice J se obtiene en la etapa de prueba.

La Figura (4.6), ilustra como el método es implementado para optimizar el desempeño de un modelo neuronal propuesto. Se usaron diferentes conjuntos de entrenamiento de terremotos ocurridos en Italia:

1. A partir de abril de 2017 a abril de 2018, para pronosticar magnitudes superiores a 3.5.
2. A partir de abril 2016 a febrero de 2017, para pronosticar magnitudes superiores a 4.0.
3. A partir de enero 2007 a diciembre de 2015, para pronosticar magnitudes superiores a 4.5.

Los diferentes conjuntos de entrenamiento dependen de la magnitud mínima de los eventos a pronosticar; dado que el número de eventos disminuye con la magnitud mínima, cuanto mayor sea la magnitud mínima, más largo será el conjunto de entrenamiento. Se analizan diferentes casos que se muestran a continuación, donde M indica la magnitud de los terremotos y j el número de eventos pronosticados a partir del evento actual.

Los datos de entrenamiento para $M > 3.5$ y $M > 4$ se muestran en la Figura (4.7).

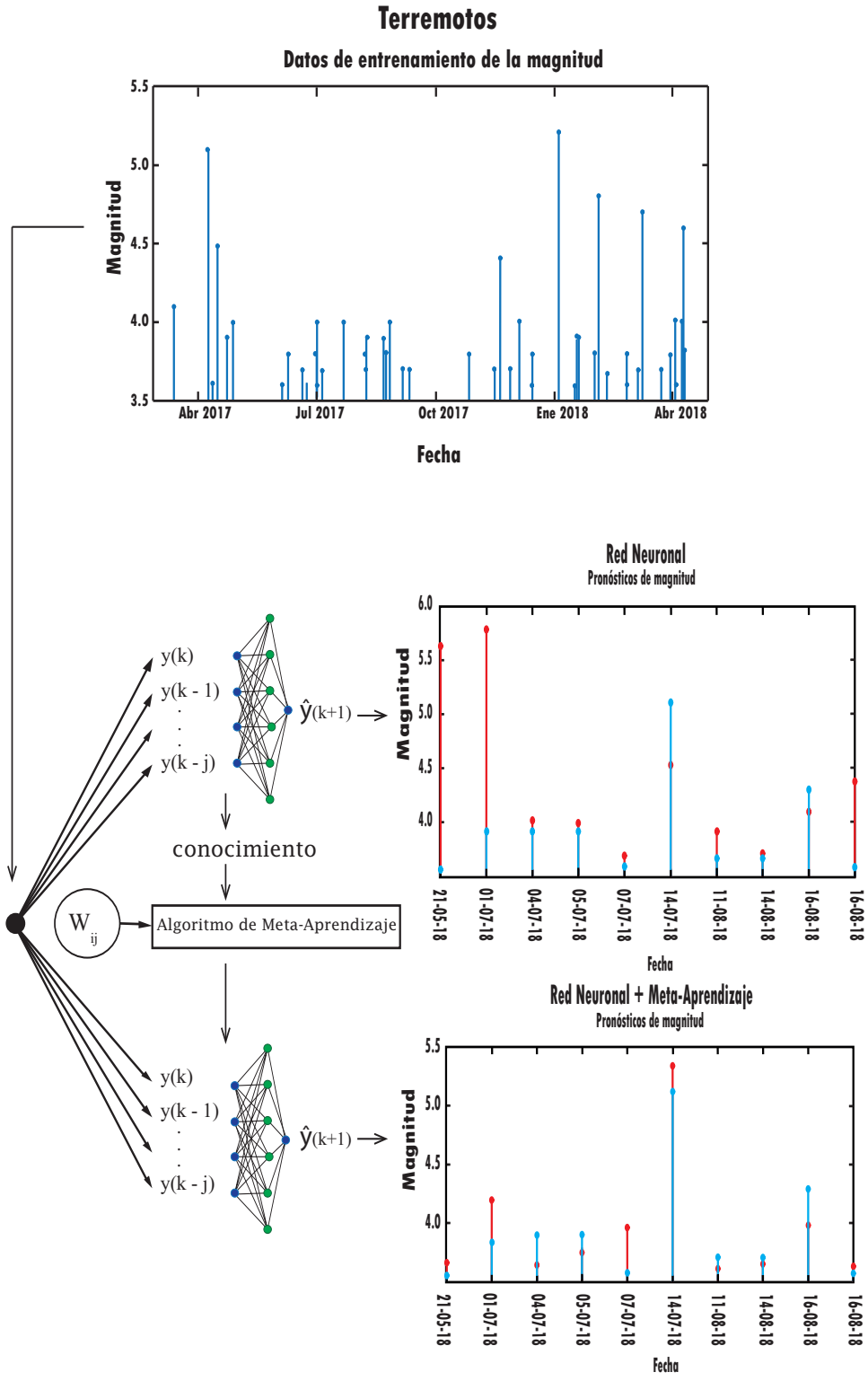


Figura 4.6: Ilustración de la convergencia con diferentes pesos iniciales.

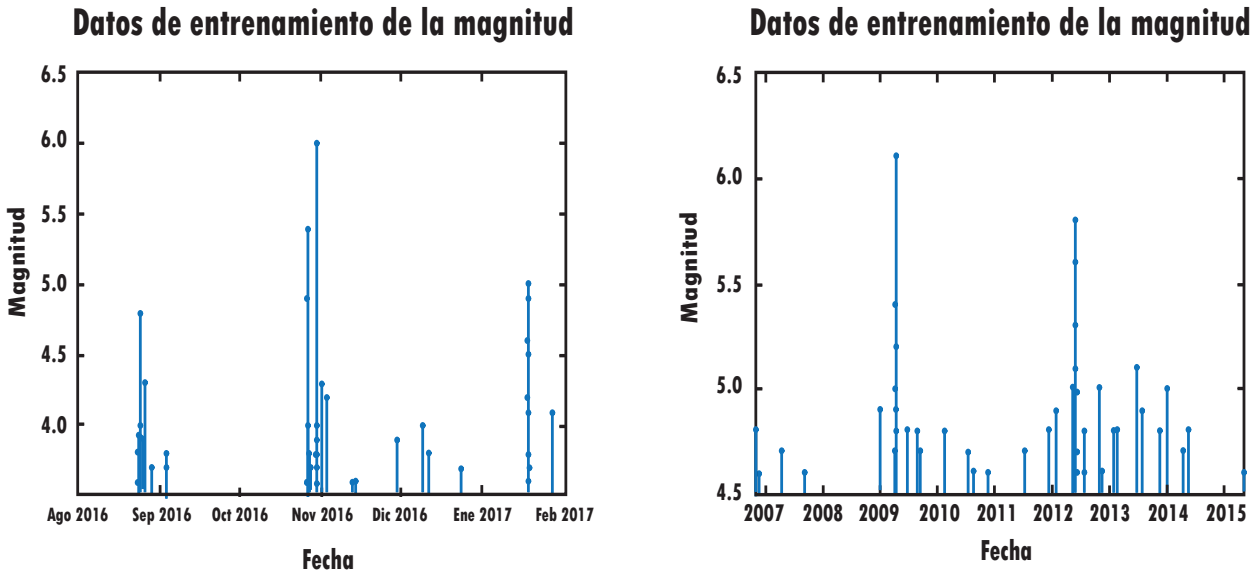


Figura 4.7: Datos de entrenamiento para $M > 3.5$ y $M > 4$.

- $M > 3.5$ y $j = 2$

  Del modelo (3.7) la predicción en multi-horizonte esta dado por:

$$\hat{y}(k+2) = NN[y(k), ..., y(k-5)],$$

  donde el vector de entrada $X(k) = y(k), ..., y(k-5)$. El segundo evento $y(k+2)$ a partir del actual $y(k)$ se puede puede predecir, porque los datos $y(k), ..., y(k-n)$ son conocidos, también es posible predecir $y(k+1)$ de modo que:

$$\hat{y}(k+1) = NN[y(k), ..., y(k-6)].$$

  El modelo neuronal tiene una capa oculta y 10 neuronas, los parámetros de aprendizaje se eligen como η=0.9946 y α=0.1881, mientras que las condiciones iniciales de los pesos se eligen aleatoriamente dentro del rango [-1,1]. La Figura (4.8) muestra los resultados de la predicción con el método ML. Aquí, la magnitud pronosticada (rojo)

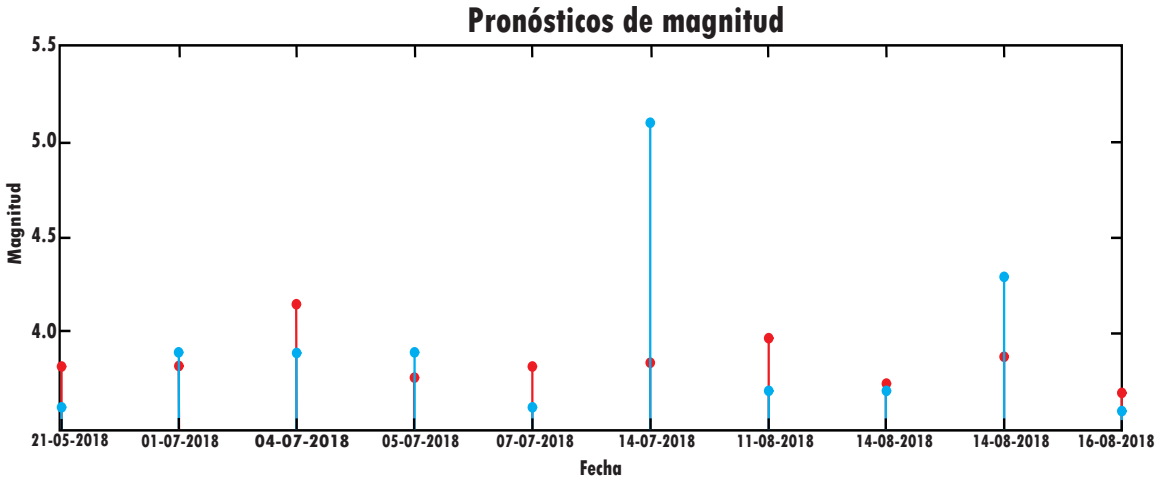


Figura 4.8: Predicción de magnitud del modelo ML para $M > 3.5$ y $j = 2$.

de los primeros 10 eventos (azul) ocurren después del período del conjunto de datos de entrenamiento. Para el modelo RNA su desempeño del índice $MSE = 1.8811$; esto indica que RNA tiene un desempeño muy pobre. El ML mejora la precisión del pronóstico al hacer una corrección de los pesos; de hecho, para ML tiene un $MSE = 0.0665$ y el rendimiento ha mejorado significativamente.

- $M > 3.5$ y $j = 3$

  Del modelo (3.7) la predicción en multi-horizonte esta dado por:

$$\hat{y}(k+3) = NN[y(k), ..., y(k-5)],$$

  donde el vector de entrada $X(k) = y(k), ..., y(k-5)$. El tercer evento $y(k+3)$ a partir del actual $y(k)$ se puede puede predecir, porque los datos $y(k), ..., y(k-n)$ son conocidos, también es posible pronosticar $y(k+1)$ y $y(k+2)$ de modo que:

$$\hat{y}(k+2) = NN[y(k), ..., y(k-6)];$$
$$\hat{y}(k+1) = NN[y(k), ..., y(k-7)].$$

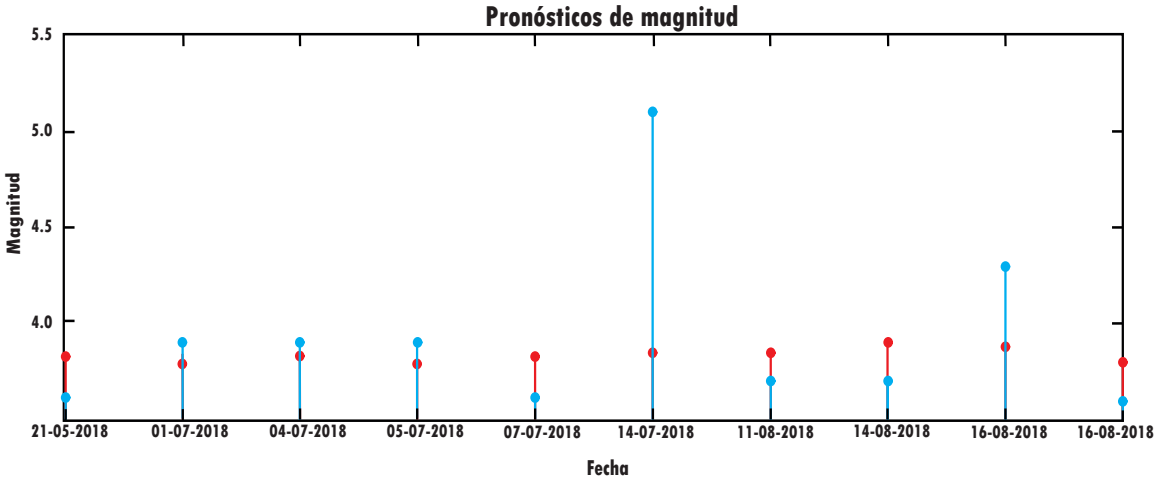


Figura 4.9: Predicción de magnitud del modelo ML para $M > 3.5$ y $j = 3$.

La Figura (4.9) muestra los resultados de la predicción del métodoML. Mientras que el error de predicción usando RNA es $MSE = 2.2781$ (Figura 7), mientras que el método ML tiene MSE = 0.3826, por lo tanto tiene una mejor predicción en promedio que RNA.

- $M > 3.5$ y $j = 4$ en este caso, el modelo RNA es:

$$\hat{y}(k+4) = NN[y(k), ..., y(k-5)].$$

El error de predicción de la RNA es MSE= 12.6642, mientras que el del ML es MSE = 0.4966, lo que indica que ML también es mejor que RNA bajo las condiciones propuestas. Es posible observar, ver Figura (4.10), que el modelo RNA ya no logra trabajar adecuadamente ya que con la información disponible en la etapa de entrenamiento no logra un buen desempeño en la etapa de validación. Por otro lado, el ML mantiene sus propiedades y puede seguir pronosticando, por lo que su eficacia queda ilustrada, ver Figura (4.11).

- Se considera el caso en el que los eventos tienen una magnitud $M > 4$. El conjunto de

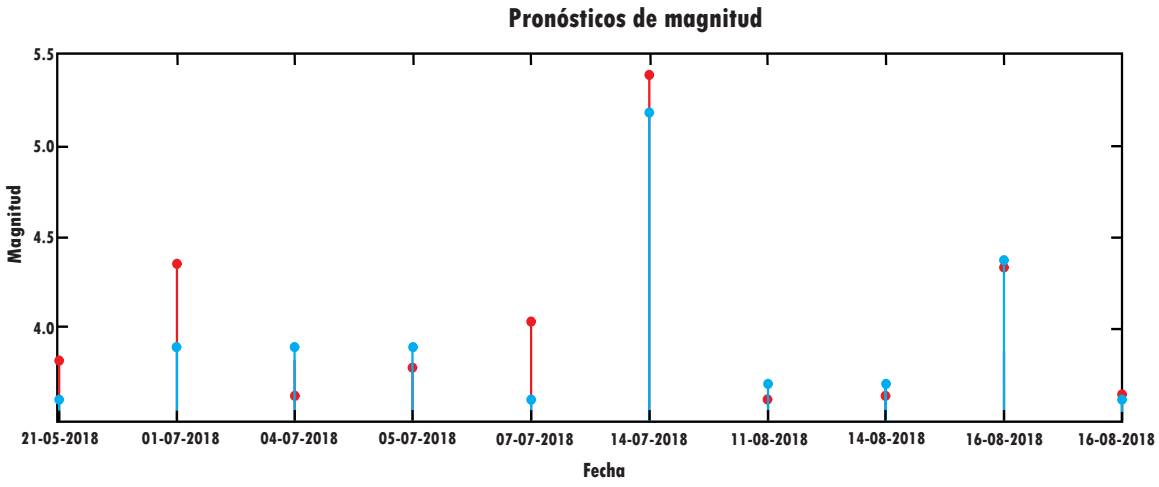


Figura 4.10: Predicción de magnitud del modelo RNA para $M > 3.5$ y $j = 4$.

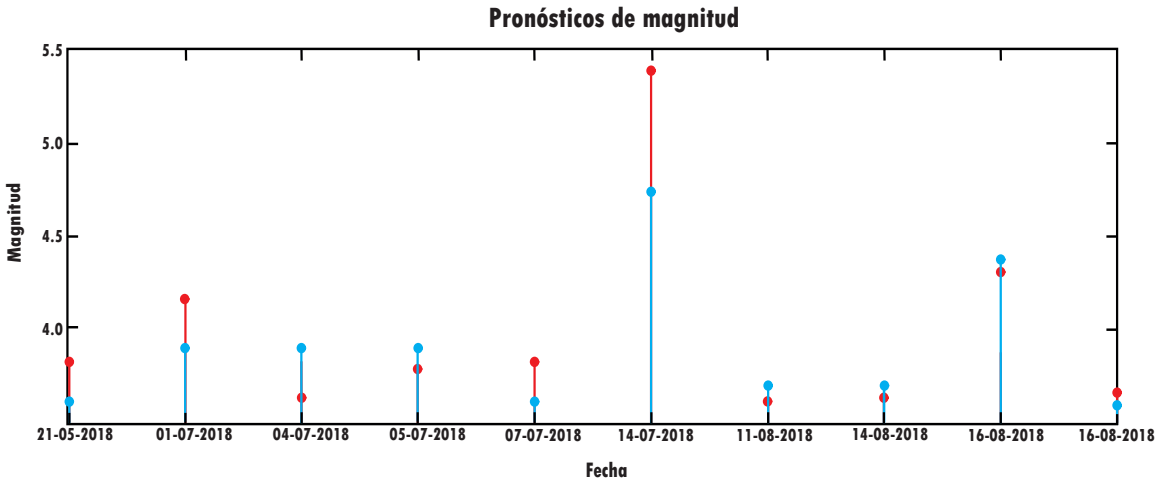


Figura 4.11: Predicción de magnitud del modelo ML para $M > 3.5$ y $j = 4$.

datos de entrenamiento, ver Figura (4.7), se caracteriza por una distribución claramente diferente al que se muestra en la Figura (4.6), esto se vuelve más complejo. El conjunto de entrenamiento aparece como compuesto por pocos conjuntos dominados por un gran evento en cada grupo. Al compararse ambos métodos RNA y ML sólo para $j = 2$. Los resultados de la predicción: para RNA el MSE= 1.2644, mientras que para ML, el MSE= 1.1685, es claro el decaimiento en el desempeño, por lo que podría decirse que la falta de información para llevar a cabo el entrenamiento es evidente y por lo tanto insuficiente.

Los métodos clásicos de aprendizaje automático para predecir la magnitud del terremoto puede funcionar bien para eventos inmediatos, pero no en el enfoque del multi-horizonte. El ML en ciencias de la computación utiliza diferentes tipos de desempeño, medidas y patrones de los datos de aprendizaje para obtener los datos latentes características. Este pronóstico de varios pasos de la magnitud del terremoto aplica el modelo de la RNA y el Meta-Aprendizaje. La predicción del método ML modelo se aplica con éxito para predecir las magnitudes de los terremotos de Italia se separan en tres umbrales diferentes de magnitud. Los resultados experimentales muestran que el modelo propuesto tiene un rendimiento mucho mejor que el métodos clásicos de aprendizaje automático para predecir varios terremotos en el futuro.

4.3. TL para la predicción de contaminación ambiental con pérdida de información

Una de las características más importantes de las RNA, al ser modelos de caja negra, es que pueden representar dinámicas no lineales con gran facilidad ya que en la mayoría de las implementaciones en aplicaciones del mundo real no existe un marco teórico completo debido a la naturaleza de los sistemas. En comparación con los modelos mecanicistas, las RNA necesitan una menor cantidad de datos [119]. Estos modelos no requieren suposiciones previas sobre las características probabilísticas de los datos [120]. En sentido general, una pregunta para las RNA es ¿Cuántos datos se necesitan para completar la etapa de entrenamiento y

dar buenos resultados?, de hecho es imposible responder ya que hay demasiados factores que determinan el correcto desempeño de una RNA. Para la tarea de predicción, se necesitan eventos pasados para estimar resultados futuros. Uno de los problemas más comunes para este tipo de tareas es la falta de información, esto puede provocar que una RNA se estanque en un mínimo local y su predicción sería inexacta. En la previsión de contaminantes atmosféricos, la ausencia de información en las estaciones de monitoreo ambiental es una situación recurrente.

Algunas formas de evitar estos problemas se tratan en [121]. Sin embargo, la más utilizada es la eliminación de los datos y el desplazamiento del uso de los datos pasados hasta que satisfaga la minimización de una función de costo en una RNA. Esto no es del todo correcto, debido a las condiciones ambientales y meteorológicas; Así como las actividades humanas en las ciudades cambian de un momento a otro, por supuesto el ejemplo de la pandemia provocada por el virus SARS-COV-2, que ha reducido la movilidad y la humanidad ha tenido que cambiar su comportamiento y por tanto su propia dinámica. Las fallas mecánicas, de medición y de almacenamiento en las estaciones de monitoreo ambiental representan un problema para el uso de métodos de pronóstico de contaminantes atmosféricos, ya que la cantidad de datos disponibles puede no ser suficiente para obtener los resultados esperados, de lo contrario se limita a pronósticos a corto plazo.

En [122], la contribución se hace a partir de una revisión de los trabajos y esfuerzos realizados por los investigadores para realizar la previsión de contaminantes atmosféricos. Sin embargo, cuando se trata de averiguar factores importantes, como la cantidad de información utilizada para las etapas de entrenamiento, así como las hiper-parámetros de una RNA, se encontró la siguiente información relevante Tabla (4.2).

En muchos de los experimentos hay características desconocidas, como el factor de aprendizaje, las épocas y lo más importante la cantidad de datos perdidos por fallas en las estaciones de monitoreo. Esto puede tener un impacto importante si se lleva a cabo una implementación, ya que su viabilidad se verá comprometida de acuerdo con la disponibilidad de información.

En México, una pequeña cantidad de trabajos para el pronóstico de series de tiempo basados en RNA se han hecho en, [136], [137], [138]. Estos trabajos abordan las series

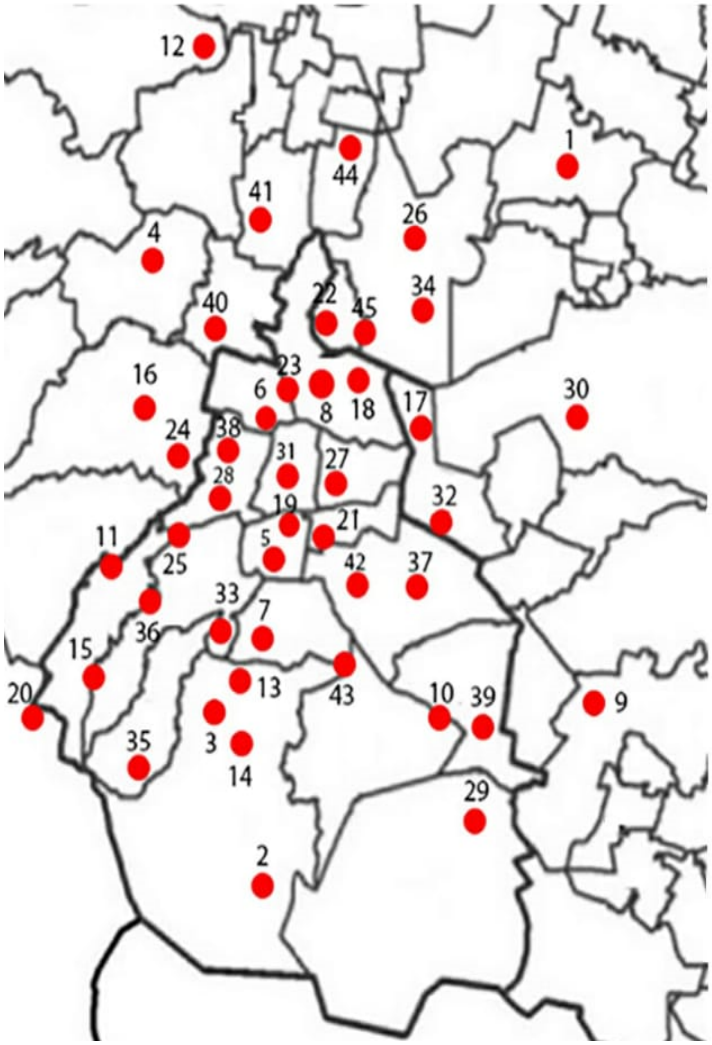
Tabla 4.2: Descripción general del pronóstico de la serie temporal para PM_{10}

Número de Referencia	Tiempo de Muestreo	Escala de pronóstico	Número de entradas	Datos de entrenamiento	Datos de prueba	Número de capas ocultas	Número de nodos	Función de activación	η	α	Épocas	% de datos faltantes	Parámetro R
[123]	Mensual	-	5	72	12	1	20	tansigmoidal	0.01-1	0.5	5000	-	0.7
[124]	Diario	Promedio y máximo un día posterior	25	1460	365	-	-	-	-	-	-	-	0.65
[125]	Diario	Un día	5	488	244	2	5	Tanh	-	-	-	15	0.8
[126]	Diario	Un día	4	Validación cruzada aleatoria	Validación cruzada aleatoria	1	-	Tanh	0.0001-1	-	-	-	0.88
[119]	Diario	Un día adelante	6	1460	365	1	4	-	-	-	-	Promediado	0.67-0.81
[127]	Cada hora	24 h	8	13140	4380	1	7	Logística	-	-	-	-	0.7-0.82
[127]	Cada hora	24 h	8	13140	4380	1	6	Logística	-	-	-	-	0.65-0.83
[128]	Diario	Máximo un día posterior	18	150	90	1	7	-	-	-	-	0	-
[129]	Diario	Máximo un día posterior	5	722	372	1	3	-	-	-	-	25	0.78
[120]	Cada hora	Cada hora	16	495	42	1	8	Sigmoidal	0.3	0.3	-	2	0.912
[130]	Cada hora	Una hora adelante	7	-	Random	1	9-36	Logística	0.1	0.3	5000	2-11	0.72
[131]	Cada hora	Máximo un día adelante	15	12800	2240	1	26	-	-	-	-	7	0.8-0.87
[132]	Cada hora	24 h	5	-	-	1	3	Tanh	-	-	-	-	0.61
[133]	Máximo diario	Un dia adelante	27	500	150	1	8	Sigmoidal	-	-	-	0	-
[134]	Diario	Maximum one day ahead	5	2000	650	1	10	Tanh-sigmoidal	-	-	1000	35	0.05-0.72
[135]	Diario	-	9	240	125	1	-	-	-	-	-	Promediado	0.68

de tiempo espacio-temporales pronóstico, en el que tienen como entradas, por ejemplo, la velocidad y la dirección del aire, la temperatura. Dos de estos trabajos obtienen sus datos a partir de un sistema de monitoreo con sede en Salamanca, donde hay gran actividad industrial, por lo que es de interés realizar análisis de series de tiempo para el componente PM_{10}.

La red de monitoreo ambiental de la ciudad de México [139] tiene 43 Estaciones de monitoreo ambiental. Pero para el componente de PM_{10}, sólo 27 estaciones de monitoreo ambiental son capaces de medir este elemento. De acuerdo con los datos de 2020, 5 presentaron un completo fallo a través del año; 7 estaciones presentaron al menos el 50 por ciento de fallas durante el año y el resto tiene un intervalo de falla entre (4.7-33) por ciento.

En la Ciudad de México, los contaminantes del aire como O_3, NO_2, SO_2, PM_{10} y $PM_{2.5}$ se miden mediante una red de monitoreo ambiental en toda la ciudad, consulte Figura (4.12). Por ejemplo, 27 estaciones de monitoreo pueden detectar el PM_{10}, en la Figura (4.13) muestra la frecuencia de fallas en porcentaje de las estaciones de monitoreo de la Ciudad de México en 2020. El color rojo corresponde al porcentaje total de fallas (5 estaciones o 18.51 %). El color naranja muestra las estaciones que presentan un alto porcentaje de falla (7 estaciones o 25,92 %). El resto (en azul son las estaciones con falla de bajo nivel porcentaje aproximadamente 50



Clave		Nombre	Alcaldía o municipio	Entidad
1	ACO	Acolman	Acolman	EDOMEX
2	AJU	Ajusco	Tlalpan	CDMX
3	AJM	Ajusco Medio	Tlalpan	CDMX
4	ATI	Atizapán	Atizapán de Zaragoza	EDOMEX
5	BJU	Benito Juarez	Benito Juárez	CDMX
6	CAM	Camarones	Azcapotzalco	CDMX
7	CCA	Centro de Ciencias de la Atmósfera	Coyoacán	CDMX
8	TEC	Cerro del Tepeyac	Gustavo A. Madero	CDMX
9	CHO	Chalco	Chalco	EDOMEX
10	COR	CORENA	Xochimilco	CDMX
11	CUA	Cuajimalpa	Cuajimalpa de Morelos	CDMX
12	CUT	Cuautitlán	Tepotzotlán	EDOMEX
13	DIC	Diconsa	Tlalpan	CDMX
14	EAJ	Ecoguardas Ajusco	Tlalpan	CDMX
15	EDL	Ex Convento Desierto de los Leones	Cuajimalpa de Morelos	CDMX
16	FAC	FES Acatlán	Naucalpan de Juárez	EDOMEX
17	FAR	FES Aragòn	Nezahualcóyotl	EDOMEX
18	GAM	Gustavo A. Madero	Gustavo A. Madero	CDMX
19	HGM	Hospital General de México	Cuauhtémoc	CDMX
20	INN	Investigaciones Nucleares	Ocoyoacac	EDOMEX
21	IZT	Iztacalco	Iztacalco	CDMX
22	LPR	La Presa	Tlalnepantla de Baz	EDOMEX
23	LAA	Laboratorio de Análisis Ambiental	Gustavo A. Madero	CDMX
24	IBM	Legaria	Miguel Hidalgo	CDMX
25	LOM	Lomas	Miguel Hidalgo	CDMX
26	LLA	Los Laureles	Ecatepec de Morelos	EDOMEX
27	MER	Merced	Venustiano Carranza	CDMX
28	MGH	Miguel Hidalgo	Miguel Hidalgo	CDMX
29	MPA	Milpa Alta	Milpa Alta	CDMX
30	MON	Montecillo	Texcoco	EDOMEX
31	MCM	Museo de la Ciudad de México	Cuauhtémoc	CDMX
32	NEZ	Nezahualcóyotl	Nezahualcóyotl	EDOMEX
33	PED	Pedregal	Álvaro Obregón	CDMX
34	SAG	San Agustin	Ecatepec de Morelos	EDOMEX
35	SNT	San Nicolás Totolapan	La Magdalena Contreras	CDMX
36	SFE	Santa Fe	Cuajimalpa de Morelos	CDMX
37	SAC	Santiago Acahualtepec	Iztapalapa	CDMX
38	SHA	Secretaria de Hacienda	Miguel Hidalgo	CDMX
39	TAH	Tláhuac	Xochimilco	CDMX
40	TLA	Tlalnepantla	Tlalnepantla de Baz	EDOMEX
41	TLI	Tultitlán	Tultitlán	EDOMEX
42	UIZ	UAM Iztapalapa	Iztapalapa	CDMX
43	UAX	UAM Xochimilco	Coyoacán	CDMX
44	VIF	Villa de las Flores	Coacalco de Berriozábal	EDOMEX
45	XAL	Xalostoc	Ecatepec de Morelos	EDOMEX

Figura 4.12: Estaciones de monitoreo ambiental en la Ciudad de México.

% de la red de monitoreo ambiental PM_{10} son inconsistentes, [139]. Los conjuntos de datos para una serie de tiempo pronosticada por los medios de una RNA pueden ser insuficientes para lograr la predicción del comportamiento del contaminante, la situación puede llevar al MLP a incurrir en los problemas comunes descrito en las secciones anteriores. Las mediciones se realizan cada hora.

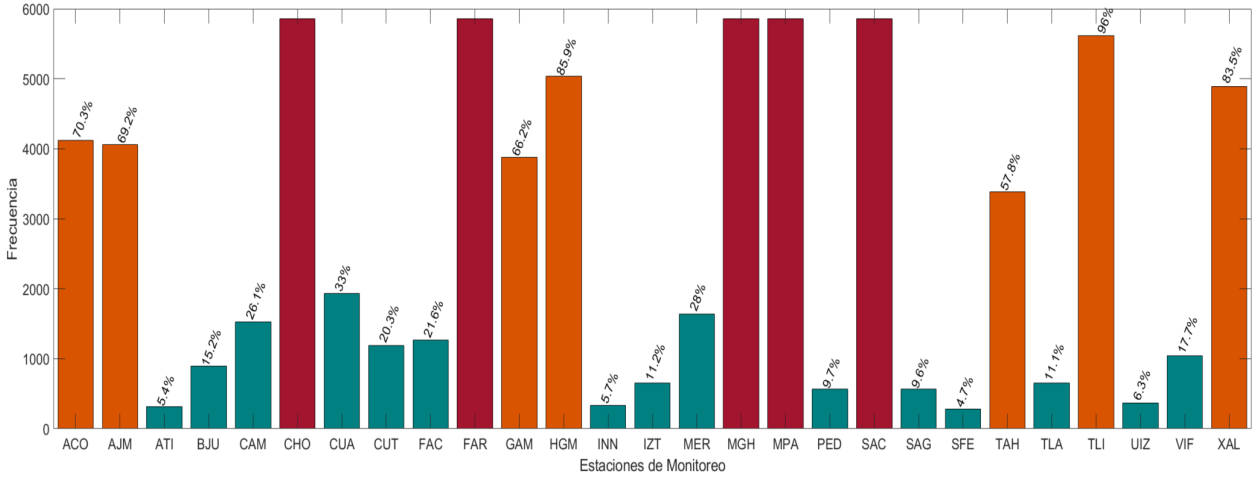


Figura 4.13: Frecuencia de fallas en porcentaje de las estaciones de monitoreo de la Ciudad de México en 2020 para el elemento PM_{10}.

Para ilustrar este método se propone el siguiente problema, para el contaminante ambiental PM_{10} la estación de monitoreo denominada “ACO" presenta un porcentaje de fallas y/o pérdida de información del 70.3 % en 2020. Cuándo ocurre este problema, es común eliminar información no útil y hacer uso de datos históricos de otros años para proceder con el entrenamiento de la *RNA* y luego generalizar el método a través del pronóstico de la serie temporal en un período determinado o instante futuro. En condiciones normales, esta suele ser una solución media de la que se obtienen resultados aceptables. Sin embargo, hay eventos externos que no se pueden predecir y que pueden afectar la técnica antes mencionada, i.e., la pandemia global provocada por el virus SARS-COV-2, que entre los innumerables efectos que ha dejado a la humanidad, está la parálisis que han sufrido distintas regiones del mundo por el encierro. Este tipo de eventos del mundo real obviamente tendrán un impacto negativo en la predicción de series de tiempo, porque muchos de los métodos o técnicas utilizados dependen de ciertas características (estacionalidad, tendencia, valores atípicos y varianza) de la información utilizada para el entrenamiento o parametrización de las series. Por tanto, puede resultar muy difícil utilizar esta información para aplicar métodos propuestos en la

literatura basados en sistemas lineales. Por otro lado, existen opciones basadas en redes neuronales, según [122] la topología de un MLP es la técnica más utilizada hasta ahora para realizar la tarea de pronóstico de series de tiempo. A través de varios experimentos, ha sido posible hacer una predicción larga de 150 eventos a partir del instante actual “k", teniendo que:

$$x(k+150) = NN[x(k), ..., x(k-10)]. \tag{4.4}$$

Primero, se proponen varias capas para evaluar el rendimiento de una sola capa oculta $W \in \Re^{15}$, un MLP clásico con dos capas ocultas $W \in \Re^{35}$ y $P \in \Re^{15}$ y dos estructuras de aprendizaje profundo con tres ($W \in \Re^{80}$, $W \in \Re^{50}$ y $W \in \Re^{35}$) y cuatro capas ocultas ($W \in \Re^{70}$, $P \in \Re^{60}$, $Q \in \Re^{35}$ y $V \in \Re^{15}$). Las topologías anteriores utilizaban condiciones iniciales aleatorias en un intervalo de [-1,1], teniendo como función de activación un sigmoidea y una tasa de aprendizaje constante de $\eta = 0.25$.

Para la etapa de entrenamiento se utilizaron los datos disponibles en el mes de abril con una dimensión de 1100 elementos del conjunto. Esta base de datos tiene un porcentaje de fallas o pérdida de información del 22,72 % y el algoritmo de entrenamiento es el clásico BP, mientras que para la etapa de generalización se utilizaron los datos disponibles en el mes de septiembre con una dimensión de 2000 elementos. La evaluación de los resultados está determinada por el índice MSE

Para mejorar el pronóstico de la serie de tiempo, se propone lo siguiente: De la Figura (4.12), el número 1 corresponde a la estación ACO entonces se toma este conjunto como Λ_p. El método TL necesita formar el conjunto Λ_σ basado en otras estaciones con un porcentaje de falla de bajo nivel del grupo de estaciones azules en la Figura (4.13). Se seleccionan las estaciones ATI (4) y PED (33), con la intención de mostrar cómo la distancia también impacta los resultados.

Las condiciones ambientales en las proximidades deben ser similares. La distancia entre ACO y ATI ronda los $36,43$ km y la distancia entre ACO y PED ronda los $46,10$ km. El dominio auxiliar ψ_a está formado por dos tareas y está relacionado con la tarea principal. Aplicando el modelo (3.7) y de acuerdo con el algoritmo (2), se estructura el conjunto W^*_σ.

Finalmente, el algoritmo (1) se utiliza para completar el método de TL. Los resultados obtenidos se pueden observar en las figuras (4.14) y (4.15).

A partir de los resultados, es fácil determinar mediante inspección visual una clara diferencia del rendimiento mostrado en cada caso para las tareas auxiliares. Las condiciones ambientales son caóticas, por lo que tiene sentido mencionar que el TL aplicado para el pronóstico de series de tiempo para contaminantes ambientales dependerá de la selección de una estación cercana con un porcentaje de falla bajo.

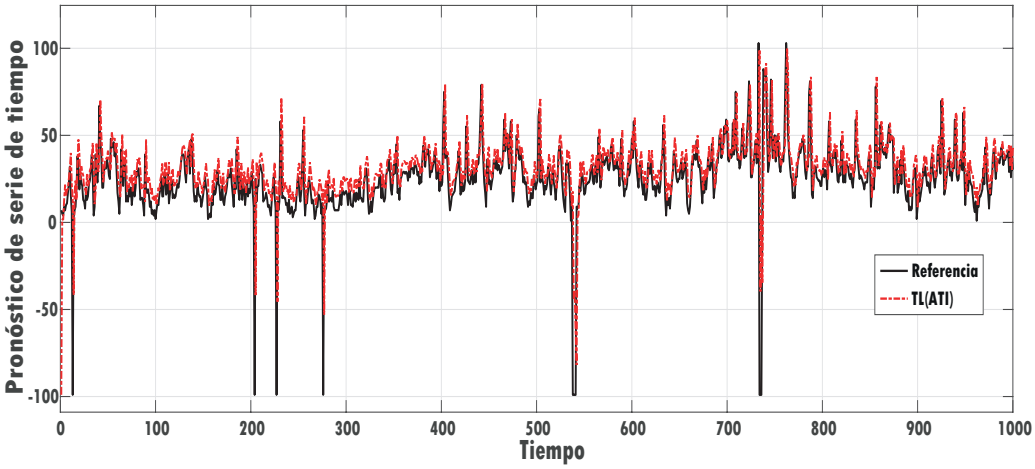


Figura 4.14: Pronóstico de series de tiempo basado en el aprendizaje por transferencia (ATI)

Por lo tanto, se muestra propuesta es generar un pronóstico de serie de tiempo aceptable para PM_{10} cuando la información disponible no sea suficiente para evitar un mínimo local. Sin embargo, un índice común para evaluar la precisión en la predicción de series de tiempo de contaminación del aire es el coeficiente de correlación R, se propone utilizar la función de error para evaluar los resultados obtenidos en la etapa de prueba.

$$R = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{4.5}$$

donde el vector $x(k)$ son los valores reales en la etapa de prueba y $y(k)$ es el pronóstico del

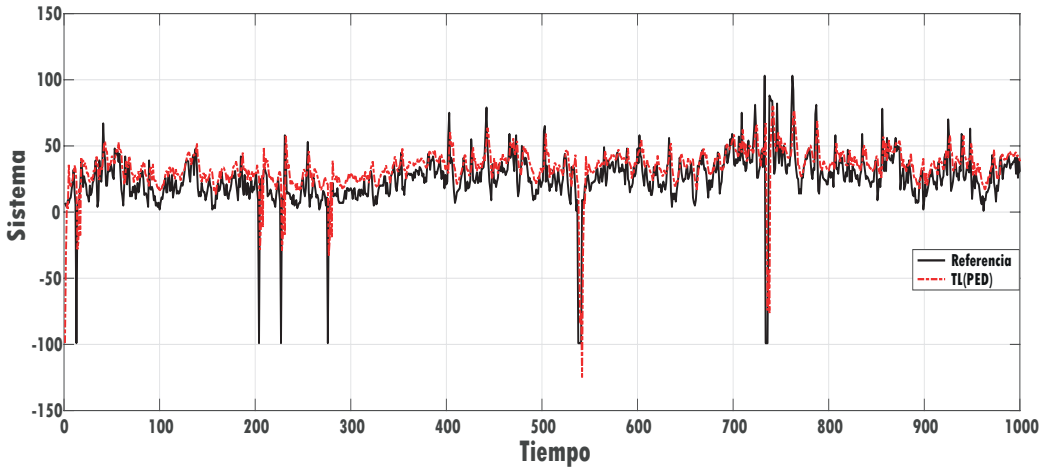


Figura 4.15: Pronóstico de series de tiempo basado en el aprendizaje por transferencia (PED)

modelo neuronal. Para mostrar la información sobre los experimentos, ver Tabla (4.2).

El método TL en la estación ATI muestra mejores resultados ya que las condiciones ambientales son una estación cercana a ACO. Los datos recopilados son cada hora.

Tabla 4.3: Características de los pronósticos experimentales de series de tiempo para PM_{10}

Método	Estación	Función de Activación	η	α	Épocas	Error x10^{-3}	Parámetro R
BP	ACO	Sigmoide	0.01	0.05	100	9.4	0.4712
BPM	ACO	Sigmoide	0.6	0.05	1	3.4	0.4815
TL	ACO-ATI	Sigmoide	0.1	0.05	1	2.1	0.6279
TL	ACO-PED	Sigmoide	0.8	0.05	1	3.2	0.4818

Los experimentos ocupan una MLP con una capa oculta y cinco nodos. Para BP y BPM, la etapa de entrenamiento usa 70 elementos en el conjunto de datos (ACO). Por otro lado, los TL-ATI y TL-PED utilizan, en la etapa de subtarea, 500 elementos en la base de datos y finalmente en la etapa de TL se utilizan los 70 elementos de la estación ACO.

En la Figura (4.16) a) se nota la correlación entre los valores reales y el pronóstico de BP, b) es la correlación entre los valores reales y el pronóstico de BPM, c) es la correlación

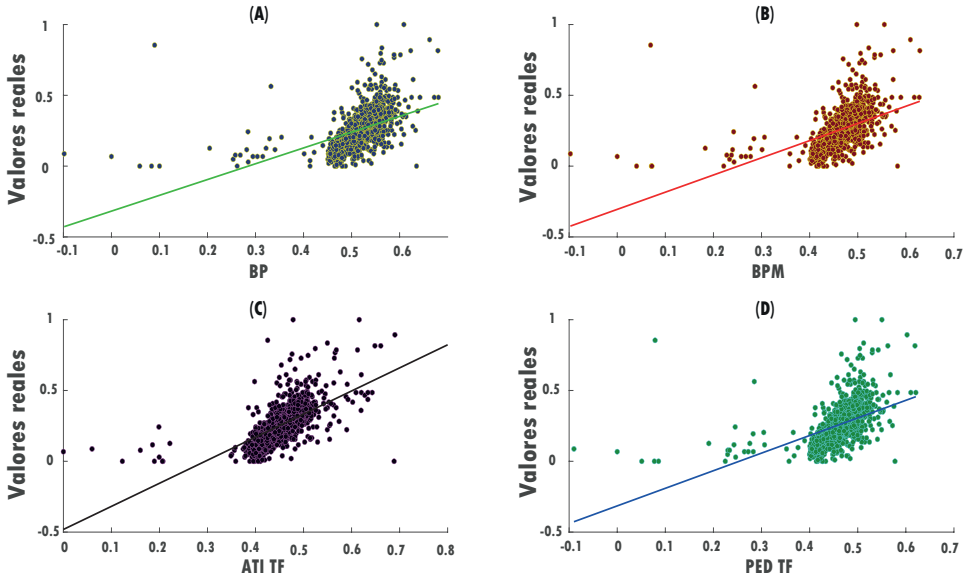


Figura 4.16: Distribución del parámetro R.

entre los valores reales y el pronóstico de TL-ATI y d) es los valores reales y la previsión TL-PED.

Para la etapa de entrenamiento se utilizaron los datos disponibles en el mes de abril con una dimensión de 1100 elementos del conjunto. Esta base de datos tiene un porcentaje de fallas o pérdida de información del 22,72 % y el algoritmo de entrenamiento es el clásico BP, mientras que para la etapa de generalización se utilizaron los datos disponibles en el mes de septiembre con una dimensión de 2000 elementos. La evaluación de los resultados está determinada por el índice MSE.

Es evidente que el desempeño no es el esperado ya que el error de predicción aumenta con el paso del tiempo, lo cual es natural, sin embargo, con la tendencia que marca la Figura (4.17), es fácil notar que los modelos RNA no cumplen satisfactoriamente con el tiempo. tarea de previsión de series.

En el campo del aprendizaje automático, los esfuerzos realizados por los investigadores apuntan a desarrollar nuevos métodos que permitan obtener mejores resultados y con una

Tabla 4.4: Error medio cuadrático experimental en el pronóstico de series de tiempo para el componente contaminante PM_{10} en la estación de monitoreo ACO con datos de 2000 en etapa de generalización.

Modelo Neuronal	**Error**$_T$ $\times 10^{-3}$
Una sola capa	490.2
MLP_2	367.5
DL_3	310.8
DL_4	257.1
TL	79.3
ML	74.5
MTA	15.6

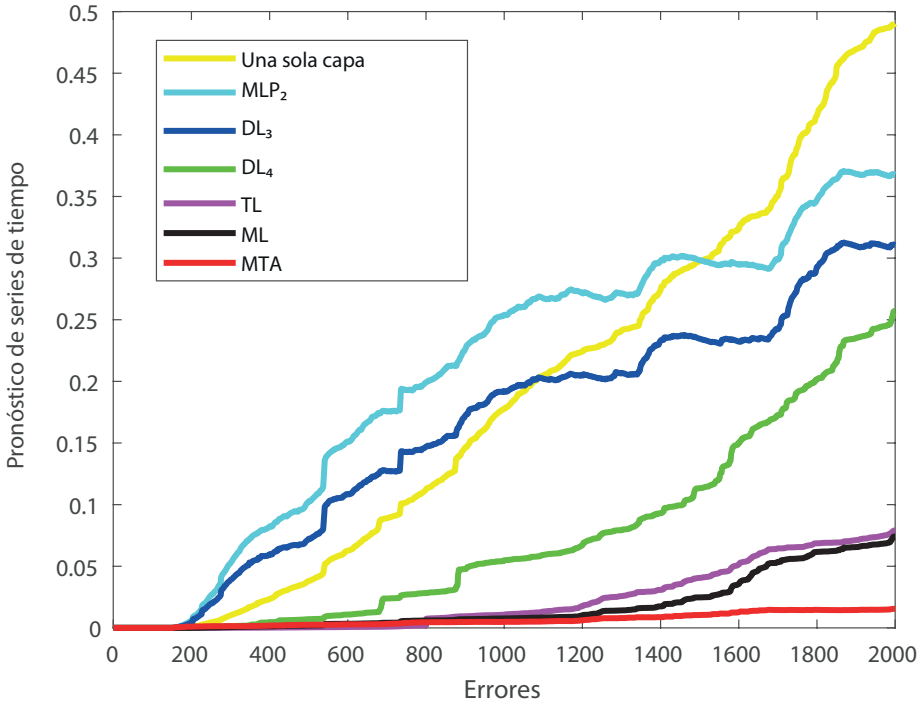


Figura 4.17: Error de validación de 2000 eventos.

previsión de futuro más amplia. El ML es una técnica relativamente nueva, que permite que un modelo basado en RNA sea capaz de mejorar sus resultados propuestos a partir de la extracción de conocimiento y experiencia a través de tareas paralelas basadas en un mismo problema, el uso de esta técnica se puede revisar en [140]. El TL es un método que permite utilizar información o conocimiento de la resolución de un problema diferente, la forma clásica en la que se ha presentado es a través de un proceso de pre-entrenamiento en el que el RNA principal no comienza con condiciones iniciales aleatorias, sino la RNA resuelve un problema auxiliar con un dominio relativo a la tarea principal y los conocimientos adquiridos a través de las matrices de ponderaciones. En este trabajo se aplica el ML con diferentes condiciones iniciales debido a la falta de información derivada de las condiciones reales del proceso, se ejecutó con 20 tareas auxiliares y con una condición de ángulo de 0.90, esta técnica se aplica bajo el algoritmo (1). Está claro que la información disponible es insuficiente para la aplicación del Meta-Aprendizaje. En cambio, para el TL se toma una estación de monitoreo ATI cercana, con una distancia de 36.43 km de la estación de monitoreo ACO, que presenta un bajo porcentaje de fallas o pérdida de información, la topología tomada es DL_3 porque presenta un resultado similar al DL_4 y estos modelos son mejores que aquellos con una sola capa oculta y MLP_2.

Sin embargo, no es posible mejorar la respuesta de la herramienta para el pronóstico de series de tiempo, ya que presenta un comportamiento similar a los modelos DL_3 y DL_4. Por tanto, se puede decir que el uso de información auxiliar puede no necesariamente mejorar el proceso de predicción de series de tiempo, por lo que es fundamental generar un método que aproveche este conocimiento extraído de otra forma.

El MTA es una combinación de enfoques de ML y TL, la aplicación del método se describe mediante los algoritmos (1) y (2). Dado que la estación de monitoreo (ATI) seleccionada para realizar la tarea auxiliar se ubica en una región “cercana", a la estación de monitoreo, considerando ahora una sola capa oculta para el método propuesto por nosotros, la predicción de series de tiempo ha obtenido un mejor desempeño en minimizar el índice de rendimiento según el obtenido en la Tabla 4.4. Además, la Figura (4.17) muestra cómo el tamaño de los datos para la etapa de generalización tiene un impacto directo en los

resultados, ya que si el conjunto no tiene suficiente cantidad de datos para demostrar la correcta ejecución de los métodos, puede ser decisivo para la interpretación de datos. En la Figura (4.18), una sección de 200 datos (1650-1800), para inspeccionar visualmente el desempeño de los modelos DL_4, TL, ML y Meta-Transferencia de Aprendizaje.

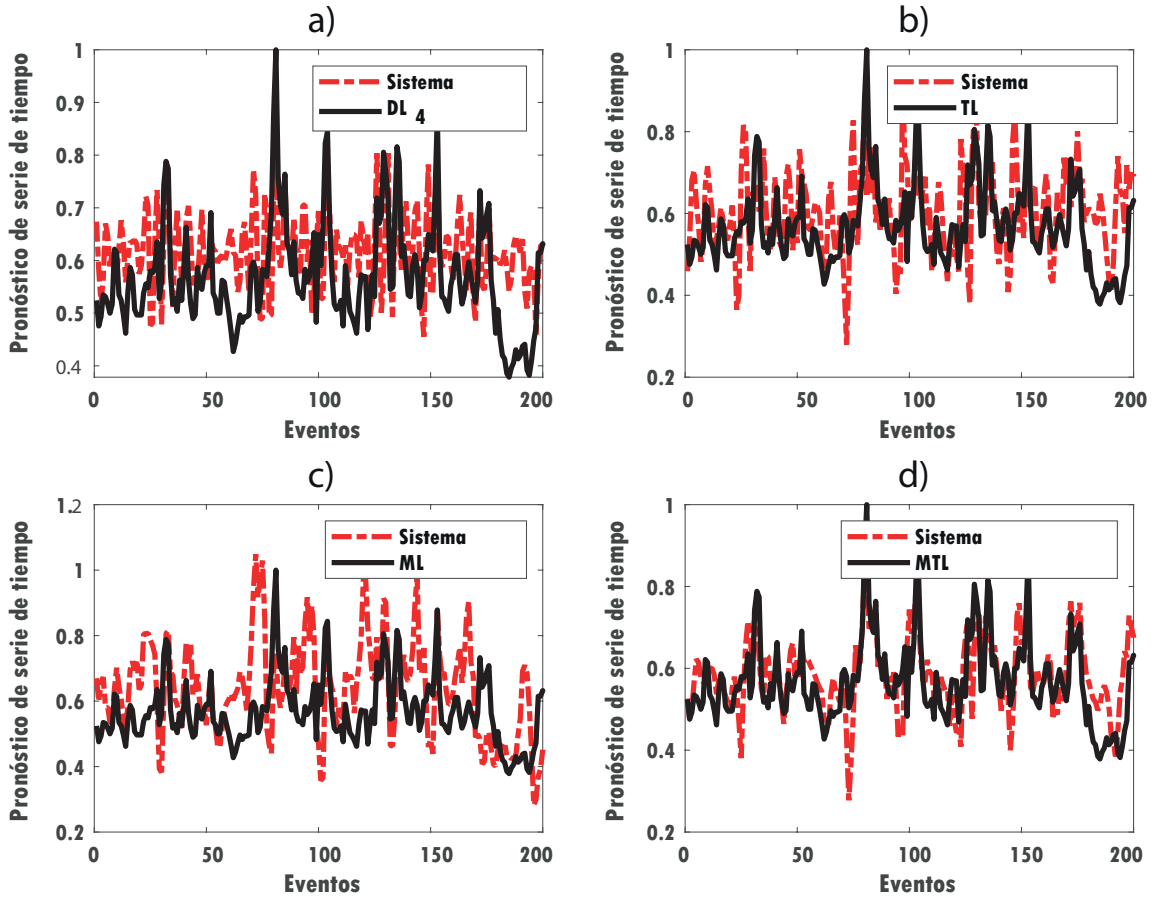


Figura 4.18: a) Aprendizaje Profundo, b) Transferencia-Aprendizaje, c) Meta-Aprendizaje, d) Meta-Transferencia de Aprendizaje

Con los resultados obtenidos, se puede ver que el Meta-Transferencia de aprendizaje tiene una predicción más cercana al valor real de la prueba que los otros métodos propuestos.

Aunque esto es sólo una conjetura basada en una sección de los datos de prueba de 2000, la Tabla (4.4) muestra que el mejor desempeño en el pronóstico de la serie temporal es el método propuesto en este trabajo, debido a un mayor índice de desempeño de minimización del MSE.

4.4. Meta-Transferencia de Aprendizaje para la predicción de magnitud de terremotos a largo plazo

Dado que la fricción es un fenómeno lineal [141], los terremotos pueden ser considerado un sistema determinista caótico [142] con limitaciones en previsibilidad. La interpretación de los terremotos puede explicarse como un proceso estocástico o como un proceso caótico determinista [143]. En general, existen dos enfoques para la predicción de terremotos:

- Los terremotos se consideran un proceso estocástico, donde el choque principal Los intervalos entre eventos son estacionarios y típicamente siguen un patrón de distribución Poisson, [144]. Los terremotos pueden tener algún tiempo de renovación, este modelo imita la teoría del rebote elástico [145]. La probabilidad del próximo gran terremoto en un intervalo de tiempo dado se puede estimar con base en la sismicidad pasada.
- Los terremotos se consideran un proceso determinista, ya que esto es un resultado del deslizamiento por fricción de dos placas tectónicas [146]. El método determinista de la predicción de los terremotos sigue siendo un tema de debate en sismología.

Los estudios teóricos y numéricos basados en ecuaciones deterministas indican ese deslizamiento puede representarse a través de una serie temporal caótica, [147]. Su horizonte de previsibilidad sería todavía objeto de debate y en su mayoría no explorado. El comportamiento caótico en terremotos regulares sigue siendo un desafío debido al corto período de tiempo de observación, [148].

Los eventos de pronóstico a largo plazo se basan en terremotos que llegan periódicamente, en general, un evento a largo plazo es demasiado difícil de predecir debido a las limitaciones

de información disponible. Un procedimiento completo de predicción de terremotos debe tener tres tipos de información: magnitud, ubicación y hora de ocurrencia. Muchos de los métodos se utilizan para predecir terremotos, como el enfoque basado en reglas. En [149], que tienen grandes dificultades debido a la rareza de los datos, la calidad de los datos históricos del terremoto, la falta de patrones y la variabilidad del rendimiento en diferentes ubicaciones geológicas. Los desafíos importantes son: la precisión del pronóstico se limita a una gran magnitud, gran error de pronóstico en la predicción a largo plazo, el efecto de las condiciones ambientales, factores e incertidumbre en los factores. Se propone el uso del método Meta-Transferencia de Aprendizaje para pronóstico de magnitud de terremotos en Italia, sobre la base del conocimiento de la sismicidad ocurrida en México.

A pesar de que se desea un pronóstico de eventos futuros en una serie de tiempo con información sísmica de Italia, la cantidad de datos disponibles es insuficiente para entrenar una red neuronal. Por lo tanto, se propone agregar eventos en series de tiempo a partir de datos de magnitud provenientes de información sísmica registrada en México. Sin embargo, la inserción de datos en la serie temporal de Italia no puede ser deliberada. Por esta razón, después de realizar una descomposición de *Wavelets* de series de tiempo tanto italianas como mexicanas, se calcula la desviación estándar de los coeficientes de *Wavelets* a partir de ambos conjuntos de información. Después de eso, se examinan las desviaciones estándar resultantes y se selecciona un conjunto de información sísmica de México. Luego, el tiempo entre eventos de ambos conjuntos de datos se calcula de modo que se encuentre una condición para agregar la información sísmica de México en la serie de tiempo de Italia. Finalmente, una función creada nos permite tomar información del conjunto de datos correspondientes a México e incluirlos en la información sísmica de Italia.

Al aplicar el algoritmo (2), se obtuvo un octavo nivel de descomposición. En la Figura (4.19) se muestra la descomposición de los datos sísmicos de Italia. En otras palabras, la desviación estándar, en este caso, nos permite saber qué conjunto de datos sísmicos de México es similar a la información sísmica de Italia. Si se agrega información similar en términos de desviación estándar al conjunto de datos de Italia, se alimentará con eventos similares, pero para entrenar correctamente una red neuronal, es necesaria la diversidad en la señal. Por lo

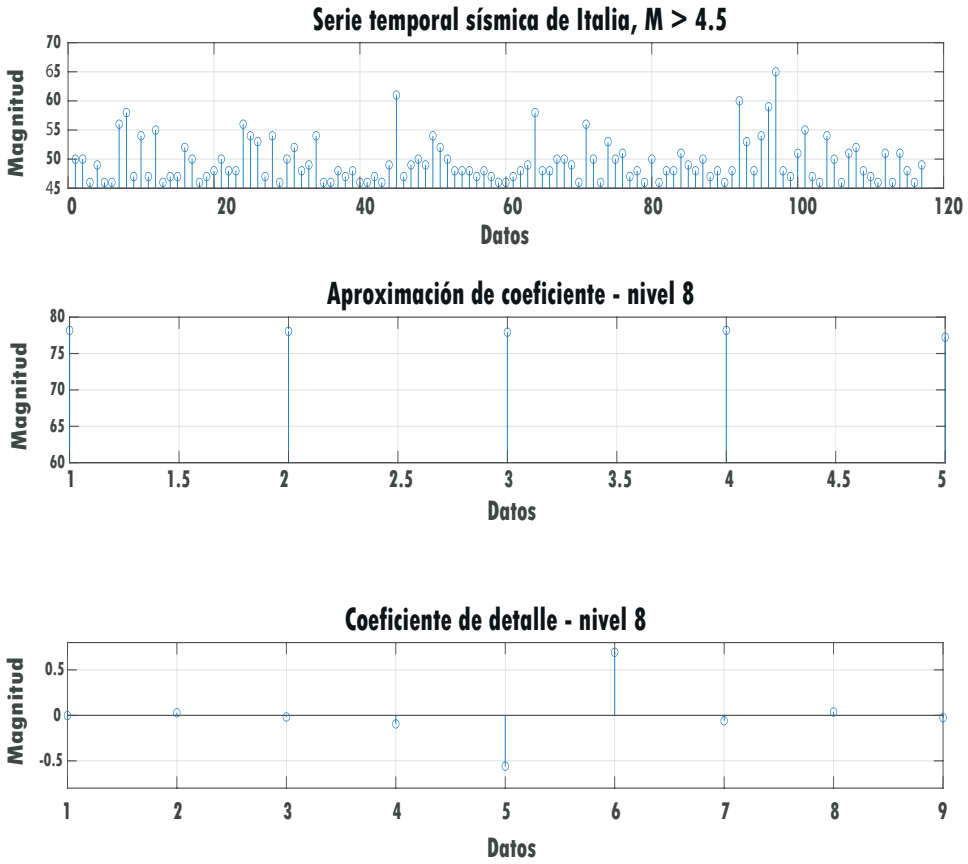


Figura 4.19: Descomposición de los datos sísmicos de Italia

tanto, la serie de tiempo de México que se selecciona es la que tiene una desviación estándar mayor de la información sísmica de Italia. En la Figura (4.20), se muestran los dos conjuntos de datos con la desviación estándar más alta de la información de Italia. Se selecciona la información de México en 2016.

El tiempo entre eventos nos permite encontrar gráficamente una condición para agregar la información sísmica de México en 2016 en la información de Italia. El tiempo entre eventos tiene la información de los intervalos de tiempo entre eventos sísmicos sucesivos. Cuando el tiempo entre eventos de Italia es menor que el umbral de tiempo mínimo entre eventos llamado γ, la información de magnitud de México se guarda en una matriz, que estará llena

	Desv(cA)	Desv(cD)
Italia:	0.338842	0.045574
México 2001:	1.152003	0.144298
México 2002:	1.847408	0.307796
México 2003:	2.248482	0.282674
México 2004:	3.640420	0.488308
México 2005:	1.315854	0.184348
México 2006:	3.050858	0.433966
México 2007:	1.057269	0.373970
México 2008:	1.035213	0.153787
México 2009:	0.637653	0.095475
México 2010:	1.368199	0.201624
México 2011:	1.205139	0.135587
México 2012:	0.622432	0.122014
México 2013:	2.083414	0.297961
México 2014:	0.624838	0.055370
México 2015:	0.201995	0.034680
México 2016:	0.296783	0.072113
México 2017:	0.733177	0.086301
México 2018:	1.133294	0.227045
México 2019:	0.364960	0.118925
México 2020:	2.990503	0.430463

Figura 4.20: Desviación estándar de todos los conjuntos de datos sísmicos

no sólo de la magnitud de los terremotos registrados sino también de ceros. Luego, se detecta la posición donde se guarda la información de magnitud en el último arreglo para que este parámetro permita saber dónde agregar la información sísmica de México en la información de Italia. Después de eso, un parámetro β indica la cantidad de datos sísmicos de México agregados al conjunto de datos de Italia. Finalmente, la nueva señal para entrenar la red neuronal está lista y se construye según el modelo (3.30):

$$f(\tau_{i_{Italia}}, \tau_{i_{Mxico}}) = \begin{cases} \tau_{i_{Italia}} \quad if \geqslant \gamma; \\ \tau_{i_{Italia}} \oplus \tau_{i+\beta_{Mxico}} \oplus \tau_{i+\beta+1_{Italia}} if < \gamma. \end{cases} \tag{4.6}$$

Se crea una nueva serie de tiempo para entrenar la red neuronal a partir de la información sísmica de ambos países, Italia y México. Esta serie temporal con más información permitirá

que el método de Meta-Transferencia de Aprendizaje mejore la etapa de formación y en consecuencia, obtenga mejores resultados en la previsión de eventos futuros en Italia. En la Figura (4.21), se puede apreciar la nueva serie de tiempo.

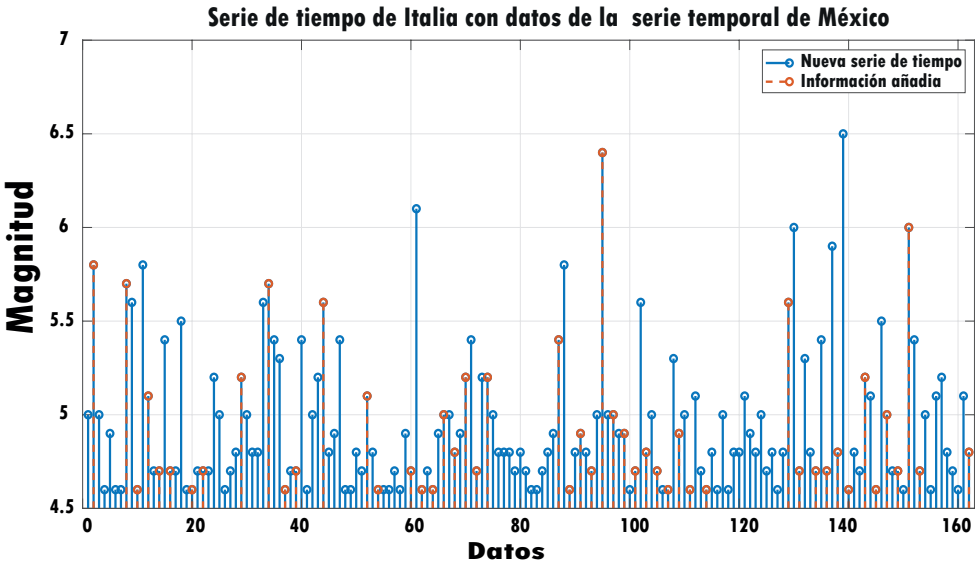


Figura 4.21: Serie temporal de Italia con datos de la serie temporal de México

La Tabla (4.5) muestra las comparaciones del algoritmo desarrollado con algunos métodos conocidos. Sólo se toman 10 eventos para la etapa de prueba. La experimentación tiene en cuenta la información disponible para una ventana de magnitud $M > 4.5$.

Como se muestra en la Figura (4.22), el rendimiento de los métodos clásicos no es satisfactorio porque la cantidad de información contenida en la base de datos del historial para $M > 4.5$ no es suficiente para superar los problemas conocidos de la red neural propuesta. En general, estos problemas ocurren en las topologías RNA conocidas. La MTA minimiza el error del índice MSE, en este caso se necesita el sentido del comportamiento del pronóstico para entender que la mínima desviación de la magnitud implica una liberación de mucha más energía por parte del terremoto. La técnica propuesta, a través del método

Tabla 4.5: MSE para experimentación con datos de Italia y México

Modelo Neuronal	**Error**$_T$ $\times 10^{-3}$
MLP_2	3.5833e-01
DL_3	3.5234e-01
DL_4	3.2369e-01
DL_4 con TL	9.8731e-02
DL_4 con ML	9.1645e-02
MLP_2 con MTA	3.1645e-02

de búsqueda basado en la Transformada *Wavelet* de resolución múltiple para encontrar el mejor W^* posible, plantea una paridad entre el conjunto de datos sobre el que se pretende realizar la predicción de la serie temporal y el conjunto de datos con del cual se extraerán las características que se transformarán en conocimiento y experiencia. Finalmente, cuando la transferencia de conocimiento a través de ML permite aprovechar los conocimientos adquiridos logrando un mejor desempeño que las otras técnicas ya que los pesos de los RNA W convergen a $W^\star$ mediante la proyección del MTA, tal que el desempeño se observa en la minimización de la función de costo MSE.

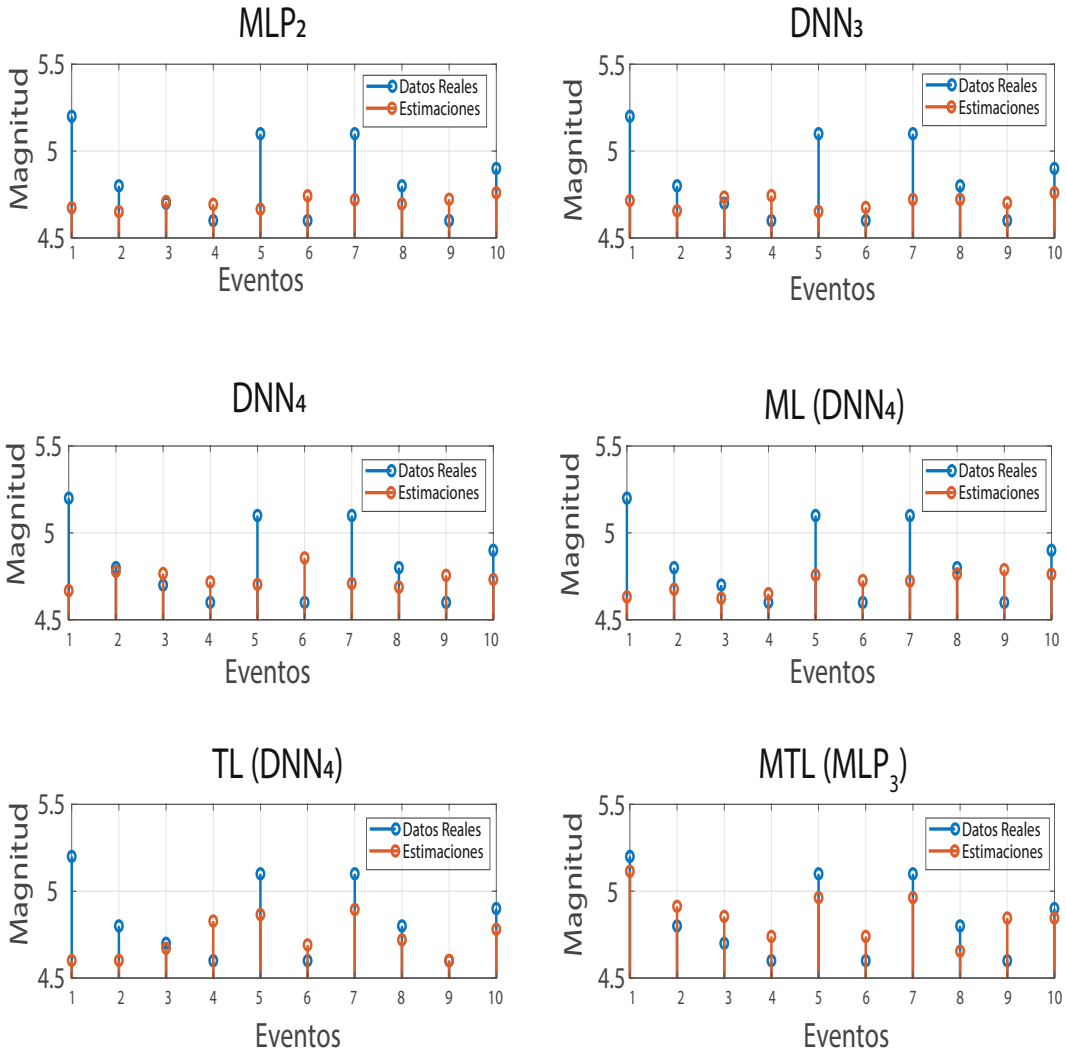


Figura 4.22: Comparación de diferentes métodos para el pronóstico de series de tiempo basado en RNA

4.5. Meta-Transferencia de Aprendizaje para rechazo de perturbaciones en controladores tipo PID

Para ilustrar cómo es posible utilizar los métodos desarrollados en este trabajo dentro del área de Control Automático, se tomara cómo caso de estudio un controlador clásico proporcional-integral-derivado (PID). Industrialmente representa el método más utilizado para el control de lazos de proceso, el cual representa alrededor de un 90 % [150], esto debido a su sencillez para su implementación y su robustez ante perturbaciones. Se han propuesto algunas reglas empíricas para el ajuste de las ganancias para mejorar el rendimiento de control de los controladores PID [151], e incluyen las reglas de ajuste de Ziegler-Nichols [152], el método de síntesis directa [153] y el control de modelo interno método [154]. Los últimos desarrollos se informan en el 3er. Conferencia de la IFAC sobre avances en el control PID.

En [155], se menciona que los esfuerzos de investigación sobre el control de PID se centran en los contextos de la teoría y la ingeniería de control, pero el principio de ajuste del rendimiento en el control PID sigue sin estar claro. Por ejemplo, el enfoque basado en errores, el control PID aún necesita información rica de la planta para un control de alto nivel. De hecho, sólo una pequeña cantidad de información de la planta es fundamental para el diseño del controlador, como el orden relativo y la ganancia de alta frecuencia. Como utilizar información tan limitada de la planta para el desempeño del control PID aún se desconoce la mejora.

Para muchas aplicaciones de control de procesos, rechazo de perturbaciones el rendimiento es más importante que el seguimiento puramente del punto de ajuste. Por lo tanto, se han propuesto algunos controladores PID modificados, denominados PID de rechazo de perturbaciones, para mejorar el rendimiento del controlador, junto con el supuesto de que el modelo de la planta se conoce exactamente [156].

En [157] aplicaron un PID neuronal con el algoritmo de entrenamiento BP para evaluar la capacidad de controlar el flujo en una tubería de una planta con modelo conocido. Otras aplicaciones de este tipo de controlador se han desarrollado en el control de nivel de dos

tanques [158] y en procesos altamente no lineales como lo puede ser la neutralización de pH en procesos industriales [159, 160]. Además, en [161] un PID neuronal de fue propuesto en función de un esquema de identificación y sintonizador en línea, basado en funciones de activación de base radial y *wavelets* para un motor de inducción trifásico sin carga; mientras que en [162] se presenta un esquema similar al anterior donde se busca el control de velocidad para un generador/turbina de aire, mostrando resultados únicamente por simulación, cabe mencionar que en estos trabajos no se consideran el desempeño de los controladores bajo perturbaciones externas. De acuerdo con [88], el no conocer el modelo matemático de un sistema afecta el desempeño de un sintonizador basado en redes neuronales, por lo que la mayoría de los esquemas dependen de emuladores que trabajan en paralelo para mejorar el proceso de sintonización.

El controlador PID clásico es ampliamente utilizado en el control de procesos debido a que tiene una arquitectura de control a lazo cerrado que compara la salida del sistema con el valor deseado. El desempeño del controlador depende de tres parámetros conocidos como ganancias Proporcional, Integral y Derivativa, que pueden obtenerse mediante técnicas convencionales que se basan en suposiciones sobre el sistema, al igual que el método de Ziegler-Nichols que está basado en la respuesta al escalón y produce un desempeño razonable para lazos simples y sólo se aplica en procesos estables, [163].

Sea el sistema linea de primero orden e invariante en el tiempo sin retador, descrito por:

$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1}. \tag{4.7}$$

Se propone un sistema de re-circulación de flujo de agua para parametrizar la ecuación anterior utilizando una bomba centrifuga de corriente alterna de acuerdo con el siguiente Diagrama de Tubería e Instrumentación.

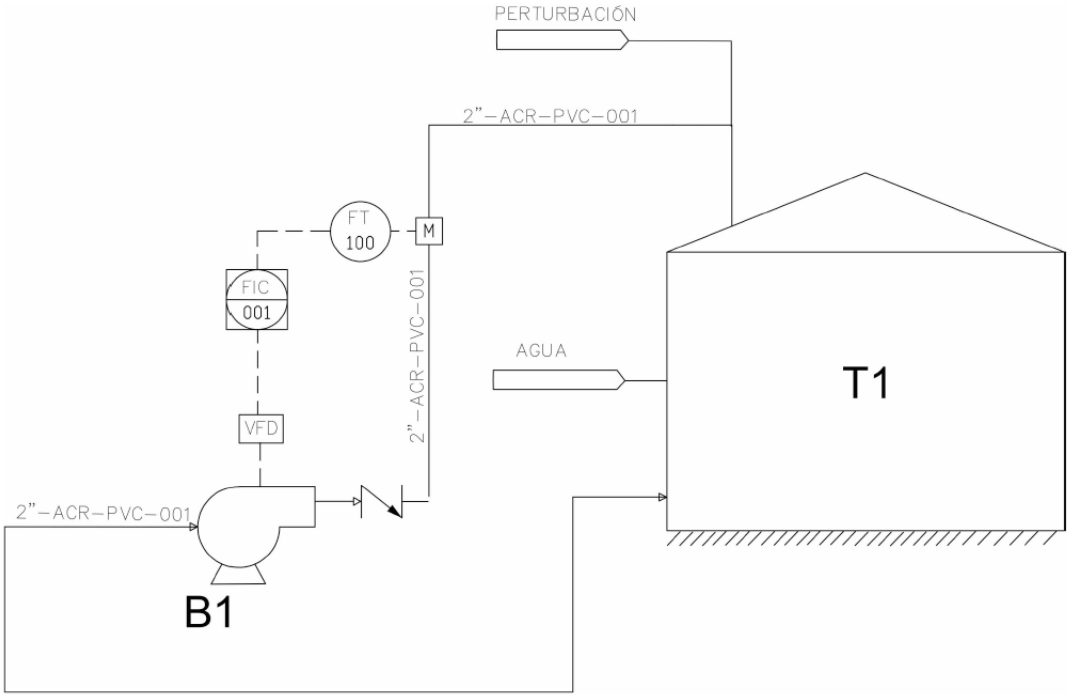


Figura 4.23: Diagrama de Tubería e Instrumentación para un sistema de flujo de agua.

De los datos del sistema se sabe que la gravedad es de 9.81 m^2/s y la densidad del agua a una temperatura de 25 $°C$ es 997 kg/m^3. Para el caso se tiene una bomba Pedrollo CP 160C, de 1.5 HP, con las siguientes características:

$$H = [32, 31, 30.5, 29.5, 28, 26, 23, 20][m];$$
$$Q = [0, 50, 75, 100, 125, 150, 175, 200][LPM].$$

donde H corresponde a la altura manométrica total y Q es el caudal o flujo. Con MATLAB/Simulink en su librería de Simscape y con los datos disponibles del fabricante es posible obtener la curva de respuesta del sistema en función del flujo volumétrico.

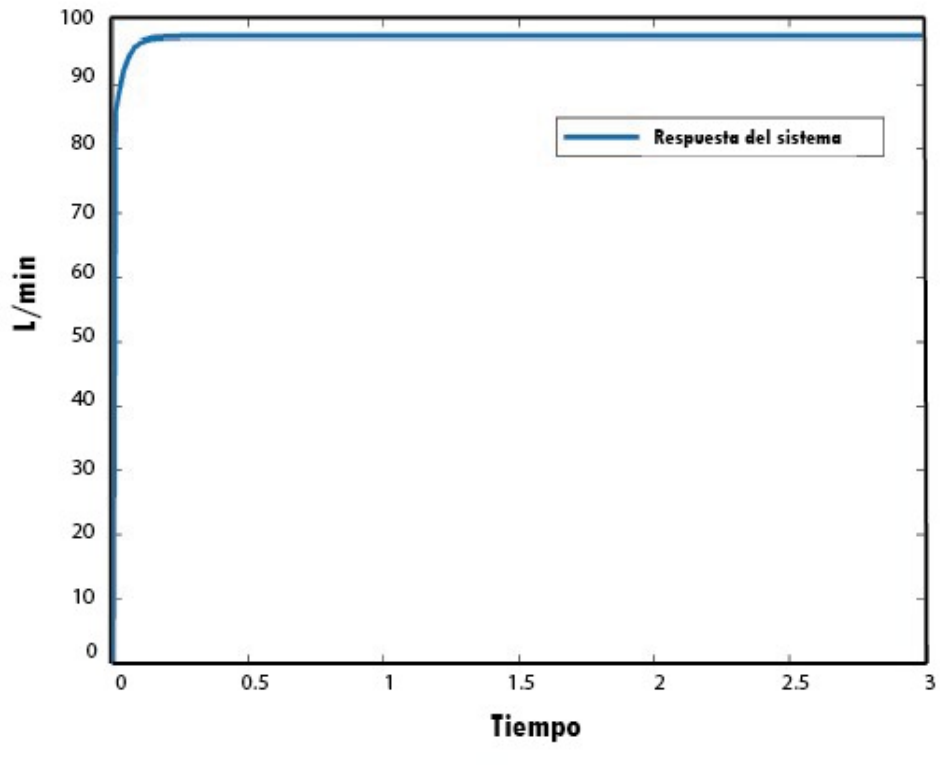


Figura 4.24: Esquema de la red neuronal.

Utilizando la técnica de mínimos cuadrados para la identificación paramétrica [164], se obtiene el siguiente modelo:

$$\frac{Y(s)}{U(s)} = \frac{3.7341}{s + 130.8171} \frac{LPM}{RPM}. \tag{4.8}$$

En la siguiente Figura (4.25), se puede observar el resultado de la comparación del modelo identificado y el sistema.

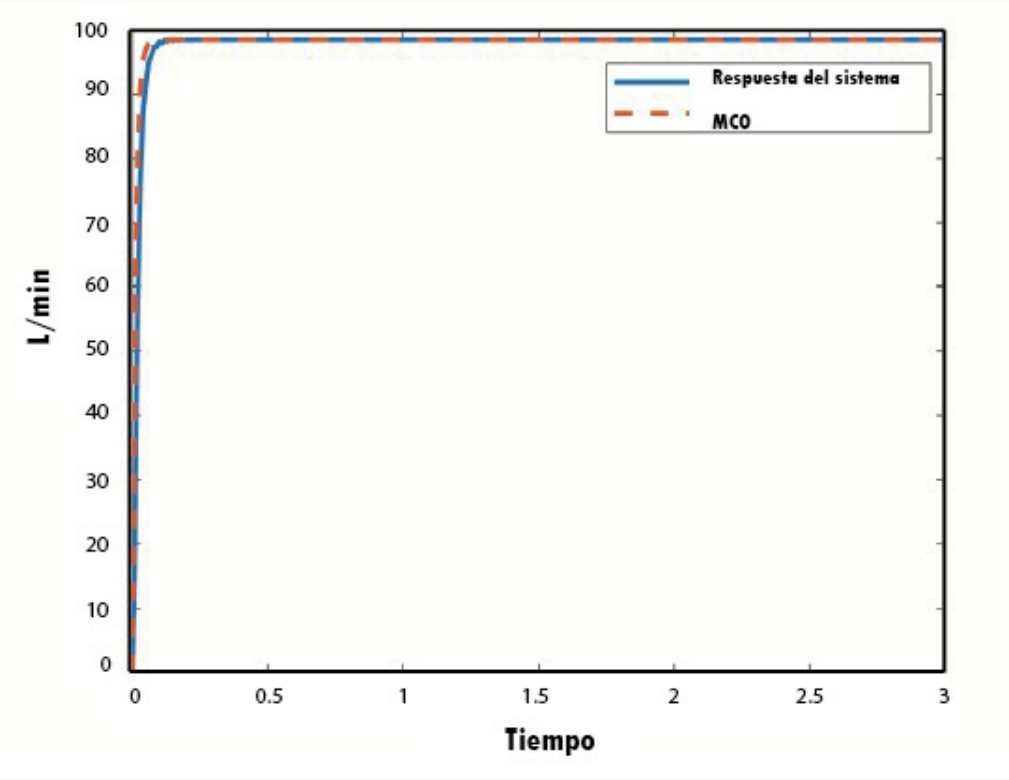


Figura 4.25: Esquema de la red neuronal.

La evaluación de la capacidad de los controladores, Neural-PID y Meta-Neural-PID, para regular el la variable de proceso con una perturbación constante, se efectuó con la siguiente función de perturbación:

$$f_1(t) = \begin{cases} 0 & si \quad 0\ seg < t \leq t_1\ seg \\ g & si \quad t_1\ seg < t \leq t_2\ seg, \end{cases} \tag{4.9}$$

donde t_1 representa el tiempo antes de la perturbación, el intervalo (t_1,t_2) donde la perturbación es constante y g representa la amplitud de la perturbación. La ilustración del método queda definido de la siguiente manera,

- Se tiene un lazo de control con un controlador Neural-PID-1 para el sistema descrito

por (4.8), bajo ciertas condiciones iniciales aleatorias (W_{1i} y V_{1i}) en un intervalo $[-1, 1]$, con 10 nodos en la capa oculta y con las constantes de aprendizaje ($\eta = 0.5416$, $\alpha = 0.1$ y $\beta = 0.04$), con una duración del experimento de 30 seg y los tiempos de acuerdo con (4.9) de $t_1 = 5seg$ y $t_2 = 30seg$ y una perturbación $g = 1$.

- Se tiene un lazo de control con un controlador Neural-PID-2 para el sistema descrito por (4.8), bajo ciertas condiciones iniciales aleatorias (W_{2i} y V_{2i}) en un intervalo $[-1, 1]$, con 10 nodos en la capa oculta y con las constantes de aprendizaje ($\eta = 0.5416$, $\alpha = 0.1$ y $\beta = 0.04$), con una duración del experimento de 30 seg y los tiempos de acuerdo con (4.9) de $t_1 = 5$ seg y $t_2 = 30$ seg; $g = 1.1$.

Teniendo los siguientes resultados, y considerando la normalización de las señales, ver Figura (4.26). Donde a través de una inspección gráfica se puede determinar que el desempeño para el rechazo de la perturbación con el controlador Neural-PID-2 es malo en comparación con el Neural-PID-1, esto se debe principalmente a las condiciones iniciales (W_{2i} y V_{2i}) y (W_{1i} y V_{1i}).

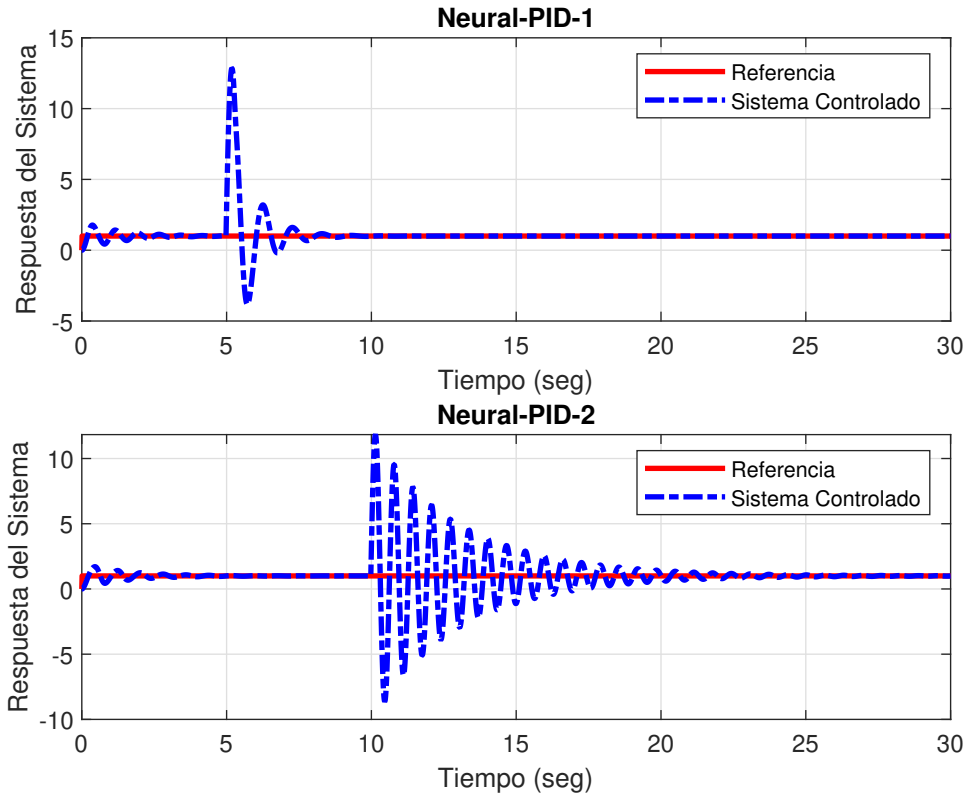


Figura 4.26: Comparación de la respuesta de los controladores Neural-PID 1 y 2.

Se busca mejorar el desempeño del rechazo de la perturbación constante ya que para el controlador Neural-PID-2 le toma un tiempo aproximado de 12.5 segundos. Para ello se implementa un controlador Meta-Neural-PID, el cual bajo las mismas condiciones iniciales y de operación usara experiencia ganada por el controlador Neural-PID-1 para mejorar su desempeño, el siguiente esquema explicara la metodología de lo mencionado. En el cuál se muestra que la sintonización del Meta-Neural-2 es realizado en linea para “comunicar" con el controlador Neural-PID-1 y aprovechar su experiencia rechazando una perturbación constante, esto es expresado en pesos óptimos ($W^\star$ y $V^\star$) para que a través del Meta-Aprendizaje estos sean usados por el control Neural-PID-2 formando de esta manera el denominado Meta-Neural-PID, ver Figura (4.29).

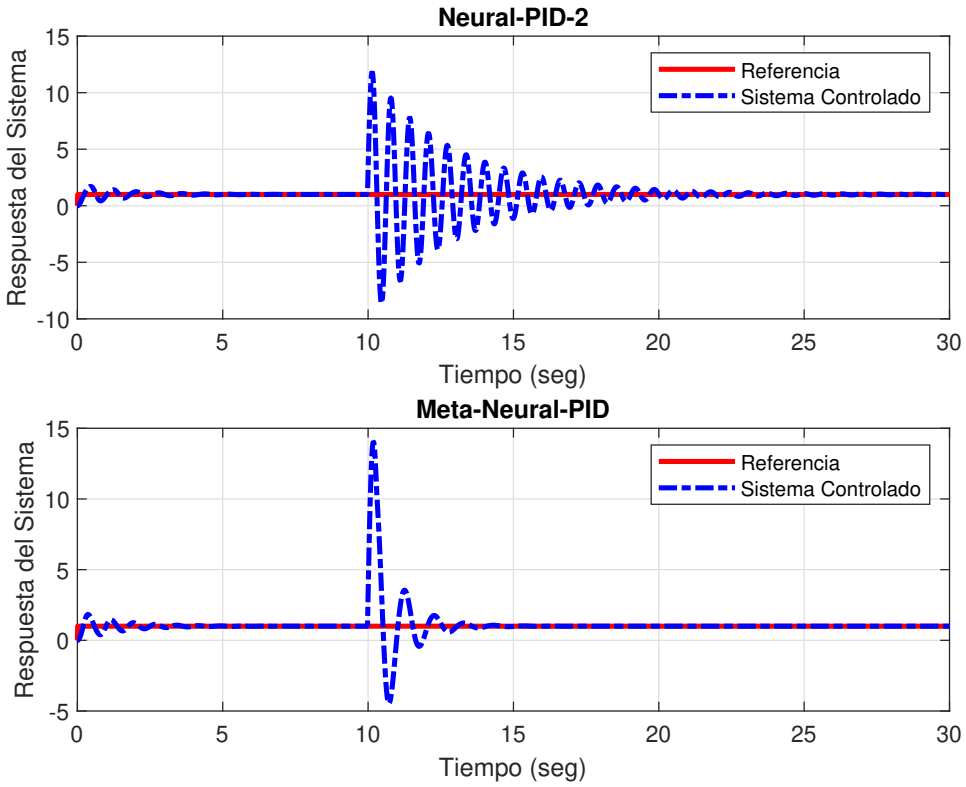


Figura 4.27: Comparación de los controladores Neural-PID-2 y Meta-Neural-PID.

Como se visualiza en la Figura (4.27), el rechazo de la perturbación se lleva acabo en un tiempo aproximado de 4.5 segundos por parte del control Meta-Neural-PID, sin embargo, el sobre impulso máximo es mayor que el observado en Neural-PID 2. Las ganancias que producen los comportamientos respectivamente son mostradas en la Figura (4.28).

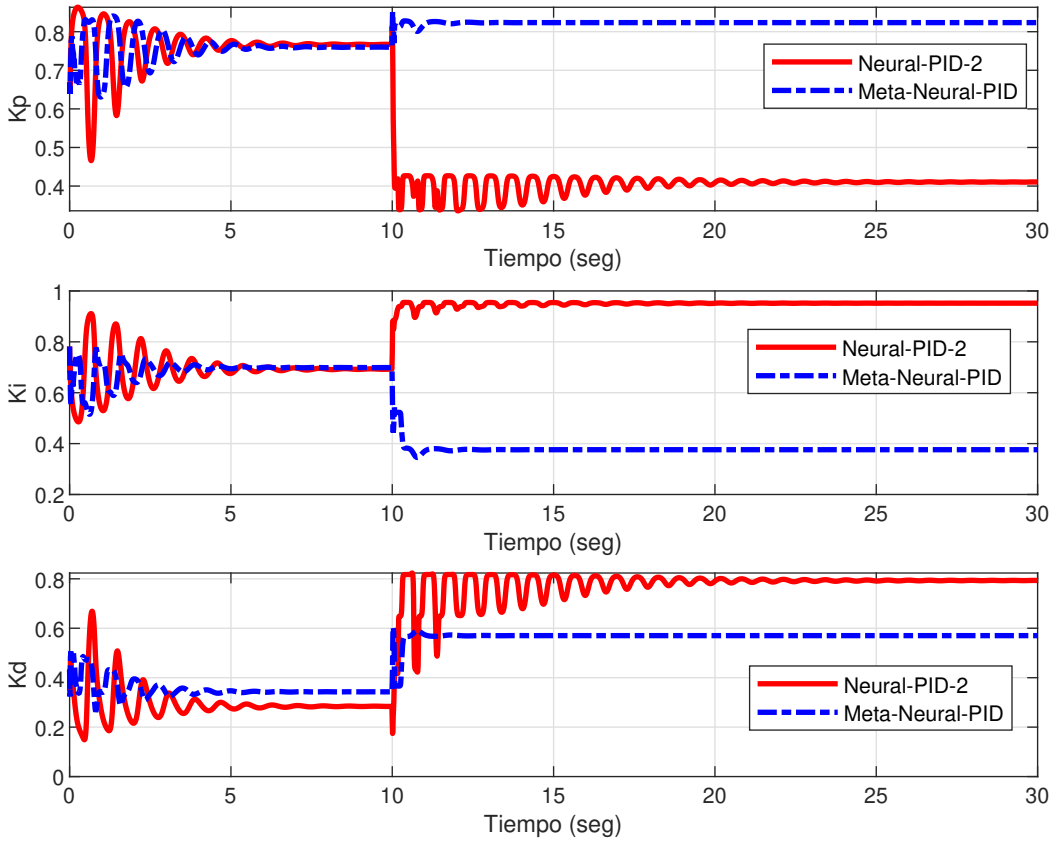


Figura 4.28: Comparación de los las ganancias producidas por los controladores Neural-PID-2 y Meta-Neural-PID.

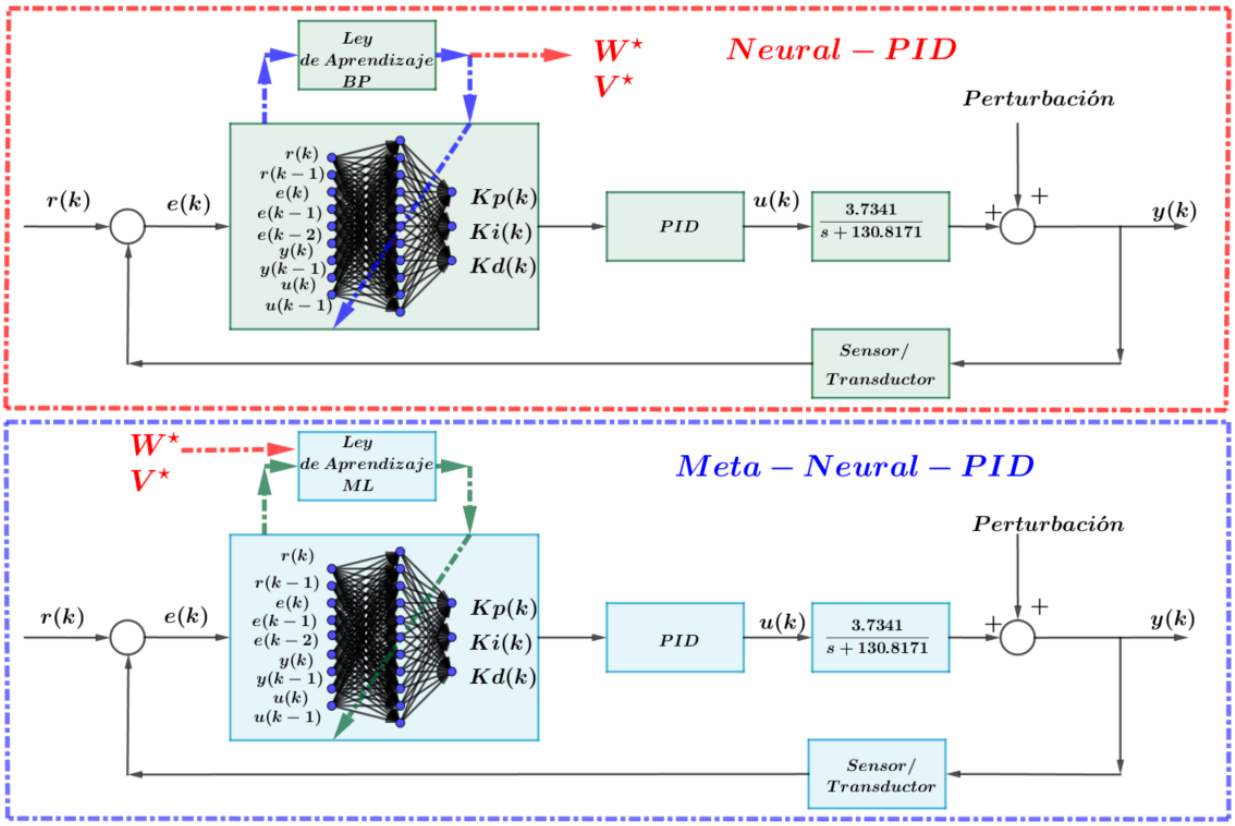


Figura 4.29: Esquema de la implementación del Meta-Aprendizaje en un controlador Neuronal PID.

Capítulo 5

Conclusiones y trabajo a futuro

En este trabajo parte de que aún no esta todo dicho en cuestión del uso de MLP y que aún es posible potenciar sus capacidades y además usarlas no sólo dentro de la teoría de control automático, sino en aplicaciones del mundo real. Se presentan modificaciones a los métodos de Meta-Aprendizaje y Transferencia-Aprendizaje, también se genera un método inédito denominado Meta-Transferencia-Aprendizaje, se analizan las propiedades de estabilidad en el proceso de aprendizaje así cómo su convergencia (fuerte y débil). Se ilustra la versatilidad de los métodos a través de diferentes aplicaciones como: pronóstico de series de tiempo caóticas, identificación de sistemas no lineales y control de sistemas. Los resultados numéricos demuestran que bajo las condiciones propuestas las ideas expuestas permiten una mejora en el desempeño de una red neuronal artificial en su topología MLP con respecto a su forma clásica y puede llegar a competir con topologías avanzadas cómo el aprendizaje profundo, todo esto no solo es observado a través de gráficas sino determinado por métricas como el error o el error cuadrático medio. Aunque en el presente trabajo se han incluido algunas formas y maneras de utilizar los conceptos del aprendizaje automático para la teoría de Control Automático, sería de un interés fuerte desarrollar aplicaciones para Industria 4.0 y gemelos digitales, ya que puede ofrecer diversas soluciones a las problemáticas presentadas en el campo e incluso abordar tópicos de mantenimiento predictivo.

Bibliografía

[1] L. Ljung, "System identification," *Wiley Encyclopedia of Electrical and Electronics Engineering*, pp. 1–19, 1999.

[2] G. Horvath, "Neural networks in system identification," *Nato Science Series Sub Series III Computer And Systems Sciences*, vol. 185, pp. 43–78, 2003.

[3] K. Narendra and K. Parthasarathy, "Gradient methods for optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, 1991.

[4] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1–1, 05 2021.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[6] G. E. P. Box, G. C. Reinsel, G. M. Ljung, and G. M. Jenkins, *Time Series Analysis: Forecasting and Control.* New Jersey: Wiley, 2015.

[7] A. Tealab, "Time series forecasting using artificial neural networks methodologies: A systematic review," *Future Computing and Informatics Journal*, 2018.

[8] M. Mushtaq, U. Akram, M. Aamir, H. Ali, and M. Zulqarnain, "Neural network techniques for time series prediction: A review," *International Journal on informatics visualization*, 2019.

[9] S. B. Taieb, A. Sorjamaa, and G. Bontempi, "Multiple-output modeling for multi-step-ahead time series forecasting," *Neurocomputing*, 2010.

[10] M. P. Clements, P. H. Franses, and N. R. Swanson, "Forecasting economic and financial time-series with non-linear models," *International Journal of Forescasting*, 2004.

[11] W. Bi, X. Wang, T. Zheng, and H. Tamura, "Avoiding the local minima problem in backpropagation algorithm with modified error function," *IEICE Trans. Fundamentals, Vol.E88*, 2005.

[12] B. Lim and S. Zohren, "Time series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A 2020*, 2020.

[13] C. Chatfield and X. Haipeng, *The Analysis of Time Series: An Introduction with R*. Chapman and Hall/CRC, 2019.

[14] A. K. Palit and D. Popovic, *Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications*. Springer, 2005.

[15] S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. United Kingdom: Wiley, 2013.

[16] J. D. Scargle, "An introduction to chaotic and random time series analysis," *International Journal of Imaging Systems and Technology*, vol. 1, no. 2, pp. 243–253, 1989. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ima.1850010213

[17] R. Brown, *Smoothing, Forecasting and Prediction of Discrete Time Series*. Prentice-Hall, 1963.

[18] Z. Liu, Z. Zhu, J. Gao, and C. Xu, "Forecast methods for time series data: A survey," *IEEE Access*, vol. 9, pp. 91 896–91 912, 2021.

[19] C. Aggarwal, *Neural Networks and Deep Learning.* Springer International Publishing, 2018.

[20] E. Agirre-Basurko, G. Ibarra-Berastegi, and I. Madariaga, "Regression and multilayer perceptron-based models to forecast hourly o3 and no2 levels in the bilbao area," *Environmental Modelling Software*, vol. 21, no. 4, pp. 430–446, 2006, urban Air Quality Modelling.

[21] G. Dudek, "Multilayer perceptron for gefcom2014 probabilistic electricity price forecasting," *International Journal of Forecasting*, vol. 32, no. 3, pp. 1057–1060, 2016.

[22] U. Orhan, M. Hekim, and M. Ozer, "Eeg signals classification using the k-means clustering and a multilayer perceptron neural network model," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13 475–13 481, 2011.

[23] R. Manger and K. Puljić, "Multilayer perceptrons and data compression," *Computing and Informatics*, vol. 26, no. 1, pp. 45–62, 2012.

[24] C. Miroslav and K. Štefan, "Neural networks for real time control systems," *IFAC Proceedings Volumes*, vol. 36, no. 18, pp. 135 – 142, 2003, 2nd IFAC Conference on Control Systems Design, Bratislava, Slovak Republic, 7-10 September 2003.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[26] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[27] S. Haykin, *Neural Networks and Learning Machines*, 2010.

[28] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach.* Pearson, 2010.

[29] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[30] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76–86, 1992.

[31] Y. Filiberto Cabrera, R. Bello Pérez, Y. C. Mota, and G. R. Jimenez, "Improving the mlp learning by using a method to calculate the initial weights of the network based on the quality of similarity measure," in *Advances in Soft Computing*, I. Batyrshin and G. Sidorov, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 351–362.

[32] B. Fallah, K. T. W. Ng, H. L. Vu, and F. Torabi, "Application of a multi-stage neural network approach for time-series landfill gas modeling with missing data imputation," *Waste Management*, vol. 116, pp. 66–78, 2020.

[33] O. Simeone, S. Park, and J. Kang, "From learning to meta-learning: Reduced training overhead and complexity for communication systems," 03 2020, pp. 1–5.

[34] H. F. Harlow, "The formation of learning sets," *Psychological Review*, 1949.

[35] J. Biggs, "The role of metalearning in study processes," *British Journal of Educational Psychology*, vol. 55, pp. 185 – 212, 05 2011.

[36] S. Thrun and L. Y. Pratt, "Learning to learn: Introduction and overview," in *Learning to Learn*, 1998.

[37] A. Schrier, "Learning how to learn: The significance and current status of learning set formation," *Primates*, 1984.

[38] J. Schmidhuber, *"Evolutionary Principles In Self-referential Learning," On learning how to learn: The meta-meta-... hook*, 1987.

[39] Y. Bengio, S. Bengio, and J. Cloutier, "Learning a synaptic learning rule," *IJCNN*, 1990.

[40] S. Bengio, Y. Bengio, and J. Cloutier, "On the search for new learning rules for anns," *Neural Processing Letters*, 1995.

[41] J. Schmidhuber, J. Zhao, and M. Wiering, "Simple principles of meta-learning," *echnical report IDSIA*, 1996.

[42] J. Schmidhuber, "Neural network that embeds its own metalevels," *IEEE International Conference On Neural Networks*, 1993.

[43] J.-A. Meyer and S. W. Wilson, *A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers*, 1991, pp. 222–227.

[44] A. S. Younger, S. Hochreiter, and P. R. Conwell, "Meta-learning with backpropagationt," *ICANN*, 2001.

[45] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," *ICANN*, 2001.

[46] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial intelligence review,*, 2002.

[47] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *ICML,*, 2017.

[48] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," *ICML*, 2017.

[49] R. Houthooft, Y. Chen, P. Isola, B. C. Stadie, J. H. F. Wolski, and P. Abbeel, "Evolved policy gradients," *NeurIPS*, 2018.

[50] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *ICLR*, 2018.

[51] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learned optimizers that scale and generalize," *ICML*, 2017.

[52] Y. Balaji, S. Sankaranarayanan, and R. Chellappa, "Metareg: Towards domain generalization using meta-regularization," *NeurNIPS*, 2018.

[53] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Learning to learn from noisy labeled data," *CVPR*, 2019.

[54] S. Ravi and H. Larochelle, "Optimization as a model for few- shot learning," *ICLR*, 2016.

[55] Y. Y. Li, W. Zhou, and T. M. Hospedales, "Feature-critic networks for heterogeneous domain generalization," *ICML*, 2019.

[56] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward and reverse gradient-based hyperparameter optimization," *ICML*, 2017.

[57] P. Micaelli and A. Storkey, "Non-greedy gradient-based hyperparameter optimization over long horizons," *arXiv*, 2020.

[58] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," *ICML*, 2018.

[59] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing millions of hyperparameters by implicit differentiation," *AISTATSL*, 2020.

[60] A. Rajeswaran, C. Finn, S. Kakade, and S. Levine, "Meta-learning with implicit gradients," *NeurIPS*, 2019.

[61] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta- learning algorithms," *ArXiv*, 2018.

[62] S. Pan, V. Zheng, Q. Yang, and D. Hu, "Transfer learning for wifi-based indoor localization," *Proc. Workshop Transfer Learning for Complex Task of the 23rd Assoc. for the Advancement of Artificial Intelligence (AAAI) Conf. Artificial Intelligence*, 2008.

[63] J. Jiang and C. Zhai, "Instance weighting for domain adaptationin nlp," *Proc. 45th Ann. Meeting of the Assoc. Computational Linguistics*, 2007.

[64] X. Liao, Y. Xue, and L. Carin, "Logistic regression with an auxiliary data source," *Proc. 45th Ann. Meeting of the Assoc. Computational Linguistics*, 2005.

[65] S. Bickel, M. Bruckner, and T. Scheffer, "Discriminative learning for differing training and test distributions," *Proc. 24th Int'l Conf. Machine Learning*, 2007.

[66] M. Sugiyama, S. Nakajima, H. Kashima, P. Buenau, and M.Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," *Proc. 20th Ann. Conf. Neural Information Processing Systems*, 2008.

[67] A. Argyriou, C. Micchelli, M. Pontil, and Y. Ying, "A spectral regularization framework for multi-task structure learning," *Proc. 20th Ann. Conf. Neural Information Processing Systems*, 2008.

[68] T. Jebara, "Multi-task feature and kernel selection for svms," *Proc. 21st Int'l Conf. Machine Learning*, 2004.

[69] S. Lee, V. Chatalbashev, D. Vickrey, and D. Koller, "Learning a meta-level prior for feature relevance from multiple related tasks," *Proc. 21st Int'l Conf. MachineLearning*, 2007.

[70] C. Wang and S. Mahadevan, "Manifold alignment using procrustes analysis," *Proc. 25th Int'l Conf. Machine Learning*, 2008.

[71] E. Bonilla, K. Chai, and C. Williams, "Multi-task gaussian process prediction," *Proc. 20th Ann. Conf. Neural Information Processing Systems*, 2008.

[72] A. Schwaighofer, V. Tresp, and K. Yu, "Learning gaussian process kernels via hierarchical bayes," *Proc. 17th Ann. Conf. Neural Information Processing Systems*, 2004.

[73] N. Lawrence and J. Platt, "Learning to learn with the informative vector machine," *Proc. 21st Int'l Conf. MachineLearning*, 2004.

[74] L. Mihalkova, T. Huynh, and R. Mooney, "Mapping and revising markov logic networks for transfer learning," *Proc. 22nd Assoc. for the Advancement of Artificial Intelligence (AAAI) Conf. Artificial Intelligence*, 2007.

[75] L. Mihalkova and R. Mooney, "Transfer learning by mapping with minimal target data," *Proc. Assoc. for the Advancement of Artificial Intelligence (AAAI '08) Workshop Transfer Learning for Complex Tasks*, 2008.

[76] J. Davis and P. Domingos, "Deep transfer via second-order markov logic," *Proc. Assoc. for the Advancement of Artificial Intelligence (AAAI '08) Workshop Transfer Learning for Complex Tasks*, 2008.

[77] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data," *Proc. 24th Int'l Conf. Machine Learning*, 2007.

[78] P. Addison, *The Illustrated Wavelet Transform Handbook.* IOP Publishing Ltd, 2002.

[79] N. G. Françoise, J. and T. T., *Wavelets: Applications.* Academic Press, 2006.

[80] S. Padma, G. Hariharan, and S. Castellucci, "A new spectral method applied to immobilized biocatalyst model arising in biochemical engineering," 2016.

[81] H. G., *Wavelet Analysis—An Overview. In: Wavelet Solutions for Reaction–Diffusion Problems in Science and Engineering.* Forum for Interdisciplinary Mathematics. Springer, 2019.

[82] a. L. V. Telesca, L. and N. Alexis, "Multiresolution wavelet analysis of earthquakes," *Chaos, Solitons and Fractals*, 2004.

[83] S. E.D., "Input to state stability: Basic concepts and results," *In: Nistri P., Stefani G. (eds) Nonlinear and Optimal Control Theory. Lecture Notes in Mathematics. Springer, Berlin, Heidelberg*, vol. 1932, no. 1, 2008.

[84] Z. P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems," *Automatica*, 2001.

[85] G. Cybenko, "Aproximation by superposition of sigmoidal activation function," *Math. Control. Signals Syst*, 1989.

[86] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," 2020.

[87] K. Narendra and K. Parthasarathy, "Gradient methods for optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, 1991.

[88] T. Fujinaka, Y. Kishida, M. Yoshioka, and S. Omatu, "Stabilization of double inverted pendulum with self-tuning neuro-pid," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 4, 2000, pp. 345–348 vol.4.

[89] W. Yu and X. Li, "Discrete-time neuro identification without robust modification," *IEE Proc.-Control Theory Appl.*, vol. 1932, no. 3, 2003.

[90] E. de la Rosa and W. Yu, "Randomized algorithms for nonlinear system identification with deep learning modification," *Information Sciences*, vol. 364-365, pp. 197–212, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025515007033

[91] V. Lapenna, G. Martinelli, and L. Telesca, "Long-range correlation analysis of earthquake-related geochemical variations recorded in central italy," *Chaos, Solitons & Fractals*, vol. 21, no. 2, pp. 491–500, 2004.

[92] N. Matsumoto, Y. Kitagawa, and N. Koizumi, "Groundwater-level anomalies associated with a hypothetical preslip prior to the anticipated tokai earthquake: Detectability using the groundwater observation network of the geological survey of japan, aist," in *Terrestrial Fluids, Earthquakes and Volcanoes: The Hiroshi Wakita Volume II*. Springer, 2007, pp. 2377–2396.

[93] E. Petraki, D. Nikolopoulos, C. Nomicos, J. Stonham, D. Cantzos, P. Yannakopoulos, and S. Kottou, "Electromagnetic pre-earthquake precursors: Mechanisms, data and models-a review," *Journal of Earth Science & Climatic Change*, vol. 6, no. 1, p. 1, 2015.

[94] P. Varotsos, N. V. Sarlis, and E. S. Skordas, *Natural time analysis: the new view of time: precursory seismic electric signals, earthquakes and other complex time series*. Springer Science & Business Media, 2011.

[95] S. Uyeda, M. Hayakawa, T. Nagao, O. Molchanov, K. Hattori, Y. Orihara, K. Gotoh, Y. Akinaga, and H. Tanaka, "Electric and magnetic phenomena observed before the volcano-seismic activity in 2000 in the izu island region, japan," *Proceedings of the National Academy of Sciences*, vol. 99, no. 11, pp. 7352–7355, 2002.

[96] Y. Yasuoka, H. Nagahama, J. Muto, and T. Mukai, "The anomaly in atmospheric radon concentrations prior to the 2011 tohoku-oki earthquake in japan," *Radiat. Environ. Med.*, vol. 7, pp. 86–94, 2018.

[97] R. Prakash and H. Srivastava, "Thermal anomalies in relation to earthquakes in india and its neighbourhood," *Current science*, pp. 2071–2082, 2015.

[98] G. Martinelli, "Hydrogeologic and geochemical precursors of earthquakes: an assessment for possible applications." *Bollettino di Geofisica Teorica ed Applicata*, vol. 56, no. 2, 2015.

[99] L. Telesca and M. Lovallo, "Non-uniform scaling features in central italy seismicity: A non-linear approach in investigating seismic patterns and detection of possible earthquake precursors," *Geophysical Research Letters*, vol. 36, no. 1, 2009.

[100] F. Corbi, L. Sandri, J. Bedford, F. Funiciello, S. Brizzi, M. Rosenau, and S. Lallemand, "Machine learning can predict the timing and size of analog earthquakes," *Geophysical Research Letters*, vol. 46, no. 3, pp. 1303–1311, 2019.

[101] H. Adeli and A. Panakkat, "A probabilistic neural network for earthquake magnitude prediction," *Neural networks*, vol. 22, no. 7, pp. 1018–1024, 2009.

[102] A. De Santis, G. Cianchini, P. Favali, L. Beranzoli, and E. Boschi, "The gutenberg–richter law and entropy of earthquakes: Two case studies in central italy," *Bulletin of the Seismological Society of America*, vol. 101, no. 3, pp. 1386–1395, 2011.

[103] Y. Ogata, "Significant improvements of the space-time etas model for forecasting of accurate baseline seismicity," *Earth, planets and space*, vol. 63, no. 3, pp. 217–229, 2011.

[104] T. Perol, M. Gharbi, and M. Denolle, "Convolutional neural network for earthquake detection and location," *Science Advances*, vol. 4, no. 2, p. e1700578, 2018.

[105] J. Wang and T.-L. Teng, "Artificial neural network-based seismic detector," *Bulletin of the Seismological Society of America*, vol. 85, no. 1, pp. 308–319, 1995.

[106] J. Wang and T.-l. Teng, "Identification and picking of s phase using an artificial neural network," *Bulletin of the Seismological Society of America*, vol. 87, no. 5, pp. 1140–1149, 1997.

[107] G. Asencio-Cortés, F. Martínez-Álvarez, A. Troncoso, and A. Morales-Esteban, "Medium–large earthquake magnitude prediction in tokyo with artificial neural networks," *Neural Computing and Applications*, vol. 28, no. 5, pp. 1043–1055, 2017.

[108] E. Florido, J. L. Aznarte, A. Morales-Esteban, and F. Martínez-Álvarez, "Earthquake magnitude prediction based on artificial neural networks: A survey," *Croatian Operational Research Review*, pp. 159–169, 2016.

[109] K. Asim, F. Martínez-Álvarez, A. Basit, and T. Iqbal, "Earthquake magnitude prediction in hindukush region using machine learning techniques," *Natural Hazards*, vol. 85, no. 1, pp. 471–486, 2017.

[110] S. Narayanakumar and K. Raja, "A bp artificial neural network model for earthquake magnitude prediction in himalayas, india," *Circuits and Systems*, vol. 7, no. 11, pp. 3456–3468, 2016.

[111] K. M. Asim, F. Javed, S. Hainzl, and T. Iqbal, "Fault parameters-based earthquake magnitude estimation using artificial neural networks," *Seismological Research Letters*, vol. 90, no. 4, pp. 1544–1551, 2019.

[112] A. Panakkat and H. Adeli, "Neural network models for earthquake magnitude prediction using multiple seismicity indicators," *International journal of neural systems*, vol. 17, no. 01, pp. 13–33, 2007.

[113] M. Ibrahim, J. Park, and N. Athens, "Earthquake warning system: Detecting earthquake precursor signals using deep neural networks," *Technical Report CS 230*, 2018.

[114] J. Huang, X. Wang, Y. Zhao, C. Xin, and H. Xiang, "Large earthquake magnitude prediction in taiwan based on deep learning neural network," *Neural Network World*, vol. 28, no. 2, pp. 149–160, 2018.

[115] D. Kirschner, N. Howes, C. Daly, J. Mukherjee, and J. Li, "Detecting p-and s-wave arrivals with a recurrent neural network," in *SEG International Exposition and Annual Meeting*. OnePetro, 2019.

[116] Q. Wang, Y. Guo, L. Yu, and P. Li, "Earthquake prediction based on spatio-temporal data mining: an lstm network approach," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 1, pp. 148–158, 2017.

[117] M. Shiblee, P. K. Kalra, and B. Chandra, "Time series prediction with multilayer perceptron (mlp): a new generalized error based approach," in *International Conference on Neural Information Processing*. Springer, 2008, pp. 37–44.

[118] L. Telesca, C.-c. Chen, and M. Lovallo, "Investigating the relationship between seismological and topological properties of seismicity in italy and taiwan," *Pure and Applied Geophysics*, vol. 177, no. 9, pp. 4119–4126, 2020.

[119] J. Hooyberghs, C. Mensink, G. Dumont, F. Fierens, and O. Brasseur, "A neural network forecast for daily average pm_{10} concentrations in belgium," *Atmospheric Environment, 39, 3279–3289.*, 2005.

[120] M. Cai, Y. Yin, and M. Xie, "Prediction of hourly air pollutant concentrations near urban arterials using artificial neural network approach," *Transportation Research Part D: Transport and Environment, 14, 32–41*, 2009.

[121] W. Wu, N. Zhang, Z. Li, L. Li, and Y. Liu, "Convergence of the gradient method with momentum for back-propagation neural networks," *Journal of Computational Mathematics*, 2008.

[122] S. M. Cabaneros, J. K. Calautit, and B. R. Hughes, "A review of artificial neural network models for ambient air pollution prediction," *Environmental Modelling and Software*, 2019.

[123] A. B. Chelani, D. Gajghate, and M. Hasan, "Prediction of ambient pm_{10} and toxic metals using artificial neural networks," *Journal of the Air and Waste Management Association, 52, 805–810.*, 2002.

[124] I. G. McKendry, "Evaluation of artificial neural networks for fine particulate pollution (pm_{10} and $pm_{2.5}$) forecasting," *Journal of the Air and Waste Management Association, 52, 1096–1101.*, 2002.

[125] A. Chaloulakou, G. Grivas, and N. Spyrellis, "Neural network and multiple regression models for pm_{10} prediction in athens: a comparative assessment," *Journal of the Air and Waste Management Association, 53, 1183–1190.*, 2003.

[126] G. Corani, "Air quality prediction in milan: feed-forward neural networks, pruned neural networks and lazy learning," *Ecological Modelling, 185, 513–529.*, 2005.

[127] G. Grivas and A. Chaloulakou, "Artificial neural network models for prediction of pm_{10} hourly concentrations, in the greater area of athens, greece," *Atmospheric Environment, 40, 1216–1229.*, 2006.

[128] P. Perez and J. Reyes, "An integrated neural network model for pm_{10} forecasting," *Atmospheric Environment, 40, 2845–2851.*, 2006.

[129] D. K. Papanastasiou, D. Melas, and I. Kioutsioukis, "Development and assessment of neural network and multiple regression models in order to predict pm_{10} levels in a medium-sized mediterranean city," *Water, Air, and Soil Pollution, 182, 325–334*, 2007.

[130] L. Hrust, Z. B. Klaić, J. Križan, O. Antonić, and P. Hercog, "Neural network forecasting of air pollutants hourly concentrations using optimised temporal averages of meteorological variables and pollutant concentrations," *Atmospheric Environment, 43, 5588–5596*, 2009.

[131] A. K. Paschalidou, S. Karakitsios, S. Kleanthous, and P. A. Kassomenos, "Forecasting hourly pm_{10} concentration in cyprus through artificial neural networks and multiple

regression models: Implications to local environmental management," *Environmental Science and Pollution Research, 18, 316–327*, 2011.

[132] H. J. S. Fernando, M. C. Mammarella, G. Grandoni, P. Fedele, R. D. Marco, R. Dimitrova, and P.Hyde, "Forecasting pm_{10} in metropolitan areas: Efficacy of neural networks," *Environmental Pollution, 163, 62–67.*, 2012.

[133] P. Perez, "Combined model for pm_{10} forecasting in a large city," *Atmospheric Environment, 60, 271–276.*, 2012.

[134] F. Nejadkoorki and S. Baroutian, "Forecasting pm_{10} in metropolitan areas: Efficacy of neural networks," *Int. J. Environ. Res. 6, 277–284.*, 2012.

[135] W. Liu, X. Li, Z. Chen, G. Zeng, T. León, J. Liang, G. Huang, Z. Gao, S. Jiao, X. He, and M. Lai, "Land use regression models coupled with meteorology to model spatial and temporal variability of no2 and pm10 in changsha, china," *Atmos. Environ. 116, 272–280*, 2015.

[136] M. G. Cortina–Januchs, J. Quintanilla–Dominguez, A. Vega–Corona, and D. Andina, "Development of a model for forecasting of pm_{10} concentrations in salamanca, mexico," *Atmospheric Pollution Research*, 2015.

[137] J. M. Barrón-Adame, M. G. Cortina-Januchs, A. Vega-Corona, and D.Andina, "Unsupervised system to classify so_2 pollutant concentrations in salamanca, mexico," *Expert Systems with Applications*, 2012.

[138] J. B. Ordieres, E. P.Vergara, R. S. Capuz, and R. E. Salazar, "Neural network prediction model for fine particulate matter ($pm_{2.5}$) on the us–mexico border in el paso (texas) and ciudad juárez (chihuahua)," *Environmental Modelling and Software*, 2005.

[139] S. d. M. A. Dirección de Monitoreo Atmosférico, "Url http://www.aire.cdmx.gob.mx/default.php," Secretaría del Medio Ambiente de la Ciudad de México., Tech. Rep., 2020.

[140] M. Maya, W. Yu, and L. Telesca, "Multi-step forecasting of earthquake magnitude using meta-learning based neural networks," *Cybernetics and Systems*, vol. 0, no. 0, pp. 1–18, 2021.

[141] J. H. Dieterich, "Modeling of rock friction: 1. experimental results and constitutive equations," *J. Geophys*, pp. 2161–2168, 1978.

[142] B. K. Shivamoggi, *Nonlinear Dynamics and Chaotic Pheonmena: An Introduction*, 2014.

[143] A. Gualandil, J.-P. Avouac, and D. F. S. Michel, "The predictable chaos of slow earthquakes," *Sci. Adv*, 2020.

[144] J. K. Gardner and L. Knopoff, "Is the sequence of earthquakes in southern california, with aftershocks removed, poissonian?" *Bulletin of the Seismological Society of America*, vol. 64, pp. 1363–1367, 1974.

[145] E. Field, G. Biasi, P. Bird, T. Dawson, K. Felzer, D. Jackson, K. Johnson, T. Jordan, C. Madden, A. Michael, K. Milner, M. Page, T. Parsons, P. Powers, B. Shaw, W. Thatcher, R. Weldon, and Y. Zeng, "Long-term time-dependent probabilities for the third uniform california earthquake rupture forecast (ucerf3)," *Bulletin of the Seismological Society of America*, vol. 105, 03 2015.

[146] W. F. Brace and J. D. Byerlee, "Stick-slip as a mechanism for earthquakes," *Science*, pp. 990–992, 1966.

[147] N. Kato, "Earthquake cycles in a model of interacting fault patches: Complex behavior at transition from seismic to a seismic slip," *Bulletin of the Seismological Society of America*, vol. 106, pp. 1172–1787, 2016.

[148] J. Huang and D. L. Turcotte, "Evidence for chaotic fault interactions in the seismicity of the san andreas fault and nankai trough," *Nature*, vol. 348, pp. 234–236, 1990.

[149] M. Banna, H. Al, K. A. Taher, M. S. Kaiser, M. Mahmud, M. S. Rahman, A. S. M. S. Hosen, and G. H. Cho, "Application of artificial intelligence in predicting earthquakes: State-of-the-art and future challenges," *IEEE Access*, vol. 8, pp. 192 880–192 923, 2020.

[150] K. Soltesz and A. Cervin, "When is pid a good choice?" *IFAC-PapersOnLine*, vol. 51, no. 4, pp. 250–255, 2018, 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control PID 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318303707

[151] A. G. Brito, "On the misunderstanding of the ziegler-nichols's formulae usage," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 1, pp. 142–147, 2019.

[152] K. Åström and T. Hägglund, "Revisiting the ziegler–nichols step response method for pid control," *Journal of Process Control*, vol. 14, no. 6, pp. 635–650, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959152404000034

[153] C. Anil and R. P. Sree, "Tuning of pid controllers for integrating systems using direct synthesis method," *ISA transactions*, vol. 57, pp. 211–219, 2015.

[154] Q. B. Jin and Q. Liu, "Imc-pid design based on model matching approach and closed-loop shaping," *ISA Transactions*, vol. 53, no. 2, pp. 462–473, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0019057813001948

[155] A. Leva, "Pid-based controls in computing systems: a brief survey and some research directions," *IFAC-PapersOnLine*, vol. 51, no. 4, pp. 805–810, 2018, 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control PID 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318304750

[156] J. Moreno-Valenzuela, R. Pérez-Alcocer, M. Guerrero-Medina, and A. Dzul, "Nonlinear pid-type controller for quadrotor trajectory tracking," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 5, pp. 2436–2447, 2018.

[157] K.-m. Hu, Y.-z. Li, and X.-m. Guan, "Research of the pipe flow measurement and control system based on bp neural networks pid," in *Advances in Technology and Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[158] K. Pirabakaran and V. Becerra, "Pid autotuning using neural networks and model reference adaptive control," *IFAC Proceedings Volumes*, vol. 35, pp. 451 – 456, 2002, 15th IFAC World Congress.

[159] J. Chen and T.-C. Huang, "Applying neural networks to on-line updated pid controllers for nonlinear process control," *Journal of Process Control*, vol. 14, pp. 211 – 230, 2004.

[160] Y. Yang and Q. Wu, "A neural network pid control for ph neutralization process," in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 3480–3483.

[161] L. E. Ramos-Velasco, O. A. Domínguez-Ramírez, and V. Parra-Vega, "Wavenet fuzzy pid controller for nonlinear mimo systems: Experimental validation on a high-end haptic robotic interface," *Applied Soft Computing*, vol. 40, pp. 199–205, 03 2016.

[162] M. Sedighizadeh and A. Rezazadeh, "Adaptive pid control of wind energy conversion systems using rasp1 mother wavelet basis function networks," *International Journal of Electrical and Computer Engineering*, vol. 2, pp. 62–66, 2008. [Online]. Available: https://publications.waset.org/vol/13

[163] A. Singh and A. Kaur, "Tuning techniques of pid controller: A review," in *National Conference on Advances in Engineering and Technology of International Journal of Engineering Research and Applications*, 2014.

[164] J. Corriou, *Models and Methods for Parametric Identification. In: Process Control*. London: Springer, 2004.