UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSÍ

FACULTAD DE CIENCIAS QUÍMICAS

DOCTORADO EN CIENCIAS EN INGENIERÍA QUÍMICA

MODELADO DE PROCESOS CON RNAs: ENTRENAMIENTO RÁPIDO USANDO PESOS ESTOCÁSTICOS CON CAPACIDAD DE GENERALIZACIÓN

TESIS PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS EN INGENIERÍA QUÍMICA

PRESENTA:

M.C. HÉCTOR JOSUÉ CANO ROCHA

DIRECTOR DE TESIS:

DR. RAÚL GONZÁLEZ GARCÍA

San Luis Potosí, S.L.P., julio 2022

El Programa de Doctorado en Ciencias en Ingeniería Química de la Universidad Autónoma de San Luis Potosí pertenece al Programa Nacional de Posgrados de Calidad (PNPC) del CONACYT, registro 000897, en el Nivel Consolidado.

Número de registro de la beca otorgada por CONACYT: 586482

UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSÍ

FACULTAD DE CIENCIAS QUÍMICAS

DOCTORADO EN CIENCIAS EN INGENIERÍA QUÍMICA

MODELADO DE PROCESOS CON RNAs: ENTRENAMIENTO RÁPIDO USANDO PESOS ESTOCÁSTICOS CON CAPACIDAD DE GENERALIZACIÓN

TESIS PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS EN INGENIERÍA QUÍMICA

PRESENTA:

M.C. HÉCTOR JOSUÉ CANO ROCHA

DIRECTOR DE TESIS:

DR. RAÚL GONZÁLEZ GARCÍA

SINODALES:

DR. RAÚL GONZÁLEZ GARCÍA _____

DRA. ALMA GABRIELA PALESTINO ESCOBEDO _____

DR. RAÚL OCAMPO PÉREZ _____

DR. ERIK CÉSAR HERRERA HERNÁNDEZ _____

DR. JUAN CARLOS CUEVAS TELLO _____

San Luis Potosí, S.L.P., JULIO 2022

UASLP
Universidad Autónoma de San Luis Potosí

FACULTAD DE
CIENCIAS QUÍMICAS

San Luis Potosí S.L.P. México

Julio, 2022

Comité Académico del Posgrado en Ciencias en Ingeniería Química

Facultad de Ciencias Químicas

Presente:

Por medio de la presente comunicamos que la tesis llevada a cabo por el alumno de Doctorado en Ciencias en Ingeniería Química, M.C. **Héctor Josué Cano Rocha**, titulada "**MODELADO DE PROCESOS CON RNAs: ENTRENAMIENTO RÁPIDO USANDO PESOS ESTOCÁSTICOS CON CAPACIDAD DE GENERALIZACIÓN**", ha sido concluida y aprobada por el comité tutorial para dar inicio a los trámites correspondientes para su titulación, la cual tendrá lugar el próximo día 20 de julio del 2022 a las 12:00 horas en sala audiovisual de la hemeroteca (K-102) de la Facultad de Ciencias Químicas.

ATENTAMENTE

Director de tesis

_____

Dr. Raúl González García

FCQ/UASLP

Sinodal

Sinodal

_____ _____

Dra. Alma Gabriela Palestino Escobedo

FCQ/UASLP

Dr. Raúl Ocampo Pérez

FCQ/UASLP

Sinodal

Sinodal externo

_____ _____

Dr. Erik César Herrera Hernández

FCQ/UASLP

Dr. Juan Carlos Cuevas Tello

FI/UASLP

www.uaslp.mx

Av. Dr. Manuel Nava Núm. 6
Zona Universitaria • CP 78210
San Luis Potosí, S.L.P.
tel. (444) 826 24 40 al 46
fax (444) 826 2372

Agradecimientos

- A mi querida esposa por estar siempre a mi lado en los momentos más difíciles y por siempre apoyarme para superar cada obstáculo. Gracias a tu amor y apoyo he alcanzado muchas metas en mi vida.
- A mi hermosa y amada hija, Celeste, por ser paciente con tu padre cuando tenía que trabajar y en algunos momentos no podía estar a tu lado y ver como cada día crecías. Te prometo estar más tiempo contigo.
- A mi querido hijo, Jona, por ser uno de mis principales motores para ser cada día mejor.
- A mi asesor, el Dr. Raúl González García, quien he pasado horas hablando de temas muy interesantes. Gracias por su paciencia conmigo, tal vez no fui el mejor de sus alumnos, pero usted para mí ha sido y siempre será el mejor profesor, quien me ha enseñado bastante y no solo de cuestiones académicas. Cada comentario, cada consejo que me ha dado en estos 8 años me ha hecho ser un mejor profesionista y una persona más capaz. En usted encontré no solo un profesor, sino también un ejemplo a seguir. En verdad muchas gracias por todo, sin usted no sería lo que soy el día de hoy.
- A mis padres, Cole y Concho, que siempre han estado al pendiente de mí y gracias a su esfuerzo y dedicatoria he alcanzado muchas victorias.
- A mis compañeros de posgrado, Diego, Bernardo, Andrés y Camilo por siempre apoyarnos como un equipo.
- Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico otorgado para la realización de este proyecto por medio de la beca con No. 586482.

Resumen

En este trabajo se estudia, análisis y aplica de un método estocásticos de un solo paso ó Stochastic One-Step (SOS), método no iterativo e instantáneo, para el entrenamiento de redes neuronales artificiales de alimentación hacia adelante. El método Stochastic One-Step hace uso de pesos estocásticos (valores aleatorios) durante la etapa de entrenamiento. Un hallazgo interesante es cuando el método propuesto se aplica a sistemas de múltiples entradas y múltiples salidas, mediante el uso de una matriz amputada, garantizando que cada variable de salida obtenga su topología óptima, dicho efecto no se encuentra en los métodos de entrenamiento tradicionales. Para el desarrollo de la metodología se hace uso de datos simulados (reconstrucción de tres superficies), y datos de situaciones reales. Además, se prueba la factibilidad del método haciendo comparaciones rigurosas contra métodos tradicionales, donde el método propuesto resultó superior hasta un 50 % bajo criterios estadísticos. En algunos casos el método propuesto logró ser hasta 100 veces más rápido que los métodos de entrenamiento tradicionales y con calidades de predicción competentes a los métodos de preferencia. Algunos trabajos presentados en la literatura mencionan que una red neuronal artificial de una sola capa oculta con alimentación hacia adelante no cuenta con la misma capacidad de abstracción que las redes neuronales profundas, por lo tanto, se utilizó el conjunto de datos de MNIST, base de datos comúnmente utilizada en técnicas de aprendizaje automático, para evidenciar que una SLFN puede ofrecer resultados aceptables si es entrenada mediante el método SOS. Los resultados para el MNIST fueron de 98.15 % de precisión, con un tiempo de entrenamiento de 1.27 horas, donde se realizó un barrido desde 1 hasta 9'000 neuronas ocultas. Con el objetivo de mejorar el método propuesto, se analizan distintos factores que influyen fuertemente en su desempeño, como el rango en el que se inicializan los parámetros (pesos estocásticos), que tienen gran impacto en el desempeño final. Como el método SOS entrena redes neuronales de una forma rápida, fue posible realizar estudios más detallados. Uno de estos estudios fue implementar técnicas de preprocesamiento como el análisis de componentes principales, logrando reducir la dimensionalidad de bases de datos sustancialmente grandes y a la par identificando la cantidad óptima de componentes principales. Al final se plantea una metodología para desarrollar un entrenamiento rápido y eficaz, donde el usuario no tenga que definir ningún parámetro de ajuste. La metodología propuesta tiene como objetivo establecer reglas fijas sobre como entrenar una red neuronal artificial.

Palabras clave: Red neuronal de alimentación hacia adelante, Redes constructivas, Entrenamiento, Validación cruzada, Red neuronal de alimentación hacia adelante de una sola capa oculta, Múltiples respuestas.

Abstract

This work analyzes and applies a Stochastic One-Step (SOS) method, a non-iterative and instantaneous method, for the training of feed-forward artificial neural networks. The Stochastic One-Step method makes use of stochastic weights (random values) during the training stage. An interesting finding is when the proposed method is applied to multi-input, multi-output systems, by using an amputated matrix, guaranteeing that each output variable obtains its optimal topology, such an effect is not found in traditional training methods. For the development of the methodology, simulated data (reconstruction of three surfaces) and data from real situations are used. In addition, the feasibility of the method is tested by making rigorous comparisons against traditional methods, where the proposed method was superior up to 50 % under statistical criteria. In some cases, the proposed method was able to be up to 100 times faster than traditional training methods and with prediction qualities comparable to the preferred methods. Some works presented in the literature mention that a neural network artificial single layer concealed feedforward does not have the same abstraction capacity as networks deep neural networks, therefore, the set of data from MNIST, a database commonly used in techniques of machine learning, to show that an SLFN can offer acceptable results if trained using the SOS method. The results for the MNIST were 98.15 % accurate, with a time of training of 1.27 hours, where a sweep was made from 1 up to 9'000 hidden neurons.

To improve the proposed method, different factors that strongly influence its performance are analyzed, such as the range in which the parameters are initialized (stochastic weights), which have a great impact on the final performance.

Since the SOS method trains neural networks quickly, it was more detailed studies possible. One of these studies was implementing preprocessing techniques such as analysis of main components, managing to reduce the dimensionality of bases of substantially large data, and at par by identifying the optimal number of principal components. Finally, a methodology to develop fast and effective training, where the user does not need to define any tuning parameters. The methodology proposal aims to establish fixed rules on how to train an artificial neural network.

Keywords: Feedforward neural network, Constructive networks, Training, Cross-validation, Single-hidden layer feedforward network, Multiple responses.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Con el fin de representar el comportamiento de un proceso y sus interrelaciones de causa-efecto, es de vital importancia la implementació n adecuada de modelos, tanto en la ingeniería de procesos como en otras áreas. Dentro de este marco existen varias formas para desarrollar modelos que representen al sistema de interés y que a su vez describan su comportamiento lo más real posible. Por un lado, los modelos teóricos se plantean a través de la aplicación de leyes naturales y otros principios que gobiernan al sistema. Por el otro lado, un modelo empírico, se desarrolla a partir de la información obtenida experimentalmente del sistema para obtener una relación descriptiva mediante una función matemática. El describir un sistema físico y posteriormente reducirlo a un modelo, cualquiera que éste sea, es un procedimiento que consta de varias etapas: la definición del problema, la formulación del modelo, la estimación de parámetros y la validación del modelo.

La desventaja de los modelos teóricos radica en varios aspectos: 1) su descripción se formula a partir de ecuaciones que son normalmente complejas y no lineales, éstas pueden involucrar ecuaciones diferenciales parciales o ecuaciones integrales; 2) errores en la formulación del modelo causadas por malas aproximaciones; 3) errores causados por simplificaciones del modelo, esto con el fin de hacerlo numéricamente más dócil; 4) errores en la estimación de parámetros causada por falta de información o una mala experimentación. Cuando no es posible superar las desventajas de los modelos teóricos es necesario implementar un método alternativo.

1.1. Antecedentes

El modelamiento empírico capta las características del sistema a partir de sus respuestas ante entradas de variables explicativas. La formulación de este modelo comienza con un apropiado análisis de los datos de entrada/salida con el objetivo de elegir o des-

cartar aquellas variables y respuestas que sean capaz de explicar el comportamiento del sistema. La principal problemática de los modelos empíricos es elegir una función capaz de proporcionar un resultado confiable [1].

Es en este punto, donde la implementación de técnicas de aprendizaje automático (machine learning) son capaces de resanar los problemas que tienen los modelos empíricos tradicionales. Esta rama de la inteligencia artificial tiene como objetivo desarrollar modelos que permiten generalizar comportamientos a partir de los datos recibidos. El aprendizaje automático nació en la década de los 60's como una idea ambiciosa por representar funciones del cerebro humano (neurona), con el objetivo de que las maquinas aprendieran a realizar tareas de la misma forma que lo hace el ser humano, a través de la experiencia. Es posible dicho aprendizaje debido a que se replican algunas facultades cognitivas del cerebro, con el fin de estructurar modelos capaces de generalizar la información que se les presente para realizar una tarea determinada. Estas técnicas han sido implementadas con gran éxito en procesos de ingeniería, medicina, matemáticas, comunicación, computación, etc. Las técnicas de aprendizaje automático se pueden clasificar en tres categorías según su aplicación.

1. Modelos de regresión, se utilizan para predecir el valor de una respuesta (valor continuo).

2. Modelos de clasificación, se utilizan para predecir el resultado de un atributo con valor discreto.

3. Modelos de agrupamiento, se utilizan para clasificar instancias de datos que tienen características similares, sin tener presente algún valor de salida (etiqueta).

Las técnicas de aprendizaje automático tienen la capacidad de encontrar relaciones y patrones entre la información que se les presenta (variables de entrada y salida) por si solos mediante procesos de aprendizaje o entrenamiento. Existen tres categorías de aprendizaje:

1. Aprendizaje supervisado, es aquel que para un conjunto de datos de entrada se conoce de antemano los datos correctos de salida, el proceso de entrenamiento se basa en comparar la salida del modelo y la respuesta deseada, generando un error, el error se utiliza para actualizar los parámetros del modelo de modo que el error llegue a un mínimo.

2. Aprendizaje no supervisado, consiste en que el modelo descubra por sí mismo a identificar características, correlaciones o categorías en los datos de entrada y sin tener presente datos correctos de salida con los cuales realizar alguna comparación.

Existen distintas técnicas de aprendizaje automático como redes neuronales artificiales, computo evolutivo, redes bayesianas, máquinas de soporte vectorial y métodos de kernel solo por mencionar los más populares. Desde inicios de los años 90's se han implementado modelos de aprendizaje automático en diferentes áreas de la ciencia [2–12]. Dando como resultado que hoy en día, el implementar este tipo de técnicas, sea una tarea común para resolver más problemas en diversas áreas. Su amplio uso se debe, principalmente, a tres factores: 1) el uso de procesadores más potentes que los del siglo pasado; 2) la facilidad de almacenar grandes cantidades de información, así como la creación de grandes bases de datos y; 3) el continuo desarrollo de mejores técnicas de aprendizaje.

Capítulo 2

Teoría general

2.1. Red neuronal artificial

Modelar el comportamiento de un sistema es de vital importancia para su estudio, análisis, control y optimización. El principal objetivo del modelamiento es encontrar una función capaz de predecir el comportamiento experimental del sistema real con una alta precisión. Cuando se elabora un modelo a partir de datos experimentales también se busca filtrar el ruido experimental de una manera eficiente. Dentro del campo de la inteligencia artificial existe el área de aprendizaje automático donde se engloban diferentes técnicas para modelar sistemas altamente complejos. Una de las principales técnicas del aprendizaje automático son las redes neuronales artificiales (RNAs) Figura 2.1.

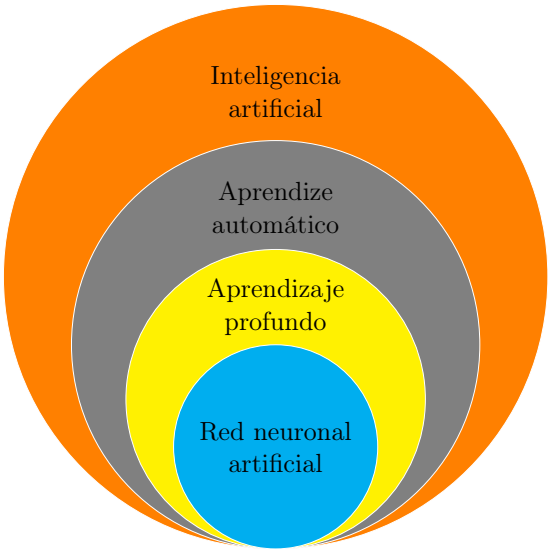
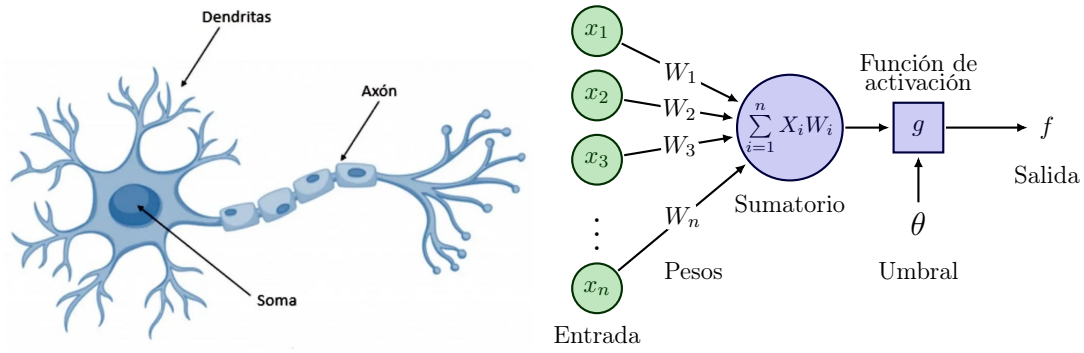


Figura 2.1: Esquema de las áreas implicadas dentro de la inteligencia artificial.

El primer modelo de una red neuronal artificial (RNA) fue desarrollado en 1943 por los científicos Warren S. McCulloch & Walter Pitts [13], con el objetivo de representar el funcionamiento neurofisiológico del cerebro, desarrollaron una representación lógica de la parte más fundamental de este, la neurona. El estudio permitió diseñar modelos abstractos de una neurona del cerebro humano, dando inicio a nuevas técnicas de aprendizaje automático y una mejor comprensión del funcionamiento neuronal. En la Figura 2.2 se puede observar la comparación entre una neurona biológica y la neurona modelada por McCulloch & Pitts (MCP).



(a) Red neuronal biológica.

(b) Modelo MCP.

Figura 2.2: Esquema comparativo de una red neuronal biológica y el modelo propuesto por McCulloch & Pitts.

La neurona biológica está formada por tres componentes principales: dendritas, cuerpo (soma) y axón. Las dendritas son las conexiones por donde se reciben las señales provenientes de otras neuronas. El cuerpo de la célula combina, integra y procesa la información que reciben las miles de dendritas y posteriormente emite señales de salida. El axón transporta la señal de salida a las terminales nerviosas, que luego la distribuye a un nuevo conjunto de neuronas, a través de conexiones sinápticas.

La analogía de una neurona biológica con el modelo MCP se describe en seguida, en la Figura 2.2b, las entradas $X_1, X_2, \ldots, X_n$ simulan las dendritas y la salida f la señal que viaja a través del axón. La función de activación g puede ser lineal o no-lineal, en la Figura 2.2b se presenta la version lineal, es decir, se genera un salida $f = 1$ si $\sum_{i=1}^{n} X_i W_i > \theta$, en caso contrario $f = 0$. Donde $W_1, W_2, \ldots, W_n$ se conocen como pesos y θ como el umbral (threshold).

La representación matemática del modelo MCP es la siguiente:

$$f = 1, \quad sí \sum_{i=1}^{n} X_i W_i > \theta \tag{2.1}$$

Para poder determinar los valores de los pesos sinápticos de esta red se implementó la regla delta desarrollada por B. Widrow and M. E. Hoff en 1962 [14]. El modelo

MCP fue implementado para realizar clasificaciones lineales y operaciones lógicas, como AND y OR. Posteriormente se detectó que este modelo no podía resolver problemas de clasificación no lineales como el XOR, lo cual causó un desinterés por casi dos décadas. Posteriormente en los años 60's el científico F. Rosenblatt [15] desarrolló el *perceptrón*, Figura 2.3, un modelo de red neuronal más general que el modelo MCP. La innovación del modelo perceptrón fue la introducción de un peso numérico (offset o bias, θ) y la introducción de una función de activación continua. Además, los pesos son adaptados mediante un algoritmo numérico (optimización).

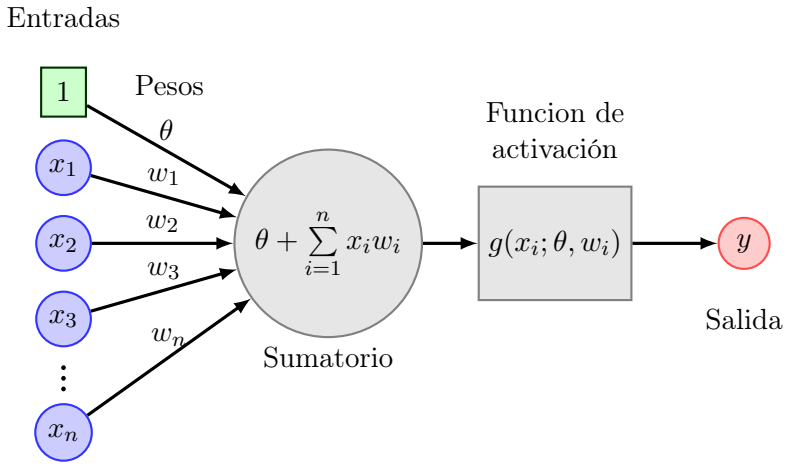


Figura 2.3: Representación de una neurona de tipo perceptrón.

Desafortunadamente, el modelo perceptrón presentaba las mismas desventajas que el modelo MCP, ya que no lograba realizar una clasificación no lineal. Para solucionar dicho problema se propuso incrementar el número de neuronas y posicionarlas en capas, obteniendo una red de perceptrón (red multicapa). Conforme la red incrementa en profundidad el modelo se vuelve altamente no lineal, por lo tanto, se requiere hacer un ajuste no lineal de sus parámetros, logrado mediante regresión no lineal y ejecutado por métodos de optimización, que en ese momento no se contaba con las técnicas y herramientas computacionales adecuadas para realizar dicho trabajo. Por lo tanto, las investigaciones de estos temas se detuvieron por más de una década.

No fue sino hasta el año de 1986 que el psicólogo David Rumelhart [16] redescubrió una forma para poder ajustar los parámetros de una red multicapa de una manera más sencilla. Dando a conocer el entrenamiento por retropropagación (backpropagation). Antes de entrar al tema del entrenamiento es necesario detallar la estructura de una red neuronal multicapa (perceptrón multicapa).

En la Figura 2.4 se ilustra, de forma general, una red neuronal tipica unida por interconexiones entre nodos (neuronas). Se tiene una capa de entrada, donde recibe la

información (variables de entrada, $x_i = y_i^1$) que posteriormente es introducida a la red para ser procesada. Después se encuentra una o varias capas ocultas, las cuales están conformadas por neuronas (perceptrones simples) interconectadas a la siguiente capa, asi sucesivamente hasta llegar a la capa de salida, donde se obtiene el resultado final del procesamiento (y_i^c). El termino *topologia de la red* determina la forma en como está estructurada una red neuronal multicapa, definiendo asi un número determinado de neuronas a la entrada y salida de la red, además, el número de capas ocultas y la cantidad de neuronas existentes en cada capa.

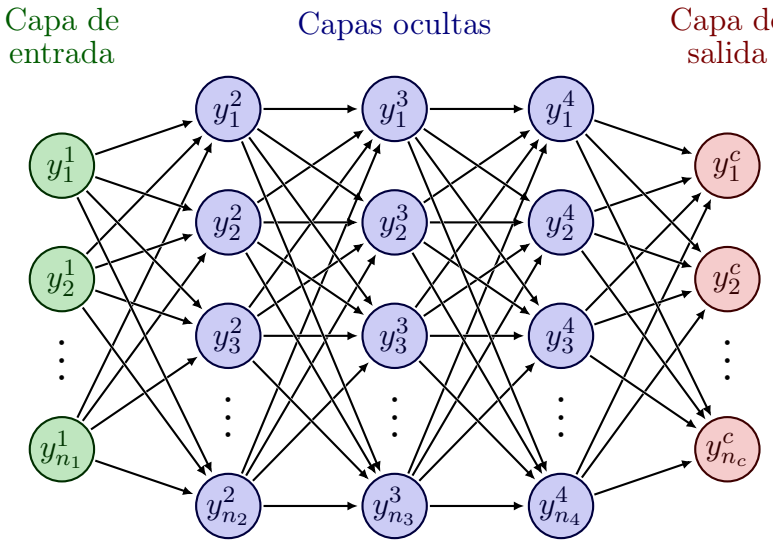


Figura 2.4: Esquema de una red neuronal multicapa de alimentación hacia adelante, Feedforward Neural Network (FFNN).

Dentro del campo de las RNAs la topología de una red se clasifica de dos formas: 1) las redes neuronales de una sola capa oculta (Single-hidden Layer Feedforward Neural Network, SLFN), son redes que se caracteriza por tener una sola capa oculta con muchas neuronas en ella; 2) las redes neuronales profundas (Deep Neural Network, DNN) son aquellas que presentan múltiples capas ocultas. Desde el año 2006 [17] las DNN han tenido un gran auge debido a su alta capacidad de abstracción, se han utilizan para el reconocimiento de patrones en grandes cantidades de datos (Big Data). Un ejemplo de ello son las plataformas de IBM Watson Developer Cloud, Amazon Machine Learning o Azure Machine Learning. En mayo del año 2020 se presentó un modelo llamado GPT-3 (Generative Pre-trained Transformer 3), el cual es un modelo de lenguaje autorregresivo que emplea aprendizaje profundo para producir textos que simulan la redacción humana, creado por OpenAI, un laboratorio de investigación de inteligencia artificial con sede en San Francisco, CA, EEUU. Dicho modelo tiene una capacidad de 175,000 millones de parámetros de aprendizaje automatizado, siendo es

un claro ejemplo de lo poderosos que pueden ser los modelos de aprendizaje automático, con el objetivo de simular y superar el comportamiento humano.

Las DNN se han hecho famosas por resolver problemas altamente complejos como la conducción automática de un automóvil, reconocimiento facial, reconocimiento de voz, entre otros. Por otro lado, se ha considerado que las SLFN no poseen la misma capacidad de abstracción que las DNN [18], por lo que su uso ha sido relegado a problemas de baja dimensión (pocas entradas y pocas salidas). A pesar de que las DNN se consideran modelos altamente eficientes existen algunas esventajas en su entrenamiento.

Las RNAs ya sean profundas o superficiales se consideran modelos matemáticos empíricos con parámetros (pesos sinápticos) que deben ser ajustados en base a datos experimentales del problema a modelar. De forma análoga para una persona seria la experiencia que acumula al desarrollar una actividad de forma continua. El procedimiento donde se realiza el ajuste de los parámetros de una RNA es conocido como *entrenamiento*, donde el objetivo es encontrar los valores apropiados de los pesos sinápticos entre cada neurona, de modo que la RNA aprenda a realizar una tarea específica. De esta manera, el entrenamiento de las RNAs se lleva a cabo formulando un problema de regresión no-lineal, donde se minimiza una función de error (entre la predicción de la RNA y el valor experimental), también conocido como aprendizaje supervisado.

2.2. Evaluación de una RNA

En una RNA la información se recibe en las neuronas de la capa de entrada, después es procesada en cada neurona, de las distintas capas ocultas y terminando en la capa de salida. El procesamiento de información, en cada neurona, se llevando a cabo mediante una combinación lineal con el vector de información de las neuronas de la capa precedente y el vector de pesos sinápticos (denotados por las flechas de conexión en la Figura 2.4) más un offset o bias y posteriormente se suaviza con una función de activación. De esta manera la salida de la neurona j en la capa i es:

$$y_j^{i+1} = g(\theta_j^i + \sum_{k=1}^{n_i} y_k^i \cdot W_{k,j}^i) \quad \forall \quad i = 1, \ldots, c-1 \quad j = 1, \ldots, n_{i+1} \tag{2.2}$$

donde $W_{k,j}^i$ es el peso que conecta la neurona k en la capa i con la neurona j en la capa $i+1$, θ_j^i es el offset aplicado a la neurona j en la capa $i+1$, y_j^{i+1} es la salida (respuesta) de la neurona j en la capa $i+1$, c es el número de capas, n_i es el número de neuronas en la capa i y $g(\bullet)$ es la función de activación.

La expresión mostrada en la Ecuación 2.2, es la manera tradicional en que se trata la teoría de las RNA. Sin embargo, la Ecuación 2.2 puede expresarse de manera matricial

si se agrupan todos los pesos y offsets entre dos capas consecutivas en una sola matriz de transformación de la capa i a la capa $i+1$, dando como resultado la expresión:

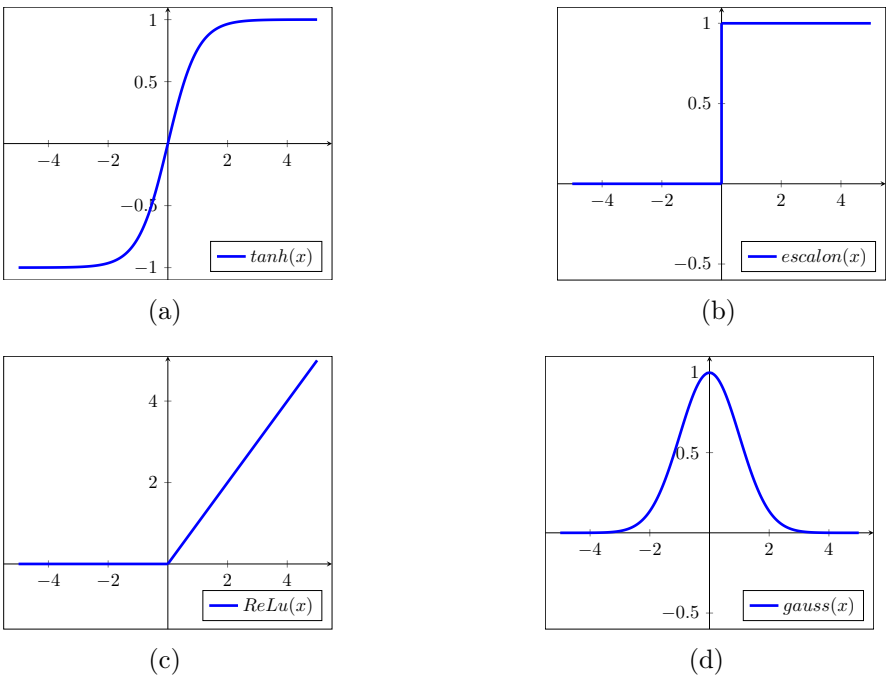
$$\underline{y}^{i+1} = g\left(\begin{bmatrix} 1 & \underline{y}^{i} \end{bmatrix} \cdot \underline{\underline{W}}^{i}\right) \in \mathbb{R}^{n_{i+1}} \tag{2.3}$$

donde

$$\underline{y}^{i} = [y_1, y_2, \ldots, y_{n_i}] \in \mathbb{R}^{n_i} \tag{2.4}$$

$$\underline{\underline{W}}^{i} = \begin{bmatrix} \theta_1^i & \theta_2^i & \cdots & \theta_{n_{i+1}}^i \\ W_{1,1}^i & W_{1,2}^i & \cdots & W_{1,n_{i+1}}^i \\ W_{2,1}^i & W_{2,2}^i & \cdots & W_{2,n_{i+1}}^i \\ \vdots & \vdots & \ddots & \vdots \\ W_{n_i,1}^i & W_{n_i,2}^i & \cdots & W_{n_i,n_{i+1}}^i \end{bmatrix} \tag{2.5}$$

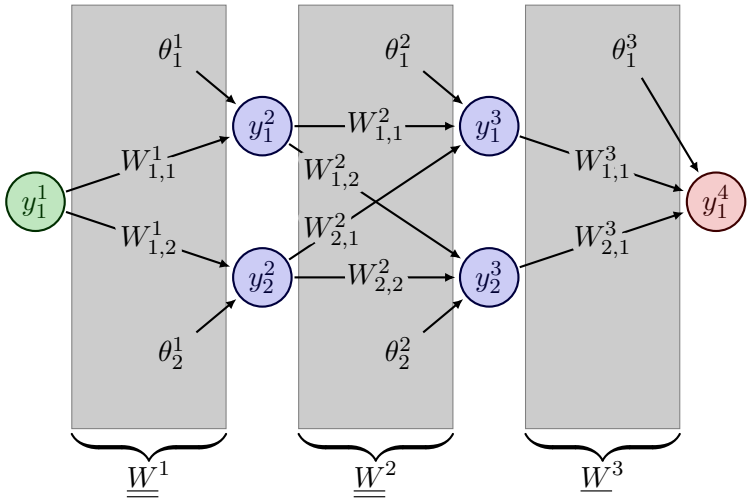
Se tiene que $\underline{y}^i$, para $i = 1$ es el vector de estímulos de entrada ($\underline{x} = [x_1, x_2, ...x_{n_1}]$) y, para $i = c$ es el vector de respuestas arrojadas por la RNA. La función de activación $g(\bullet)$ se aplica a cada uno de los elementos del vector resultante que hay en el argumento. La función de activación puede ser de diferentes tipos: lineal, escalón, sigmoidal, gaussiana o de algún otro tipo, mientras sea monotónica cresiente y acotada (Figura 2.5).



(a) (b) (c) (d)

Figura 2.5: Funciones de activación: a) función sigmoidal, b) función escalon, c) función ReLu, d) función gaussiana.

Conforme aumenta la profundidad de una RNA, tanto en capas ocultas como en el número de neuronas por capa, su modelo matemático se vuelve más complejo, en la Figura 2.6 se presenta un ejemplo de una red profunda cuya topología es: una neurona de entrada, dos neuronas por cada capa oculta y una neurona de salida, la cual se define como $\{1, 2, 2, 1\}$.



$$\underline{\underline{W}}^1 = \begin{bmatrix} \theta_1^1 & \theta_2^1 \\ W_{1,1}^1 & W_{1,2}^1 \end{bmatrix} \quad \underline{\underline{W}}^2 = \begin{bmatrix} \theta_1^2 & \theta_2^2 \\ W_{1,1}^2 & W_{1,2}^2 \\ W_{2,1}^2 & W_{2,2}^2 \end{bmatrix} \quad \underline{W}^3 = \begin{bmatrix} \theta_1^3 \\ W_{1,1}^3 \\ W_{2,1}^3 \end{bmatrix}$$

$$y_1^4 = g\left(\begin{bmatrix} 1 & g\left(\begin{bmatrix} 1 & g\left(\begin{bmatrix} 1 & y_1^1 \end{bmatrix} \cdot \begin{bmatrix} \theta_1^1 & \theta_2^1 \\ W_{1,1}^1 & W_{1,2}^1 \end{bmatrix}\right) \end{bmatrix} \cdot \begin{bmatrix} \theta_1^2 & \theta_2^2 \\ W_{1,1}^2 & W_{1,2}^2 \\ W_{2,1}^2 & W_{2,2}^2 \end{bmatrix}\right) \end{bmatrix} \cdot \begin{bmatrix} \theta_1^3 \\ W_{1,1}^3 \\ W_{2,1}^3 \end{bmatrix}\right)$$

$$y_1^4 = g\left(\begin{bmatrix} 1 & g\left(\begin{bmatrix} 1 & g\left(\begin{bmatrix} y_1^2 & y_2^2 \end{bmatrix}\right) \end{bmatrix} \cdot \begin{bmatrix} \theta_1^2 & \theta_2^2 \\ W_{1,1}^2 & W_{1,2}^2 \\ W_{2,1}^2 & W_{2,2}^2 \end{bmatrix}\right) \end{bmatrix} \cdot \begin{bmatrix} \theta_1^3 \\ W_{1,1}^3 \\ W_{2,1}^3 \end{bmatrix}\right)$$

$$y_1^4 = g\left(\begin{bmatrix} 1 & g\left(\begin{bmatrix} y_1^3 & y_2^3 \end{bmatrix}\right) \end{bmatrix} \cdot \begin{bmatrix} \theta_1^3 \\ W_{1,1}^3 \\ W_{2,1}^3 \end{bmatrix}\right)$$

Figura 2.6: Red neuronal artificial multicapa con topología $\{1, 2, 2, 1\}$.

Desarrollando el modelo matemático de la Figura 2.6 se obtienen la siguiente expresión:

$$
\begin{aligned}
y_1^2 &= g\left(\theta_1^1 + y_1^1 \cdot W_{1,1}^1\right) \\
y_2^2 &= g\left(\theta_2^1 + y_1^1 \cdot W_{1,2}^1\right) \\
y_1^3 &= g\left(\theta_1^2 + \left(y_1^2 \cdot W_{1,1}^2 + y_2^2 \cdot W_{2,1}^2\right)\right) \\
y_2^3 &= g\left(\theta_2^2 + \left(y_1^2 \cdot W_{1,2}^2 + y_2^2 \cdot W_{2,2}^2\right)\right) \\
y_1^4 &= g\left(\theta_1^3 + \left(y_1^3 \cdot W_{1,1}^3 + y_2^3 \cdot W_{2,1}^3\right)\right)
\end{aligned} \tag{2.6}
$$

Sustituyendo:

$$
\begin{aligned}
y_1^4 = g\Big(\theta_1^3 + \Big(g\Big(\theta_1^2 + \Big(g\left(\theta_1^1 + y_1^1 \cdot W_{1,1}^1\right) \cdot W_{1,1}^2 + g\left(\theta_2^1 + y_1^1 \cdot W_{1,2}^1\right) \cdot W_{2,1}^2\Big)\Big) \cdot W_{1,1}^3 + \\
g\Big(\theta_2^2 + \Big(g\left(\theta_1^1 + y_1^1 \cdot W_{1,1}^1\right) \cdot W_{1,2}^2 + g\left(\theta_2^1 + y_1^1 \cdot W_{1,2}^1\right) \cdot W_{2,2}^2\Big)\Big) \cdot W_{2,1}^3\Big)\Big)
\end{aligned} \tag{2.7}
$$

La Ecuación 2.7 también se puede representar de forma matricial:

$$
\begin{aligned}
\underline{y}^4 &= g\left(\begin{bmatrix} 1 & \underline{y}^3 \end{bmatrix} \cdot \underline{W}^3\right) \in \mathbb{R}^1 \\
\underline{y}^3 &= g\left(\begin{bmatrix} 1 & \underline{y}^2 \end{bmatrix} \cdot \underline{\underline{W}}^2\right) \in \mathbb{R}^2 \\
\underline{y}^2 &= g\left(\begin{bmatrix} 1 & \underline{y}^1 \end{bmatrix} \cdot \underline{\underline{W}}^1\right) \in \mathbb{R}^2
\end{aligned} \tag{2.8}
$$

y de manera compacta como:

$$
\underline{y}^4 = g\left(\begin{bmatrix} 1 & g\left(\begin{bmatrix} 1 & g\left(\begin{bmatrix} 1 & \underline{y}^1 \end{bmatrix} \cdot \underline{\underline{W}}^1\right) \end{bmatrix} \cdot \underline{\underline{W}}^2\right) \end{bmatrix} \cdot \underline{W}^3\right) \in \mathbb{R}^1 \tag{2.9}
$$

A pesar de que la red no es demasiado grande, se puede observar que, el modelo es altamente no lineal, debido a las funciones de activación en cada una de las neuronas, Ecuación 2.7. Por esta razón las reglas de aprendizaje implementadas por Rosenblatt no eran adecuadas para ajustar los parámetros de una red multicapa. Es aquí donde los estudios de Rumelhart tuvieron gran impacto al proponer un cambio de paradigma en la regla de aprendizaje delta generalizada.

Distintas reglas de aprendizaje se basan en minimizar la diferencia que existe entre t_i y y_i (valor real y valor calculado), por lo que el problema de ajustar los pesos sinápticos y offsets de una red neuronal se convierte en un problema de optimización. En los últimos años se ha propuesto el usa de algoritmos genéticos como una alternativa a la solución de dicho problema.

2.3. Entrenamiento tradicional

Para lograr que una RNA realice una tarea específica, se debe conocer su arquitectura (topología) y el valor de sus parámetros (pesos sinápticos y offsets). Parte de la

arquitectura queda determinada por el problema a modelar, de donde se determina el número de entradas (variables predictoras) y el número de salidas de la red (variables de respuesta). El número de capas ocultas y el número de neuronas en cada capa oculta se determina, hasta hoy en día, de manera tanto arbitraria y heurística, dependiendo mucho de la experiencia y de la habilidad del investigador.

En el contexto de modelado, las RNAs son modelos matemáticos empíricos con parámetros que deben ser ajustados en base a datos experimentales del problema a modelar. El entrenamiento de una RNA, es el proceso por el cual los parámetros de la red (modelo) son ajustados de manera apropiada, de modo que la RNA aprenda a realizar una tarea específica. De esta manera, el entrenamiento se lleva a cabo formulando un problema de regresión no-lineal, en el que se minimiza una función de error (entre el valor experimental y la predicción de la red).

Partiendo de un conjunto de m datos experimentales en los que se estudian n_1 variables predictoras (matriz de entradas$\underline{\underline{X}} \in \mathbb{R}^{m\times n_1}$), una variable de respuesta conocida (vector de targets $\underline{T} \in \mathbb{R}^{m\times 1}$) y tomando un modelo de SLFN de estructura $\{n_1,n_2,1\}$, con función de activación sigmoidal en la capa oculta y función lineal en la capa de salida, la función error a minimizar, puesta en forma matricial, es:

$$e(\underline{w}) = SSE = \sum_{i=1}^{m}(t_i - y_i^3)^2 = (\underline{T} - \underline{y}^3)^T \cdot (\underline{T} - \underline{y}^3) \tag{2.10}$$

donde $\underline{y}^3$ es el vector de predicciones de la RNA que depende de los parámetros ajustables $\underline{w}$ (vector que aglomera todos los parámetros ajustables, $\underline{w} = vec\{\underline{\underline{W}}^1, \underline{W}^2\} \in \mathbb{R}^p$). De acuerdo con la Ecuación 2.3, $\underline{y}^3$ se obtiene al evaluar la SLFN.

$$\underline{y}^3 = \left[\underline{1} \;\; \underline{\underline{y}}^2\right] \cdot \underline{W}^2 \in \mathbb{R}^{m\times 1} \tag{2.11}$$

$$\underline{\underline{y}}^2 = \tanh\left(\left[\underline{1} \;\; \underline{\underline{X}}\right] \cdot \underline{\underline{W}}^1\right) \in \mathbb{R}^{m\times n_2} \tag{2.12}$$

donde $\underline{\underline{W}}^1 \in \mathbb{R}^{(n_1+1)\times n_2}$, $\underline{W}^2 \in \mathbb{R}^{n_2+1}$ y $\underline{1} \in \mathbb{R}^m$, el vector $\underline{1}$ es un vector columna de unos.

En el entrenamiento tradicional, $\underline{w}$ se obtiene por minimización (optimización no restringida) de la Ecuación 2.10 a través de un procedimiento iterativo (p. ej. gradiente conjugado,Levenberg-Marquardt u otra técnica) en que el vector de parámetros, $\underline{w}$, se actualiza de manera recursiva de la siguiente forma:

$$\underline{w}_{(k+1)} = \underline{w}_k + \eta_k\,(w_k) \cdot \underline{B}_k\,(w_k, w_{k-1}) \tag{2.13}$$

donde k es la iteración actual, $\underline{B}_k$ indica la dirección en la cual se hace la búsqueda y η_k es un parámetro que indica el tamaño del paso (rapidez de entrenamiento en la

literatura de redes neuronales) en dicha dirección. Los elementos η_k y $\underline{B}_k$ dependen de $\underline{w}_k$. Además, el valor de η_k puede obtenerse mediante algún método de optimización unidimensional (como la sección dorada), o se puede fijar en un valor arbitrario. La mayoría de los métodos de optimización que utilizan una ecuación de recurrencia, solo difieren la manera de calcular el vector de direcciones $\underline{B}_k$. En el caso del gradiente conjugado $\underline{B}_k$ se calcula de la siguiente forma:

$$\underline{B}_k = -\nabla e_k(\underline{w}_k) + \beta_k(\underline{w}_k, \underline{w}_{k-1})\underline{B}_{k-1} \tag{2.14}$$

Donde ∇e_k es el vector de primeras derivadas de la función error, Ecuación 2.10, con respecto a los parámetros $\underline{w}$ en la iteración k; β_k es un parámetro que busca garantizar que el vector de dirección actuales $\underline{B}_k$ sea conjugado al conjunto de vectores de direcciones anteriores $\{\underline{B}_{k-1}, \underline{B}_{k-2}, ..., \underline{B}_1\}$. La mayoría de los métodos del gradiente conjugado que existen actualmente difieren en la forma en que calculan el parámetro β, la forma más utilizada es:

$$\beta_k = \frac{\nabla {e_k}^T \cdot \nabla e_k}{\nabla {e_{k-1}}^T \cdot \nabla e_{k-1}} = \frac{\sum_{i=1}^{p} \left(\frac{\partial e}{\partial w_i}\right)_k^2}{\sum_{i=1}^{p} \left(\frac{\partial e}{\partial w_i}\right)_{k-1}^2} \tag{2.15}$$

donde p es el número de parámetros contenidos en $\underline{w}$, ∇e_k y ∇e_{k-1} son vectores de los respectivos jacobianos de la función error $e(\underline{w})$ en la iteración k y $k-1$.

En la iteración inicial (1), no se conoce la información de la iteración 0, por lo tanto, B se calcula como $B_1 = -\nabla e_1$. El método se reinicia cada p iteraciones para evitar el deterioro por errores de aproximación, cada vez que se reinicia el método se le denomina *época*. El número de épocas queda definido por el investigador y es preciso indicar que no existe un número de epocas fijo, además, si se utilizan pocas épocas cabe la posibilidad de que el método no obtenga buenos resultados, por el contrario, al utilizar muchas épocas el proceso de optimización se extendería a tiempos muy largos y el coste computacional sería grande.

Determinar el vector $\nabla e(\underline{w})$ es difícil directamente desde la función error, Ecuación 2.10, para calcular dicho vector de derivadas se implementa la regla delta generalizada, la cual determina de forma analítica las derivadas de la función error con respecto de los pesos sinápticos y los offsets.

Todos los métodos de entrenamiento iterativo difieren en la manera en que se calculan sus elementos [19,20] y, presentan el inconveniente de que para diferentes valores iniciales de los parámetros de ajuste pueden dar, como resultado, diferentes soluciones (se llega a distintos mínimos locales). Además, por ser iterativo, requiere de tiempos muy extensos en el entrenamiento para alcanzar un mínimo, que no necesariamente es global, lo que resulta en una de las causas que desalientan el uso de las RNA. En la

siguiente sección se describe la forma en se obtiene desde la función error, y después se describe la técnica de validación cruzada que sirve como criterio de paro en el proceso de entrenamiento.

2.4. Backpropagation o Regla Delta

La mayoría de métodos de entrenamiento iterativo de RNA, que usan la Ecuación 2.13, suelen necesitar el cálculo del vector de derivadas $\nabla e(\underline{w})$ respecto de cada uno de los parámetros ajustables de la red, para lograrlo se pueden calcular las derivadas de forma numérica, pero para ello se requiere demasiado poder computacional. Rumelhart redescubrió en la tesis de Werbos una forma creativa de calcular la derivada de la función error de forma analítica [21, 22]. Dicho algoritmo, puesto de manera matricial y de acuerdo con las Ecuaciones 2.10 a 2.12 es como sigue:

Primero se evalúa la red y se calcula la delta de la capa de salida

$$\underline{\delta}^2 = \underline{T} - \underline{y}^3 \in \mathbb{R}^{m\times 1} \tag{2.16}$$

Después, se calculan las derivadas de la función error respecto de los parámetros que conectan hacia la última capa $\left(\underline{W}^2\right)$.

$$\frac{\partial e}{\partial \underline{W}^2} = -\left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \underline{\delta}^2 \in \mathbb{R}^{(n_2+1)\times 1} \tag{2.17}$$

Posteriormente, se calcula la delta la penúltima capa

$$\underline{\underline{\delta}}^1 = g'\left(\left[\left[\underline{1} \;\; \underline{\underline{X}}\right] \cdot \underline{\underline{W}}^1\right]\right) .* \left[\underline{\delta}^2 \cdot \left(\underline{W}^{(2)}\right)^T\right] \in \mathbb{R}^{m\times n_2} \tag{2.18}$$

Por último, se calculan las derivadas de la función error respecto de los parámetros que conectan hacia la penúltima capa $\left(\underline{\underline{W}}^1\right)$.

$$\frac{\partial e}{\partial \underline{\underline{W}}^1} = -\left[\underline{1} \;\; \underline{X}\right]^T \cdot \underline{\underline{\delta}}^1 \in \mathbb{R}^{(n_1+1)\times n_2} \tag{2.19}$$

donde $g'(x)$ es la derivada de la función de activación, el operador $.*$ representa multiplicación elemento a elemento entre dos matrices y la matriz $\underline{W}^{(2)} \in \mathbb{R}^{n_2\times 1}$ es una modificación de la matriz $\underline{W}^2$ a la que se le ha eliminado el primer renglón (vector renglón de los offsets $\underline{\theta}^2$).

De acuerdo con las Ecuaciones 2.16 a 2.19, el error (el delta de la capa de salida) se propaga hacia atrás, partiendo de la capa de salida hacia las neuronas que se encuentran en las capas ocultas, y es la razón del nombre del método. Si la red tiene más capas ocultas, se repiten los pasos de las Ecuaciones 2.18 y 2.19.

La regla delta generalizada es un algoritmo eficiente para calcular el vector de derivadas $\nabla e(\underline{w})$ que es necesario en algunos métodos de optimización, el cual trabaja sobre dos fases

El método por el cual los parámetros son ajustados se puede resumir de la siguiente forma. Cuando se introduce un conjunto de variables de entrada a la red ($\underline{x}_i$), éste "estimulo" se propaga desde la primera capa oculta a la segunda capa oculta y así sucesivamente, hasta llegar a la capa de salida ($\underline{y}^{c-1}$). La respuesta de la red se compara con la salida deseada ($\underline{t}_i$) y se calcula el error para cada dato del conjunto, Ecuación 2.10.

El error calculado se propaga hacia atrás, partiendo de la capa de salida hacia las neuronas que se encuentran en las capas ocultas, de tal forma que las neuronas de cada capa oculta solo reciben una fracción del error total. Posteriormente, se calcula la contribución relativa que haya aportado cada neurona al error, Ecuación 2.17 y 2.19, con el fin de actualizar el valor de los pesos sinápticos y offsets de la red en el proceso de optimización, 2.13, para hacer que la función error converja a un mínimo. Lo importante de este método es que conforme la red se entrene los pesos y offsets se organizan a sí mismas en la magnitud en que influyen al error. Logrando así que el proceso de entrenamiento trabaje sobre dos fases (propagación y adaptación). Es interesante indicar que cada neurona logra reconocer distintas características de los patrones de entrada/salida.

2.5. Validación cruzada

Los procesos iterativos por los cuales se efectúa el ajuste de un modelo, en base a datos experimentales, generalmente necesitan un criterio que les indique en qué momento detenerse. El criterio que se elija para detener el proceso de ajuste o entrenamiento debe tomar en cuenta algunas consideraciones, por ejemplo, la calidad de la salida del modelo, la cantidad en que se filtra el ruido experimental, la presencia de un sobreajuste de los datos, etc.

La validación cruzada es una técnica que indica el momento oportuno para detener el entrenamiento de una RNA, de modo que indica la capacidad de generalización de la RNA durante su proceso de entrenamiento.

La validación cruzada consiste en dividir los datos experimentales, datos de entrada (x_i) y datos de salida deseados (t_i), en dos conjuntos. Uno de ellos es el conjunto de entrenamiento y el otro es llamado conjunto de validación. El conjunto de entrenamiento es utilizado para ajustar los parámetros de la red durante el proceso de entrenamiento, a su vez, el conjunto de validación se utiliza para medir el rendimiento de la red mientras es entrenada de forma iterativa.

Cuando se construye una gráfica de la función error a partir del conjunto de validación respecto del número de iteraciones, Figura 2.7, se observa un comportamiento irregular, es decir, el error presenta muchos picos. La evolución del error de validación, aunque irregular, empieza a disminuir hasta alcanzar un mínimo para después aumentar. Por lo tanto, el error de validación funciona como criterio para determinar el momento adecuado en que debe detenerse el proceso de entrenamiento.

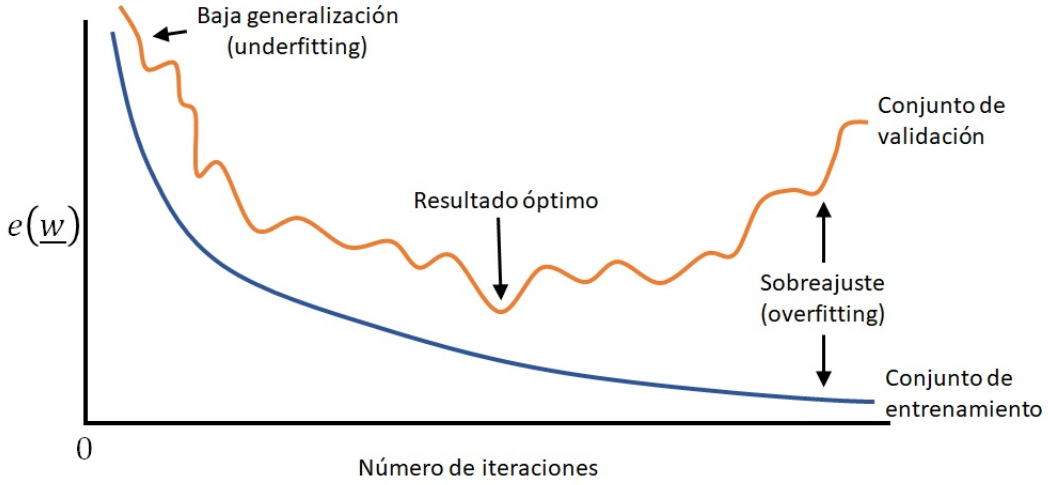


Figura 2.7: Comportamiento de los errores del conjunto de entrenamiento y validación durante el ajuste de parámetros de la RNA.

Durante el proceso de entrenamiento es posible identificar tres etapas: 1) la etapa de sobreajuste (overfitting), Figura 2.8*a*, es decir, que la red no tiene la capacidad de distinguir entre la señal y el ruido que contienen los datos de entrada, el entrenamiento debe detenerse antes de llegar a este punto; 2) la etapa de baja generalización(underfitting), Figura 2.8*c*, se presenta cuando los parámetros no se ajustan adecuadamente y la red no puede predecir el comportamiento real del sistema, generalmente ocurre cuando el procedo de entrenamiento es detenido antes de tiempo; 3) resultado óptimo, Figura 2.8*b*, es donde la red presenta la mejor capacidad de generalización.

2.6. Análisis de componentes principales

Al implementar una RNA para un problema específico es necesario definir algunos parámetros internos, por ejemplo: número de capas y neuronas ocultas (profundidad de la red), funciones de activación, pesos sinápticos iniciales, método de entrenamiento, tipo de validación, etc. Pero en pocas ocasiones se suele abarcar el tema del número óptimo de variables de entrada.

Al intentar modelar sistemas con bastantes variables de entrada se suele cuestionar

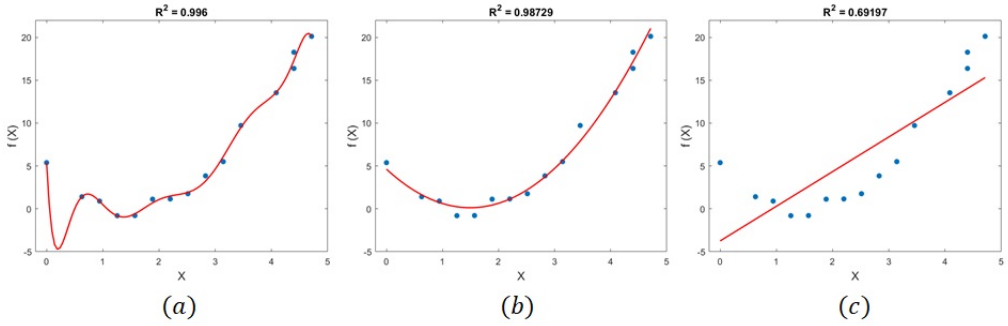


Figura 2.8: Distintos tipos de ajuste en el entrenamiento de una RNA: a) sobreajuste, b) ajuste óptimo, c) bajo-ajuste.

cuál de estas son realmente significativas a la respuesta o respuestas. Cuando el sistema posee bastantes variables de entrada existe la posibilidad de encontrar correlaciones entre ellas, indicando que los vectores columna de la matriz de entrada son *linealmente dependientes*. En algunos casos es posible utilizar diagramas de dispersión y encontrar dichas correlaciones entre las variables de entrada, pero esto no suele ser práctico en la mayoría de las veces. Una alternativa es utilizar el análisis de componentes principales (ACP) [23].

La técnica de ACP tiene como objetivo reducir el número de variables que describen el sistema estudiado, disminuyendo la dimensionalidad del problema y logrando que toda la información quede contenida en unas cuentas variables (componentes principales).

El objetivo del ACP es capturar la variación de un número n de variables originales, $\underline{x} \in \mathbb{R}^n$, en un número reducido de q componentes ($q \leq n$). Las variables de entrada sufren una transformación lineal dando como resultado un conjunto de nuevas variables (componentes principales, $\underline{\underline{Z}}$) que tienen la característica de ser ortogonales entre sí, lo que significa que las nuevas variables no presentan alguna correlación entre ellas. Los primeros q componentes principales (CPs) suelen explicar la mayor parte de variación del sistema.

Asumiendo m observaciones, se tiene una matriz de datos originales $\underline{\underline{X}} = [\underline{x}(1), \underline{x}(2), \ldots, \underline{x}(m)]^T \in \mathbb{R}^{m \times n}$, donde $\underline{\underline{X}}$ es una matriz de datos centrados, es decir, se le ha restado el promedio a cada columna. La transformación lineal de la matriz $\underline{\underline{X}}$ en componentes principales se define como

$$\underline{\underline{Z}} = \underline{\underline{X}} \cdot \underline{\underline{P}} \in \mathbb{R}^{m \times n} \tag{2.20}$$

Donde $\underline{\underline{P}} = [\underline{p}_1, \ldots, \underline{p}_n] \in \mathbb{R}^{n \times n}$ es una matriz ortonormal en la que cada vector columna $\left(\underline{p}_i\right)$ representa un eigenvector o vector de transformación que está asociado

con cada eigenvalor de la matriz de covarianza $\left(\underline{\underline{C}}\right)$ definida por:

$$\underline{\underline{C}} = \frac{1}{m-1}\underline{\underline{X}}^T \cdot \underline{\underline{X}} = \underline{\underline{P}} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{P}}^T \in \mathbb{R}^{n \times n} \tag{2.21}$$

Donde $\Lambda = diag\left(\lambda_1, ..., \lambda_n\right)$ es una matriz diagonal que representa los eigenvalores de $\underline{\underline{C}}$. Los eigenvalores están ordenados de manera decreciente de acuerdo con su varianza explicativa ($\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{q-1} \geq \lambda_q \geq \lambda_{q+1} \geq \ldots \geq \lambda_n \geq 0$). La tarea más difícil es encontrar y seleccionar el número óptimo de CPs denotado por q. Seleccionar un valor de q permite separar la matriz de eigenvectores en dos sub-matrices:

$$\underline{\underline{P}} = \left[\underline{\underline{\hat{P}}} \;\; \underline{\underline{\check{P}}}\right] \tag{2.22}$$

Donde $\underline{\underline{\hat{P}}} \in \mathbb{R}^{n \times q}$ y $\underline{\underline{\check{P}}} \in \mathbb{R}^{n \times (n-q)}$.

Los primeros q eigenvectores construyen el subespacio de componentes principales (SCP, $\underline{\underline{\hat{P}}}$) mientras que el resto de eigenvectores constituyen el complemento ortogonal conocido como subespacio de residuales (SR, $\underline{\underline{\check{P}}}$). La correcta selección de CPs es de vital importancia dentro del modelado de un sistema, es decir, si se eligen menos CPs de los necesarios se obtiene un mal modelo resultado de la incompleta representación del espacio de atributos, por el contrario, al seleccionar más CPs de los necesarios el modelo estará sobre parametrizado e incluye factores que no son relevantes hacia la respuesta. Para solucionar este problema se presentan distintas estrategias con el objetivo de determinar el valor óptimo de q, en algunos casos este valor puede ser obtenido a priori y posteriormente usarlo en la construcción de un modelo.

En la siguiente sección se describen y se detallan algunos criterios para identificar el número óptimo de CPs. En general, los criterios tratan de concentrar la mayor cantidad de información de un conjunto de variables explicativas.

2.6.1. Criterios de evaluación heurísticos

Estos criterios se basan en la experiencia y experimentación de algunos investigadores, por lo tanto, carecen de sustento teórico, pero suelen ser utilizados en gran medida en la práctica debido a su simplicidad y fácil entendimiento. En ocasiones, proporcionan soluciones subóptimas dando un primer acercamiento hacia modelos ACP factibles. Siendo criterios útiles si no se desea un modelo tan riguroso. Por lo general, se evalúan de forma visual y quedan a interpretación de cada investigador, llegando a ser de carácter subjetivo.

Criterio de Kaiser-Guttman

Es una técnica clásica donde se seleccionan aquellos CPs con eigenvalores mayores a la unidad. Fue sugerida por Guttman (1954) y popularizada por Kaiser en 1960. Se

basa en que la mayor información debe estar contenida por encima de la unidad [24,25].

$$\lambda^{KG} = \lambda_i > 1,\ \forall i = 1, \ldots, n \tag{2.23}$$

Broken-Stick (Palo roto)

Propuesto por Frontier en 1976 para identificar el número de CPs que se deben conservar, definiendo un criterio de corte donde la varianza total, es decir, la suma de los eigenvalores se divide aleatoriamente entre los distintos CPs, con lo que se espera que la distribución de los valores propios siga una distribución de palo roto. Los eigenvalores elegidos son aquellos que superen los eigenvalores críticos generados por el criterio de palo roto. La distribución se calcula fácilmente como [25]:

$$BS_k = \frac{1}{n} \sum_{i=k}^{n} \frac{1}{i},\ \forall k = 1, \ldots, n \tag{2.24}$$

Donde n es el número de variables y BS_k representa los eigenvalores críticos del criterio de palo roto. Si la proporción de la varianza explicativa del *k-ésimo* CP es mayor que BS_k entonces esos CPs se mantienen.

$$\lambda^{BS} = \lambda_k > BS_k,\ \forall k = 1, \ldots, n \tag{2.25}$$

Proporción de varianza total

Es un de los criterios más populares para determinar el número apropiado de CPs para reducir la dimensionalidad de un espacio de factores. El criterio permite retener aquellos CPs que contribuyen un porcentaje especifico de la varianza total acumulada, seleccionando un límite de forma subjetiva, comúnmente se elige el 95 % de la varianza total acumulada. El porcentaje específico de varianza se calcula de la siguiente forma:

$$Pv_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{n} \lambda_j} \times 100,\ \forall k = 1, \ldots, n \tag{2.26}$$

$$\lambda^{Pv} = \lambda_k > Pv_k,\ \forall k = 1, \ldots, n \tag{2.27}$$

Aunque el método es usado con regularidad no se recomienda su aplicación por ser poco fiable [25].

2.6.2. Criterios de evaluación estadísticos

En esta sección se presentan algunos criterios de selección de CPs que se basan en fundamentos estadísticos.

Prueba de esfericidad de Bartlett

La prueba evalúa si cada eigenvalor es significativamente diferente de los eigenvalores restantes [25]. La prueba intenta revelar el punto donde los CPs resumen una distribución esférica de puntos, la cual se calcula de la siguiente manera:

$$m\left[(n-k)\ln\left(\sum_{i=k+1}^{n}\frac{\lambda_i}{n-k}\right)-\sum_{i=k+1}^{n}\lambda_i\right]\sim\chi_k^2\quad\forall k=1,...,n \tag{2.28}$$

Donde n es el número de variables, k representa *k-ésimo* CP, λ_i es el eigenvalor del CP_i, m es el número de observaciones y los grados de libertar (df) son calculados de acuerdo a:

$$df=\frac{(n-k-1)\,(n-k+2)}{2} \tag{2.29}$$

El criterio establece elegir aquellos CPs que cumplan la condición:

$$\wp\left(\chi_k^2,df\right)>0.05 \tag{2.30}$$

Criterio de información de Akaike (AIC) y mínima descripción de longitud (MDL)

Los criterios AIC y MDL son frecuentemente utilizados en el procesamiento de señales, estos criterios trabajan bajo ciertas consideraciones: 1) trabajan sobre la matriz de covarianza en el ACP; 2) las variaciones del ruido en cada medición se consideran idénticas; 3) proporcionan un mínimo sobre el número de CPs.

AIC y MDL fueron propuestos por Akaike y Rissanen [26] para la selección de modelos. Partiendo de la Ecuación 2.21, se asume que la matriz de covarianza, $\underline{\underline{C}}$, es la contribución de la señal más el ruido experimental, representada como:

$$\underline{\underline{C}}=\bar{\underline{\underline{C}}}+\sigma^2\underline{\underline{I}} \tag{2.31}$$

Donde $\underline{\underline{C}}$ tiene un rango $q\leq n$ adecuado. Por lo tanto, los *k-ésimos* eigenvalores más pequeños ($k\leq n-q$) son iguales a σ^2 (desviación estándar):

$$\lambda_{q+1}=\lambda_{q+2}=\ldots=\lambda_n=\sigma^2 \tag{2.32}$$

Y donde $\lambda_1,\ldots,\lambda_q$ y $\underline{p}_1,\ldots,\underline{p}_q$ son los q más importantes eigenvalores y eigenvectores de $\underline{\underline{C}}$. A partir de aquí se define el vector:

$$\underline{J}=\left[\lambda_1,\ldots,\lambda_1,\sigma^2,\underline{p}_1^T,\ldots,\underline{p}_q^T\right]^T\in\mathbb{R}^{(n+1)q+1} \tag{2.33}$$

Como el vector de parámetros para el modelo de ACP. Originalmente estos criterios se desarrollaron para identificar los parámetros óptimos dentro de un modelo. Wax&Kailath [27] realizaron algunas modificaciones a los criterios AIC y MDL para aplicarlos en la identificación de CPs óptimos dentro de un modelo basado en ACP. Donde se considera que las m observaciones son independientes y tienen una distribución gaussiana de media cero, dando como resultado las siguientes ecuaciones.

$$AIC\,(k) = -2\ln\left(\frac{\prod_{i=k+1}^{n}\lambda_i^{\left(\frac{1}{n-k}\right)}}{\frac{1}{n-k}\sum_{i=k+1}^{n}\lambda_i}\right)^{(n-k)m} + 2M \tag{2.34}$$

$$MDL\,(k) = -2\ln\left(\frac{\prod_{i=k+1}^{n}\lambda_i^{\left(\frac{1}{n-k}\right)}}{\frac{1}{n-k}\sum_{i=k+1}^{n}\lambda_i}\right)^{(n-k)m} + M\ln(m) \tag{2.35}$$

$$\forall k = 1,\ldots,n$$

Donde M es el número de parámetros independientes en el vector de modelo $\underline{J}$ definido por:

$$M = \frac{k}{2}\,(n-k+1) \tag{2.36}$$

Las Ecuaciones 2.34 y 2.35, teóricamente, contienen un mínimo, ya que al incrementar el valor de $k \in [1,n]$ los primeros términos del lado derecho decrecen y los segundos términos aumentan. Wax&Kailath observaron que para bases de datos con bastantes observaciones (m), MDL suele ser más consistente mientras que AIC tiende a sobreestimar el número de CPs.

Varianza del error de reconstrucción (VRE)

Qin&Dunia [28] describieron un criterio para determinar el número de CPs, mediante la obtención de una función convexa. La VRE es separada en dos subespacios: 1) subespacio de componentes principales (SCP) y 2) subespacio de residuales (SR). La proporción del SCP incrementa con el número de CPs, mientras que SR decrece, resultando en un mínimo.

El criterio consiste en estimar una variable a partir de las demás en base a la redundancia de algunas variables, lo que genera una mejor reconstrucción de un subespacio libre de ruido e identifica variables redundantes.

Dado un vector de muestra $\underline{x} \in \mathbb{R}^n$ corrompido con ruido a lo largo de una dirección $\xi_i \in \mathbb{R}^n$. Por ejemplo, asumiendo $n = 5$ e $i = 2$, el vector de dirección es $\underline{\xi}_2^T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ y el vector de muestra se expresa como:

$$\underline{x} = \underline{\tilde{x}} + r\underline{\xi}_i \tag{2.37}$$

Donde $\underline{\tilde{x}}$ es un vector libre de ruido, r la magnitud del ruido y $\left\|\underline{\xi}_i\right\| = 1$. Para remover el ruido en la *i-ésima* variable se calcula un vector de estimación.

$$\underline{\hat{x}} = \underline{x} - \hat{r}_i\underline{\xi}_i \tag{2.38}$$

Donde $\hat{r}_i$ es la magnitud estimada del ruido r en la *i-ésima* variable. La estimación del ruido es óptima al minimizar el error cuadrado residual.

$$\hat{r}_i = \arg\min_r \|\underline{\tilde{x}} - \underline{\hat{x}}\| = \arg\min_r \left\|\underline{\underline{\check{P}}} \cdot \underline{\underline{\check{P}}}^T \cdot \underline{\hat{x}}\right\|^2 = \frac{\underline{\xi}_i^T \underline{\underline{\check{S}}}\, \underline{x}}{\underline{\xi}_i^T \underline{\underline{\check{S}}}\, \underline{\xi}_i} \tag{2.39}$$

Donde $\underline{\underline{\check{S}}} \in \mathbb{R}^{(n\times n)}$ es el espacio de proyección de $\underline{\underline{\check{P}}} \cdot \underline{\underline{\check{P}}}^T$, donde $\underline{\underline{\check{P}}}$ es el complemento ortogonal del subespacio de CPs. La diferencia $\underline{\tilde{x}} - \underline{\hat{x}}$ es conocida como la reconstrucción del error. Obsérvese que la reconstrucción del error puede ser representada como la proyección del vector de estimación en el subespacio residual y depende del número de CPs retenidos en el ACP. Qin&Dunia definen la varianza de la reconstrucción del error como:

$$\mathcal{U}_i \equiv var\left\{\underline{\xi}_i^T\left(\underline{x} - \underline{\hat{x}}\right)\right\} = var\left\{\hat{r}_i\right\} = \frac{\underline{\xi}_i^T \underline{\underline{\check{S}}}\, \underline{\underline{C}}\, \underline{\underline{\check{S}}}\, \underline{\xi}_i}{\left(\underline{\xi}_i^T \underline{\underline{\check{S}}}\, \underline{\xi}_i\right)^2} \tag{2.40}$$

La varianza no reconstruida es la varianza de la magnitud estimada del ruido. La reconstrucción del error utilizando cuidadosamente el número óptimo de CPs debería proporcionar un mínimo. Por lo tanto, las n variables deben considerarse simultáneamente de la siguiente forma:

$$VRE\left(k\right) = \sum_{i=k}^{n} \frac{\mathcal{U}_i}{\underline{\xi}_i^T \underline{\underline{C}}\, \underline{\xi}_i} = \sum_{i=k}^{n} \frac{\underline{\xi}_i^T \underline{\underline{\check{S}}}\, \underline{\underline{C}}\, \underline{\underline{\check{S}}}\, \underline{\xi}_i}{\left(\underline{\xi}_i^T \underline{\underline{\check{S}}}\, \underline{\xi}_i\right)^2 \underline{\xi}_i^T \underline{\underline{C}}\, \underline{\xi}_i} \tag{2.41}$$

Para evitar el problema de escala entre las variables individuales no reconstruidas, se deben ponderar cada una de ellas por su varianza individual $\left(\underline{\xi}_i^T \underline{\underline{C}}\, \underline{\xi}_i\right)$. Si los datos están escalados a varianzas unitarias, entonces $\underline{\xi}_i^T \underline{\underline{C}}\, \underline{\xi}_i = 1$. La selección óptima de los CPs utilizando el criterio de VRE se logra eligiendo aquellos CPs que minimicen la Ecuación 2.41, es decir:

$$\lambda^{VRE} = \arg\min_k VRE\left(k\right) \tag{2.42}$$

Además, debe tenerse en cuenta que para un ruido en particular en la dirección $\underline{\xi}_i$ es posible que $\mathcal{U}_i \geq var\left\{\underline{\xi}_i^T \underline{x}\right\}$, lo cual indica que el criterio da una predicción peor

que la media de los datos, resultando que la variable *i-ésima* tiene poca correlación con otras variables y debe ser eliminada del criterio.

2.6.3. Criterios de evaluación para la selección óptima de CPs

Los criterios anteriores tratan de establecer el número óptimo de CPs que deben ser utilizados para obtener una mejor calidad en la predicción mediante la eliminación de variables redundantes o poco significativas, Con tales criterios se trata de obtener una información a priori antes de ser introducidos a un modelo de regresión o clasificación. Una clara desventaja de estos criterios es que no contemplan los datos de las respuestas $(\underline{\underline{T}})$ para el análisis y selección de los CPs. Por tal razón es común encontrar casos de aplicación donde no utilizan dichos criterios.

Por lo general, en la mayoría de los casos de aplicación [29–34] se suele evaluar el modelo usando diferente número de CPs y de ellos se elige el que tiene mejor calidad de predicción. La manera de evaluar las diferentes alternativas puede ser de forma creciente, ir añadiendo de uno en uno los CPs al modelo o de forma decreciente, ir retirando CPs del modelo y posteriormente bajo una búsqueda de mallado elegir el modelo más adecuado según su calidad de predicción. El proceso de añadir o eliminar CPs puede ser demandante computacionalmente hablando si la base de datos que se desea modelar es considerablemente grande. En sentido contrario, cuando se tienen bases de datos pequeñas es posible realizar la búsqueda de mallado a un costo menor. Pero aún sigue siendo una forma muy habitual en la práctica. Además, se debe considerar el hecho de que cada vez que se cambia el número de CPs para ser modelados es necesario cambiar las características del modelo, ocasionando que se invierta más tiempo y trabajo.

2.7. Justificación

En la ingeniería química, así como en otras áreas de la ciencia e ingeniería, la implementación de modelos es de gran importancia, ya que permiten representar el comportamiento de un proceso y sus interrelaciones de causa y efecto. Siempre se busca que los modelos implementados representen fielmente al sistema de interés y que a su vez sean lo más reales posibles. La creación de modelos puede llevarse a cabo por alguna de las siguientes formas: modelos teóricos o modelos empíricos. Los modelos teóricos se plantean a través de la aplicación de leyes naturales y fenómenos que gobiernan al sistema. Por otro lado, un modelo empírico, se desarrolla a partir de la información obtenida experimentalmente con el fin de obtener una relación de causa-efecto por alguna función matemática.

La desventaja de los modelos teóricos radica en varios aspectos: 1) su descripción

se formula a partir de conocimientos científicos que resultan en ecuaciones que son generalmente complejas y no lineales; 2) errores en la formulación del modelo causadas por malas aproximaciones; 3) errores en las simplificaciones del modelo, al tratar de hacerlo numéricamente más dócil; 4) errores en la estimación de parámetros causada por falta de información o una mala experimentación.

Además, los métodos empíricos también presentan algunas desventajas, tales como: 1) requieren datos experimentales; 2) en primera instancia se desconoce la ecuación fundamental del fenómeno, por lo que generalmente se proponen ecuaciones simples, tales como polinomios; 3) los parámetros del modelo carecen de una representación física del sistema.

Cuando no es posible crear o desarrollar modelos teóricos, es momento de enfocarse en los modelos empíricos. El modelamiento empírico capta las características del sistema a partir de sus respuestas ante entradas de variables explicativas. La formulación del modelo comienza con un apropiado análisis de los datos de entrada/salida con el objetivo de elegir o descartar aquellas variables y respuestas que sean capaces de explicar el comportamiento del sistema. La principal problemática de los modelos empíricos es elegir una función capaz de proporcionar un resultado confiable.

Se ha demostrado que los modelos de redes neuronales artificiales (RNAs) tienen capacidades de aproximadores universales [35], por lo que los modelos de RNA son capaces de resanar los problemas que tienen los modelos empíricos. Las RNAs son una rama de la inteligencia artificial o sistemas inteligentes que pretende estudiar el reconocimiento de patrones y el aprendizaje realizado por las computadoras.

En la actualidad los modelos de RNAs son entrenados mediante técnicas iterativas (descenso del gradiente, método de Newton y cuasi-Newton, Levenberg-Marquardt, etc.), que resultan en una alta demanda computacional y en largos tiempos de entrenamiento. Los largos tiempos de entrenamiento de las RNAs hacen que se desalienten los estudios detallados en el propio funcionamiento de la RNA.

En los métodos de entrenamiento tradicionales, la minimización de la función error se realiza mediante una optimización no restringida (p.e. descenso del gradiente), donde es necesario el cálculo de las derivadas parciales con respecto de los parámetros de la red (regla delta), donde estas son propagadas desde la capa de salida hasta la capa de entrada. En la década de los 90's científicos como G. Cybenko y Hornik [35, 36] se centraron en estudiar las ventajas que presentan las SLFN. Demostrando teóricamente que una RNA de solo una capa oculta y con un número finito de neuronas tiene la capacidad de aproximar cualquier función no lineal, presentando una alta generalización.

Sin embargo, hay pocos estudios detallados sobre arquitecturas de RNAs, mejoramiento en los tiempos del entrenamiento, el impacto que puede tener las funciones de activación para tareas específicas, efectos de los pesos sinápticos con los que se inicia

el proceso de entrenamiento, utilización de técnicas de análisis de componentes principales y análisis de componentes principales no-lineales para mejorar la dependencia lineal de los datos que recibe la RNA, efecto del número de variables de entrada y su relación con los pesos sinápticos, etc.

2.8. Hipótesis

Es posible desarrollar una metodología para entrenar una RNA, de una forma rápida y con capacidad de generalización, es decir, que capture la señal y filtre el ruido de los datos experimentales. En base a los trabajos de Cybenko [35], Hornik [36], Schimitd [37] y Huang [38] se puede inferir que mediante un ajuste lineal es posible entrenar una RNA de forma rápida en comparación con las técnicas ya existentes. Además, existen varios factores que están relacionados con la calidad de la predicción de la RNA de forma directa. Por lo tanto, al entrenar de forma rápida una RNA, se puede analizar de manera detallada distintos factores implicados durante el entrenamiento de la red.

2.9. Objetivos

2.9.1. Objetivo general

El objetivo general es desarrollar una metodología clara, precisa y sencilla de cómo implementar una RNA de manera óptima, es decir, que el usuario no tenga que especificar parámetros adicionales, como: estructura de la red, número de épocas o iteraciones, rango de los pesos sinápticos, etc.

2.9.2. Objetivos particulares

1. Desarrollar un método de entrenamiento rápido.
2. Garantizar que la red entrenada tenga una alta capacidad de generalización, es decir, que disponga de la habilidad de distinguir entre señal y ruido proveniente de los datos a modelar.
3. Comparar el método propuesto contra métodos tradicionales.
4. Evaluar el efecto que tienen las neuronas ocultas hacia la calidad de la predicción.
5. Definir un criterio estadísticos para seleccionar la estructura óptima de la red.
6. Identificar los factores que llevan a la RNA hacia el underfitting y overfitting.
7. Identificar aquellos factores que optimicen la calidad de respuesta de la RNA.

8. Evaluar el efecto que tiene el rango de los pesos estocásticos hacia la respuesta de la RNA.

9. Extender la metodología propuesta a sistemas MIMO.

10. Mejorar la respuesta de la RNA mediante la técnica de Análisis de Componentes Principales (ACP).

11. Evaluar el efecto sinérgico del número de variables de entrada y el rango de los pesos estocásticos hacia la calidad de la respuesta de la RNA.

12. Definir una metodología clara y sencilla para entrenar una RNA mediante pesos estocásticos.

Capítulo 3

Metodología

3.1. Método propuesto (Stochastic One Step, SOS)

El objetivo principal de la tesis es presentar y desarrollar un método de entrenamiento rápido para RNAs con una sola capa oculta (SLFN). Se aplica la teoría para una RNA con una capa oculta ($c = 3$), función de activación sigmoidal, $g(x) = \tanh(x)$, en la capa oculta, función lineal (identidad) y una sola neurona, $n_3 = 1$, en la capa de salida. Sin embargo, el procedimiento puede ser aplicado a otras arquitecturas de RNA de alimentación hacia delante. En la Sección 4.7 se presenta la expansión de esta metodología aplicada a más de una neurona de salida, $n_3 > 1$.

Partiendo de un conjunto de m datos experimentales en los que se estudian n_1 variables predictoras (matriz de entradas $\underline{\underline{X}} \in \mathbb{R}^{m \times n_1}$), una variable de respuesta (vector de observaciones $\underline{T} \in \mathbb{R}^{m \times 1}$) y tomando un modelo de SLFN con una estructura $\{n_1, n_1, 1\}$. La función error a minimizar, tal como se presenta en la Ecuación 2.10 de la Sección 2.3, puesta en forma matricial, es

$$e(\underline{w}) = SSE = \sum_{i=1}^{m} \left(t_i - y_i^3\right)^2 = \left(\underline{T} - \underline{y}^3\right)^T \cdot \left(\underline{T} - \underline{y}^3\right) \tag{3.1}$$

donde $\underline{y}^3$ es el vector de predicciones de la SLFN que depende de los parámetros ajustables $\underline{w}$ (vector que aglomera todos los parámetros ajustables, $\underline{w} = vec\left\{\underline{\underline{W}}^1, \underline{W}^2\right\} \in \mathbb{R}^p$). De acuerdo con la Ecuación 2.3, $\underline{y}^3$ se obtiene al evaluar la SLFN de la siguiente manera

$$\underline{y}^3 = \left[\underline{1} \;\; \underline{\underline{y}}^2\right] \cdot \underline{W}^2 \in \mathbb{R}^{m \times 1} \tag{3.2}$$

$$\underline{\underline{y}}^2 = \tanh\left(\left[\underline{1} \;\; \underline{\underline{X}}\right] \cdot \underline{\underline{W}}^1\right) \in \mathbb{R}^{m \times n_2} \tag{3.3}$$

donde $\underline{\underline{W}}^1 \in \mathbb{R}^{(n_1+1) \times n_2}, \underline{W}^2 \in \mathbb{R}^{n_2+1}$ y $\underline{1} \in \mathbb{R}^m$.

Para el desarrollo de este trabajo se ha tomado las ideas propuesta en la literatura [37–39] en donde se desarrolla el concepto de una RNA con pesos aleatorios (Neural Network Random Weight, NNRW). Donde se propone ajustar solo la matriz de pesos de la capa de salida $\underline{\underline{W}}^2$, manteniendo constante la matriz de pesos de la capa de entrada $\underline{\underline{W}}^1$, donde se especifican los parámetros con valores de manera estocástica. En trabajos semejantes se ha propuesto entrenar los parámetros de la última capa de manera iterativa [40], que mejora la velocidad de entrenamiento, pero sigue siendo un método iterativo y tardado. Otros autores han propuesto ajustar los parámetros de la última capa haciendo uso de la matriz *pseudoinversa* [41]. El inconveniente en dichos trabajos es que no proporcionan una metodología general para determinar el número óptimo de neuronas en la capa oculta y en algunos casos el problema se soluciona a prueba y error, dando como resultado RNAs deficientes.

A diferencia de algunos trabajos previos, aquí se propone ajustar los pesos hacia la última capa en un solo paso, mediante regresión lineal, entre las salidas de la capa oculta y la capa de salida, lo cual puede hacerse con o sin la pseudoinversa y se dan las razones de hacerlo, además se plantea dar reglas claras de cómo hacerlo.

La salida de una SLFN es una relación lineal respecto de las salidas de la penúltima capa y, además los parámetros de la SLFN quedan agrupados en matrices de parámetros entre cada dos capas de neuronas consecutivas, Figura 3.1. Así que en una SLFN hay una matriz de parámetros entre la capa de entrada y la capa oculta $\underline{\underline{W}}^1$ y una matriz (un vector columna cuando se tiene una neurona de salida) de parámetros entre la capa oculta y la capa de salida $\underline{W}^2$. La estrategia presentada es, ajustar los parámetros $\underline{W}^2$ mediante regresión lineal, mientras que los parámetros $\underline{\underline{W}}^1$ se generan de forma estocástica dentro de un rango especifico, $\omega = [-1, +1]$.

Para ajustar los parámetros $\underline{W}^2$ se hace una minimización de la función error (Ecuación 3.1) igualando a cero la Ecuación 2.17.

$$-\left[\underline{1} \;\; \underline{\underline{y}}^2\right] \cdot \underline{\delta}^2 = \underline{0}$$

Sustituyendo la delta de salida (Ecuación 2.16) y despejando los parámetros optimizados, $\underline{\tilde{W}}^2$, se obtiene

$$\left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \left(\underline{T} - \left[\underline{1} \;\; \underline{\underline{y}}^2\right] \cdot \underline{\tilde{W}}^2\right) = \underline{0} \tag{3.4}$$

$$\left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1} \;\; \underline{\underline{y}}^2\right] \cdot \underline{\tilde{W}}^2 = \left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \underline{T} \tag{3.5}$$

$$\underline{\tilde{W}}^2 = \left(\left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1} \;\; \underline{\underline{y}}^2\right]\right)^{-1} \left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \underline{T} \tag{3.6}$$

que es la clave del método propuesto.

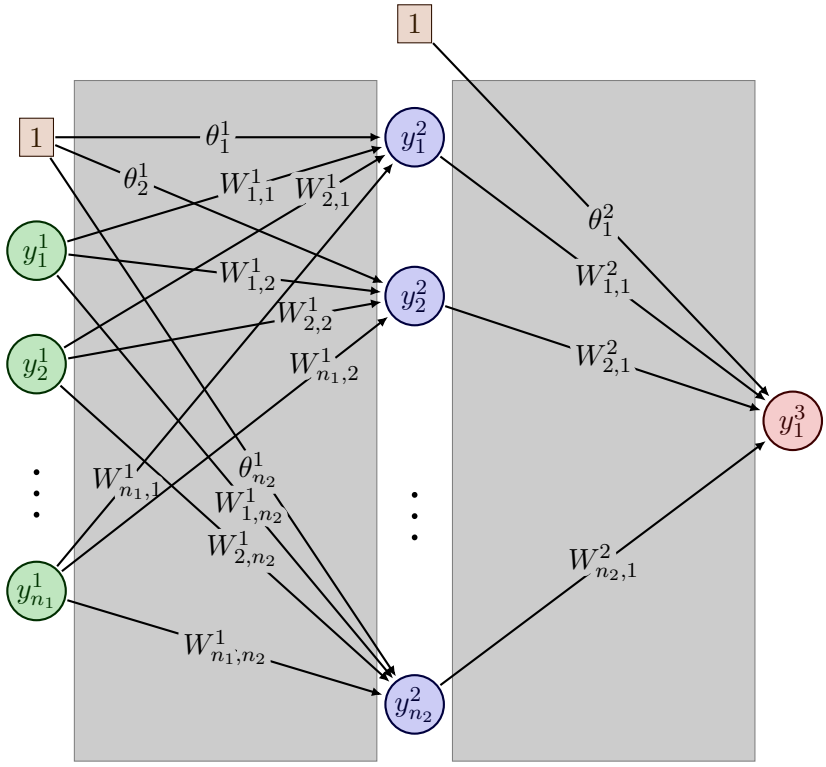


Figura 3.1: Red neuronal de una sola capa oculta de alimentación hacia adelante (SLFN).

Observaciones sobre la Ecuación 3.6.

1. La matriz $\left(\left[\underline{1}\ \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1}\ \underline{\underline{y}}^2\right]\right)^{-1}\left[\underline{1}\ \underline{\underline{y}}^2\right]^T$ es la inversa izquierda de $\left[\underline{1}\ \underline{\underline{y}}^2\right]$ y es conocida como *pseudoinversa*. Es necesario que dicha matriz exista para poder obtener $\underline{\tilde{W}}^2$.

2. La condición necesaria y suficiente para que la *pseudoinversa* exista es que exista la inversa de la matriz cuadrada $\left[\underline{1}\ \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1}\ \underline{\underline{y}}^2\right]$. Una manera de garantizarlo es que los vectores columna de $\left[\underline{1}\ \underline{\underline{y}}^2\right]$ sean linealmente independiente. En un problema práctico, se puede alejar de la dependencia lineal si el número de renglones es mucho mayor que el número de columnas en $\underline{\underline{y}}^2$, es decir, que el número de observaciones experimentales sea mucho mayor que el número de neuronas ocultas, $m \gg n_2$. Para entrenar una RNA, se requieren muchas observaciones experimentales. En caso extremo de que $\left[\underline{1}\ \underline{\underline{y}}^2\right]$ tenga vectores columna linealmente dependientes, la *inverza izquierda* se puede obtener por descomposición en valores singulares como *pseudoinversa*.

3. La matriz $\underline{\underline{y}}^2$ depende de los parámetros $\underline{\underline{W}}^1$, por lo tanto, también $\underline{\tilde{W}}^2$ depende

de $\underline{\underline{W}}^1$. Puesto que en cada ajuste de $\underline{\tilde{W}}^2$ se tiene que $\underline{\underline{W}}^1$ es constante, entonces el entrenamiento se hace en un solo paso y es instantáneo con la Ecuación 3.6.

4. El número de parámetros ajustables es $n_2 + 1$, por lo que al menos se requieren de $m = n_2 + 1$ datos experimentales. En la práctica, para que la matriz inversa izquierda se aleje de la singularidad, es necesario que $m \gg n_2 + 1$.

5. El cálculo de $\underline{\underline{\tilde{W}}}^2$, es prácticamente instantáneo comparado con el tiempo requerido en los métodos de entrenamiento iterativo. Así pues, aquí se propone llevar a cabo varios ajustes de $\underline{\tilde{W}}^2$, haciendo un barrido en el número de neuronas de la capa oculta, n_2, desde 1 hasta $m-1$. Conjeturamos que al aumentar n_2, entre 1 hasta $m-1$, el valor de $e(\underline{w})$ (Ecuación 3.1) disminuye hasta un mínimo en donde existe una estructura de RNA óptima con capacidad de generalización. De esta manera $\underline{\underline{\tilde{W}}}^1$ no se optimiza, más bien se asigna estocásticamente, en un rango ω, en cada cálculo de $\underline{\tilde{W}}^2$ y su número de columnas crece o disminuye con el número de neuronas ocultas. El método lo hemos llamado SOS (Stochastic One Step) debido a que la RNA se entrena en un solo paso con valores estocásticos de $\underline{\underline{W}}^1$ que se actualizan moviendo el número de neuronas ocultas.

6. En caso de querer optimizar por completo los parámetros hacia la capa oculta se requiere resolver, para $\underline{\underline{\tilde{W}}}^1$, la Ecuación 2.19 igualada a cero

$$-\left[\underline{1} \;\; \underline{\underline{X}}\right]^T \cdot \underline{\delta}^1 = \underline{0} \tag{3.7}$$

Pero esto resulta en un trabajo igual de desafiante que el proceso iterativo actual de entrenamiento de RNA.

3.2. Validación del método SOS

Durante el entrenamiento, las RNAs pueden llegar a ajustar el ruido experimental, confundiéndolo con la señal de interés. Es por lo que, en el entrenamiento de RNAs se debe llevar un control de la calidad del ajuste, el cual se hace con validación cruzada. La validación cruzada se realiza partiendo el conjunto de datos experimentales en dos, el conjunto de entrenamiento, que consta de m_e elementos y el conjunto de validación, que cuenta con m_v elementos. Donde el conjunto de entrenamiento se utiliza para el ajuste de los parámetros dentro de la red, mientras que el conjunto de validación se usa para verificar la calidad del ajuste. La proporción en la cual se deben dividir estos conjuntos es heurística, pero se recomienda una proporción del 80 % para el conjunto de entrenamiento y 20 % para el de validación.

En el presente trabajo, se propone seguir usando la validación cruzada para controlar la calidad del ajuste. A diferencia de la mayoría de los trabajos en el campo de las RNAs, aquí se usa el coeficiente de determinación para cada respuesta (Ecuación 3.8) como parámetro de control en el entrenamiento.

$$R_i^2 = \frac{SSR_i}{SST_i} = 1 - \frac{SSE_i}{SST_i} \tag{3.8}$$

donde SSE_i (Sum Square Error) depende de los parámetros de la RNA y se obtiene de la Ecuación 3.1, SST_i (Sum Square Total) es una constante que representa la variación total de los datos de respuesta

$$SST_i = \sum_{j=1}^{m_i} t_j^2 - \left(\sum\nolimits_{j=1}^{m_i} t_j\right)^2 / m_i, \quad i = \{e, v\} \tag{3.9}$$

Así pues, se tienen dos valores de R^2: uno calculado con el conjunto de entrenamiento $\left(R_e^2\right)$ y otro con el conjunto de validación $\left(R_v^2\right)$.

Adicionalmente, para controlar la inclusión excesiva de parámetros al modelo, tal como se hace en la regresión lineal, se usa el coeficiente de determinación ajustado, R_{adj}^2, que castiga la inclusión de parámetros

$$R_{adj,i}^2 = 1 - \tau \cdot \left(1 - R_i^2\right) \tag{3.10}$$

donde τ es el control que castiga la inclusión de parámetros,por cuestiones prácticas, aquí se toma como:

$$\tau = \frac{m_e - 1}{m_e - n_2 - 1} \tag{3.11}$$

Por lo tanto, es posible obtener cuatro criterios para evaluar el desempeño de la RNA durante su entrenamiento $R_e^2, R_v^2, R_{adj.e}^2, R_{adj,v}^2$.

Se escogen estos parámetros debido a que representan medidas estandarizadas en $[0, 1]$. El R^2 es una medida relacionada con SSE y el R_{adj}^2 es una medida relacionada con MSE(Mean Squared Error). Ahora la cuestión es ¿Cuál de todos estos parámetros sirve como criterio para elegir el número de neuronas en la capa oculta, n_2? La respuesta es dada en la sección de resultados.

3.3. Datos de estudio

La mejor manera para evaluar el desempeño de un método de entrenamiento es mediante datos simulados a los que se les añade una cantidad de ruido experimental (p.e. 30 % de su amplitud en la región de trabajo). Con el objetivo de tener control

sobre el estudio de la calidad de las aproximaciones, ya que se pueden comparar con los datos reales. Se trabaja con las tres superficies que se presentan en la Figura 3.2.

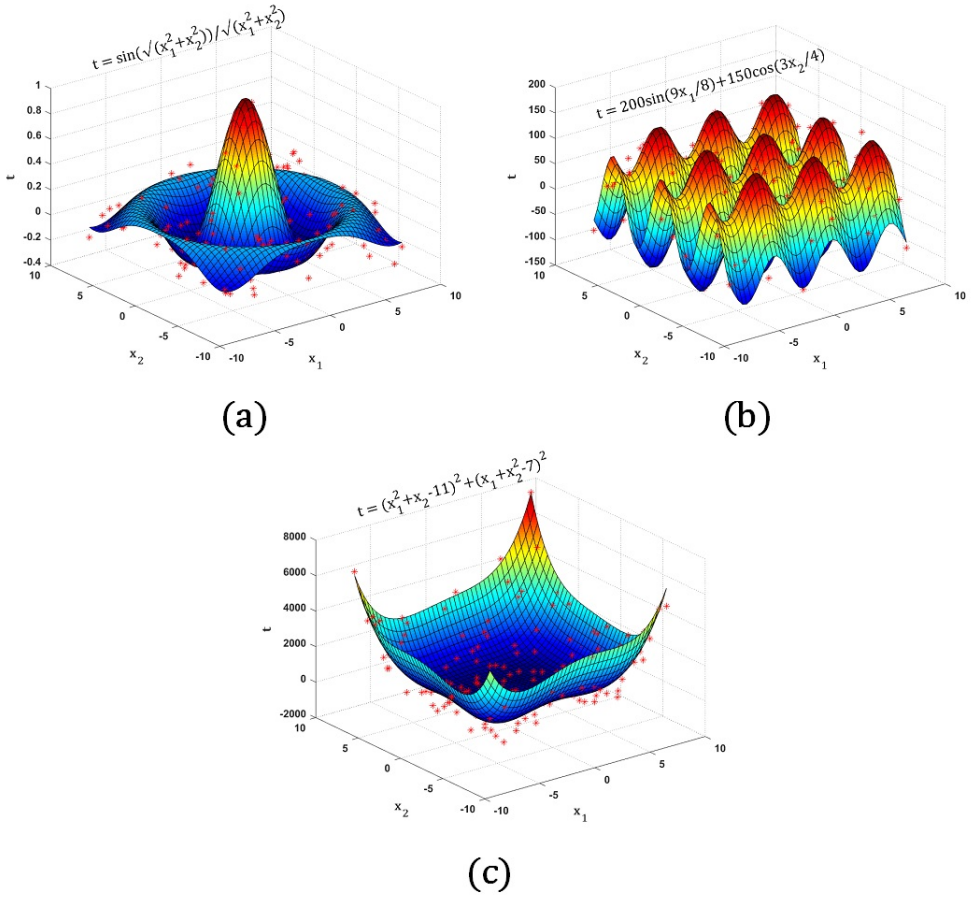


Figura 3.2: Superficies usadas para obtener los datos simulados en el entrenamiento de la red: a) superficie de la función gaussiana, b) superficie de la función conocida como cartón de huevo, c) superficie de la función de Himmelblau.

En la Figura 3.2 se presentan las tres superficies que se desean reconstruir a partir de datos simulados con el 30 % de ruido añadido (asteriscos en rojo).

Capítulo 4

Resultados y discusión

Los resultados presentados aquí tienen la intención de mostrar de forma experimental las ventajas que presenta el método propuesto SOS en comparación con distintos algoritmos de entrenamiento, además, se presentan distintas estrategias para mejor la eficiencia del método propuesto. En la Sección 4.1 se presenta la simulación de datos sintéticos, reconstrucción de superficies, mediante el método SOS y se comparan con distintos métodos de entrenamiento que se utilizan con regularidad para el entrenamiento de RNAs. En la Sección 4.2 se presenta el efecto que tiene el número de neuronas ocultas en una RNA, en la Sección 4.3 se determina el criterio para seleccionar la mejor topología de una RNA mediante el método SOS. Los efectos del underfitting y overfitting son presentados en la Sección 4.4. En la Sección 4.5 se expande el método SOS para múltiples respuestas (sistemas MIMO). Un estudio de ANOVA es presentado en la sección 4.6, con el fin de identificar aquellos factores que influyen en la eficiencia del método SOS. En la Sección 4.7 se implementa la técnica de reducción de dimensión, análisis de componentes principales, para el preprocesamiento de datos antes de ser introducida a una RNA, presentando ventajas y desventajas de su aplicación. Por último, en la Sección 4.8 se presenta el efecto que tiene el rango de los pesos sinápticos (ω) en el método SOS.

Todos los experimentos se llevaron a cabo en el entorno Matlab R2018a, que se ejecuta en un ordenador portátil Surface Pro 3 con procesador Intel(R) Core(TM) i5-4300U, CPU @ 2.50 GHz, 4 GB de RAM, y sistema operativo Windows 10 Pro a 64 bits.

4.1. Método SOS vs métodos iterativos

Para mostrar las ventajas del método SOS, se usaron datos sintéticos a partir de las superficies de la Figura 3.2. Los datos para el entrenamiento consistieron de 800

puntos muestreados aleatoriamente dentro de la misma región de variables independientes (x_1, x_2) mostrada en la Figura 3.2. Con los datos de forma la matriz de entradas $\underline{\underline{X}} \in \mathbb{R}^{800\times 2}$ necesaria en las Ecuaciones 3.3 y 3.6. El valor de la variable de respuesta consistió en el valor de la función, $t\,(x_1, x_2)$, más un ruido experimental de un 30 % respecto de la amplitud de la superficie correspondiente. Los valores de la respuesta simulada y con ruido forman el vector de target $\underline{T} \in \mathbb{R}^{800\times 1}$ usado en la Ecuación 3.6. El conjunto de los 800 puntos experimentales fue partido en dos grupos: uno de entrenamiento y el otro de validación. El conjunto de entrenamiento consistió de $m_e = 640$ puntos experimentales (80 %) y el conjunto de validación consistió de $m_v = 160$ puntos experimentales. Para el entrenamiento de la SLFN, todos los datos experimentales, tanto de variables de entrada como de respuesta, fueron escalados en el rango $[-1, +1]$.

Para demostrar la rapidez, precisión y capacidad de generalización del método SOS, se realizó una comparación contra distintos métodos de entrenamiento iterativos incorporado en el ambiente de Matlab®: descenso del gradiente con momento (traingdm), Levenberg-Marquardt (trainlm), máquina de soporte vectorial (Support Vector Machine, SVM, fitrsvm) y máquina de aprendizaje extremo (Extreme Learning Machine, ELM). Cabe mencionar que SVM es un algoritmo de aprendizaje automático que consiste en el uso de funciones de kernel y mediante procesos de optimización (proceso iterativo) se calculan los parámetros internos del modelo, así como su estructura interna. ELM también es un modelo de aprendizaje automático que tiene varias semejanzas con el método SOS, inicialmente hace uso de la pseudoinversa para calcular $\left(\left[\underline{1}\ \ \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1}\ \ \underline{\underline{y}}^2\right]\right)^{-1}$ en la Ecuación 3.6. Además, para utilizar SVM no es necesario especificar alguna topología interna. En todos los casos, excepto para SVM, se eligió una SLFN con estructura $\{2, 55, 1\}$ y los parámetros ajustables para SOS fueron inicializados aleatoriamente en el rango $[-1, +1]$.

En la Tabla 4.1 se presenta un comparativo de los tiempos de entrenamiento, error cuadrático medio (MSE) y el coeficiente de determinación $\left(R^2\right)$. Los resultados reportados son el promedio de diez corridas por cada método y función. Se observa que el tiempo de procesamiento es de al menos unas 80 veces más corto para el entrenamiento SOS en comparación con trainml (Levenberg-Marquardt), 70 veces más corto en comparación con traingdm (descenso del gradiente con momento), 13 veces más rápido que SVM y 3 veces más para ELM. En general, los valores de MSE reportados por SOS son menores en comparación con los métodos iterativos, indicando que el método SOS genera una mejor aproximación a los datos, salvo en el caso de la función Cartón de huevo el método trainlm reporta un valor menor de MSE, pero su tiempo de entrenamiento es 115 veces mayor que el de SOS. Es posible apreciar que los coeficientes de determinación del método propuesto son mejores que los reportados por los otros

Tabla 4.1: Parámetros del desempeño de las RNAs en la reconstrucción de las superficies.

Función	Tiempo (seg)	MSE	R^2
SOS			
Gaussiana	**0.0056**	**0.0021**	**0.947**
Cartón de huevo	**0.0065**	611.92	0.830
Himmelblau	**0.0069**	$\mathbf{4.285\times10^4}$	**0.960**
Matlab (trainlm)			
Gaussiana	0.4944	0.0036	0.913
Cartón de huevo	0.7510	**172.550**	**0.951**
Himmelblau	0.4438	1.236×10^5	0.919
Matlab (traingdm)			
Gaussiana	0.7023	1.5839	-38.1
Cartón de huevo	0.3289	4.974×10^5	-136.3
Himmelblau	0.2291	2.951×10^8	-191.1
Matlab (fitrsvm)			
Gaussiana	0.0697	0.0041	0.905
Cartón de huevo	0.1288	219.849	0.938
Himmelblau	0.0714	2.551×10^5	0.840
ELM			
Gaussiana	0.0169	0.0028	0.931
Cartón de huevo	0.0304	593.957	0.832
Himmelblau	0.0262	5.986×10^4	0.960

métodos. Cabe señalar que los códigos usados por Matlab son altamente optimizados, mientras que los códigos usados en el método SOS fueron generados por nosotros mismos y aún no han sido optimizados. En base a los resultados presentados, se puede concluir que el método SOS, comparado con los métodos iterativos, es prácticamente instantáneo. Además, arroja resultados de calidad muy aceptable, logrando en un solo paso resultados con capacidades altas de generalización.

A continuación, en las Figuras 4.1, 4.2 y 4.3 se muestran la reconstrucción de las superficies, donde se observa la calidad de reconstrucción de cada superficie proporcionada por cada método.

Las superficies generadas por el método SOS (Figura 4.1*a*, Figura 4.2*a* y 4.3*a*) presentan una mayor suavidad en comparación con las superficies que arroja los métodos iterativos de Matlab (trainlm, traingdm y fitrsvm). En las gráficas se muestra que el método de entrenamiento SOS proporciona resultados que pueden competir, en calidad de la aproximación, con los métodos tradicionales. No obstante, las superficies generadas con el método ELM compiten con las del método SOS, en cuestión de suavidad, lo cual puede atribuirse a que ambos métodos tienen semejanza en su

entrenamiento (regresión lineal). Observe que los resultados del método SOS no son un golpe de suerte, ya que, en la reconstrucción de las superficies, se ha logrado filtrar adecuadamente el error experimental. La evidencia de lo anterior se muestra en las gráficas (esquina inferior derecha) de la respuesta predicha por la red contra la respuesta experimental, se observa que el ruido ha sido filtrado adecuadamente. Además, las superficies en las Figuras 4.1*b*, 4.2*b* y 4.3*b*, reportan una mala predicción dando como resultado superficies muy deformadas en comparación con los otros métodos. Los métodos de Levenberg-Marquardt y SVM generan resultados más agradables, pero sin ser los mejores.

4.2. Efecto de neuronas ocultas en SOS

Teniendo un método de entrenamiento prácticamente instantáneo, ahora se pueden hacer estudios más detallados en los que se varíe el número de neuronas en la capa oculta y se observe la calidad del desempeño de la red a través del coeficiente de determinación $\left(R^2\right)$. El número de neuronas ocultas puede variarse, desde 1 hasta $m_e - 1$, ya sea por barrido secuencial o barrido esporádico. Realizar un estudio así era difícil de llevar a cabo con el entrenamiento iterativo debido a su lentitud. En la Figura 4.4 se muestra la evolución de los distintos coeficientes de determinación $\left(R^2_e, R^2_v, R^2_{adj.e}, R^2_{adj,v}\right)$ en función del número de neuronas en la capa oculta. Se observa la evolución solo hasta 200 neuronas, ya que, con un número muy grande de neuronas, la calidad de la reconstrucción comienza a deteriorarse, tal como se aprecia en el lado derecho de cada gráfica, Figura 4.4. De acuerdo a lo mostrado en la Figura 4.4, los coeficientes de determinación sufren variaciones al aumentar el número de neuronas ocultas, pero de manera general, se observa lo siguiente:

1. El coeficiente de determinación del conjunto de entrenamiento, R^2_e, (curva de puntos de color azul) siempre va en aumento y se acerca a 1 cuando el número de neuronas ocultas tiende a $m_e - 1$. Por lo tanto, el R^2_e no proporciona un criterio válido para obtener la estructura óptima de la RNA.

2. El coeficiente de determinación ajustado del conjunto de entrenamiento, $R^2_{adj.e}$, (curva de puntos de color negro) comienza aumentando rápidamente hasta alcanzar un máximo y después de ahí disminuye suavemente, pero al final disminuye rápidamente. Para los casos mostrado, los valores máximos son alcanzados en 84, 98 y 50 neuronas ocultas respectivamente, $N^*_{adj,e}$. El parámetro $R^2_{adj.e}$ podría proporcionar un criterio válido para obtener la estructura óptima de la RNA, pero se descarta debido a que son valores relativamente altos que se encuentran en la zona de sobreajuste (overfitting).

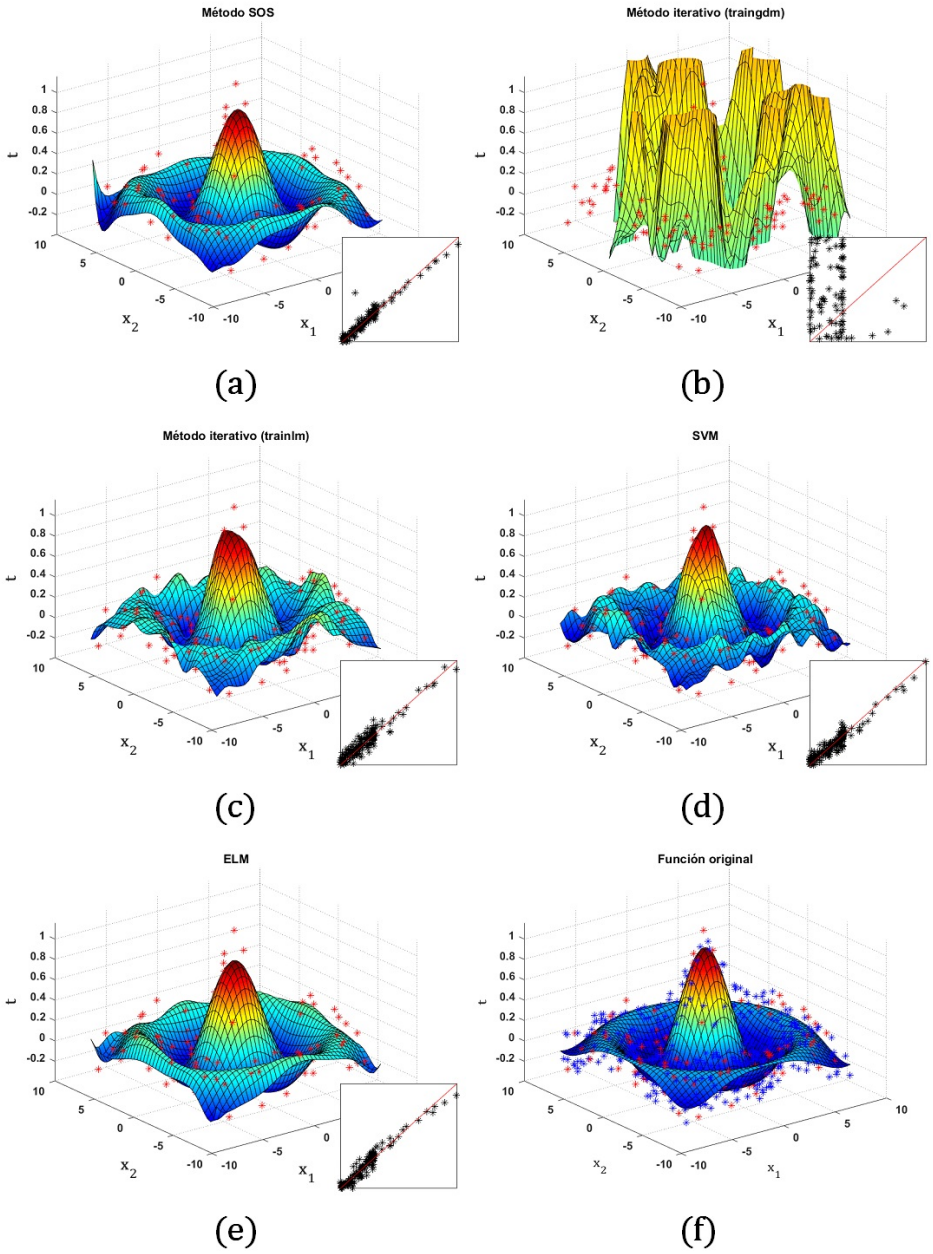


Figura 4.1: Reconstrucción de superficies por los distintos métodos. Superficie de la función Gaussiana obtenida con: (a) SOS, (b) descenso del gradiente con momento, (c) Levenberg-Marquardt, (d) Support Vector Machine, (e) Extreme Learning Machine y en (f) se presenta la función original donde (*) en azul son los datos del conjunto de entrenamiento y (*) en rojo corresponden a los datos de validación. Cada SLFN tiene una topología {2,55,1}, excepto SVM.

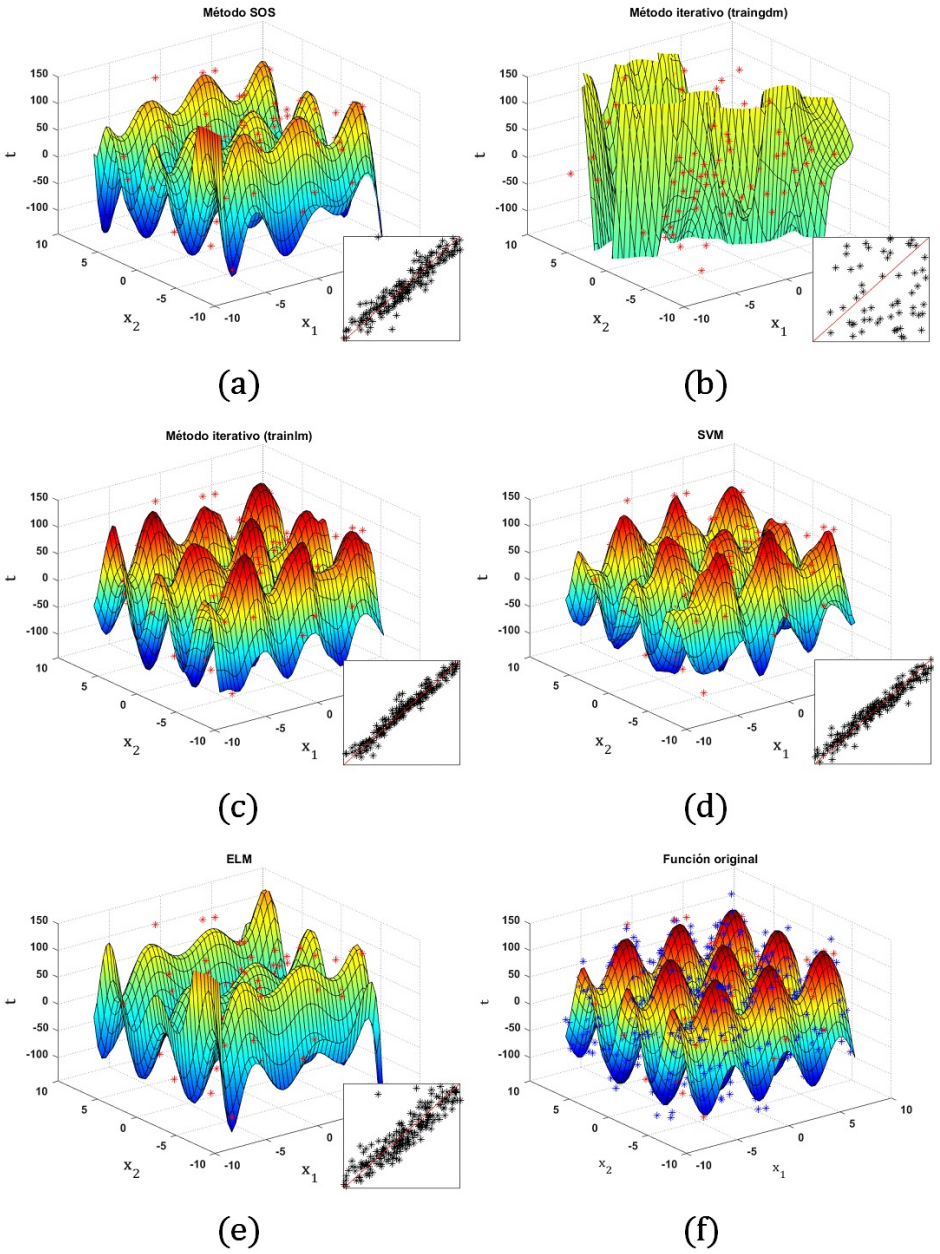


Figura 4.2: Reconstrucción de superficies por los distintos métodos. Superficie de la función Cartón de huevo obtenida con: (a) SOS, (b) descenso del gradiente con momento, (c) Levenberg-Marquardt, (d) Support Vector Machine, (e) Extreme Learning Machine y en (f) se presenta la función original donde (*) en azul son los datos del conjunto de entrenamiento y (*) en rojo corresponden a los datos de validación. Cada SLFN tiene una topología {2,55,1}, excepto SVM.

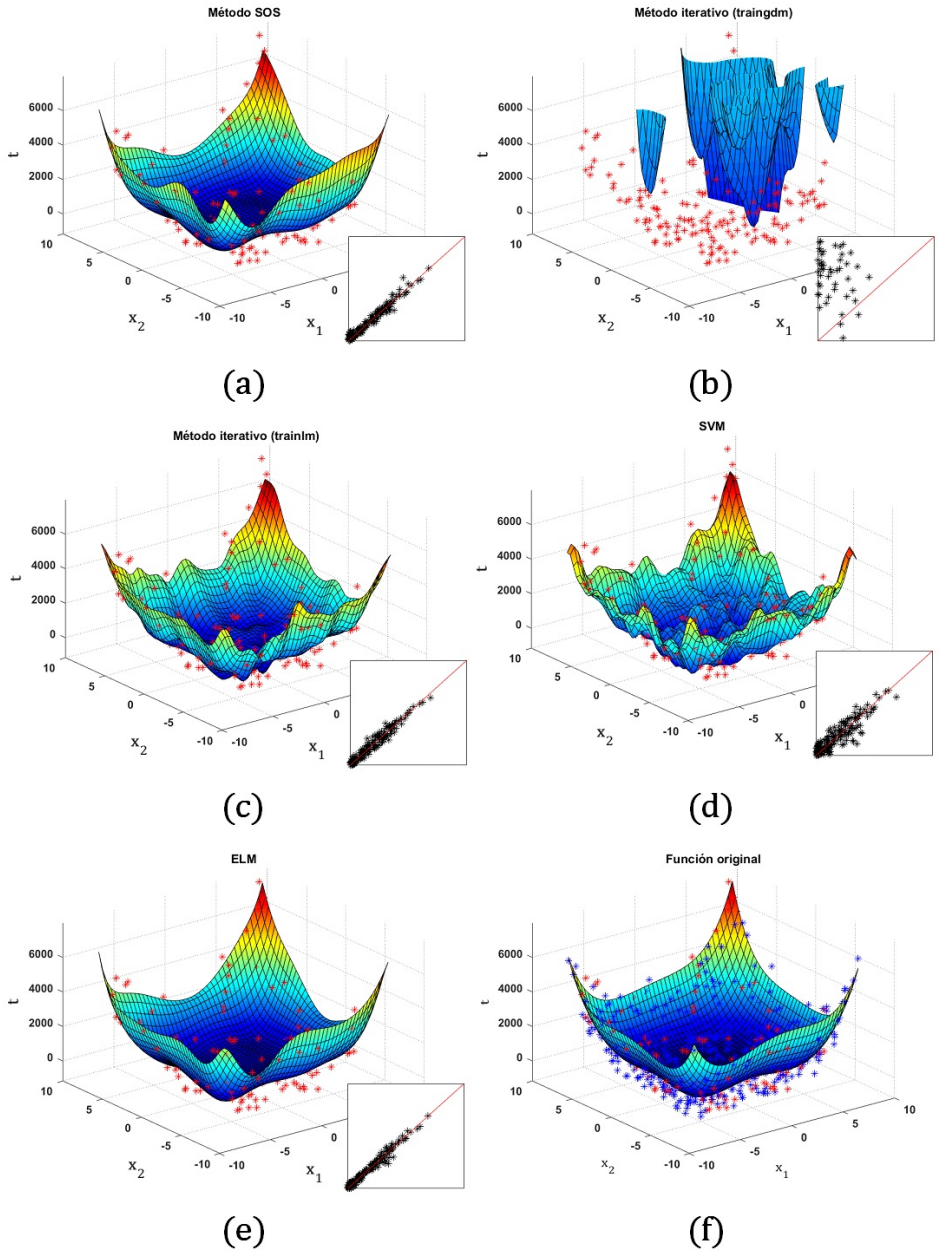


Figura 4.3: Reconstrucción de superficies por los distintos métodos. Superficie de la función Himmelblau obtenida con: (a) SOS, (b) descenso del gradiente con momento, (c) Levenberg-Marquardt, (d) Support Vector Machine, (e) Extreme Learning Machine y en (f) se presenta la función original donde (*) en azul son los datos del conjunto de entrenamiento y (*) en rojo corresponden a los datos de validación. Cada SLFN tiene una topología {2,55,1}, excepto SVM.

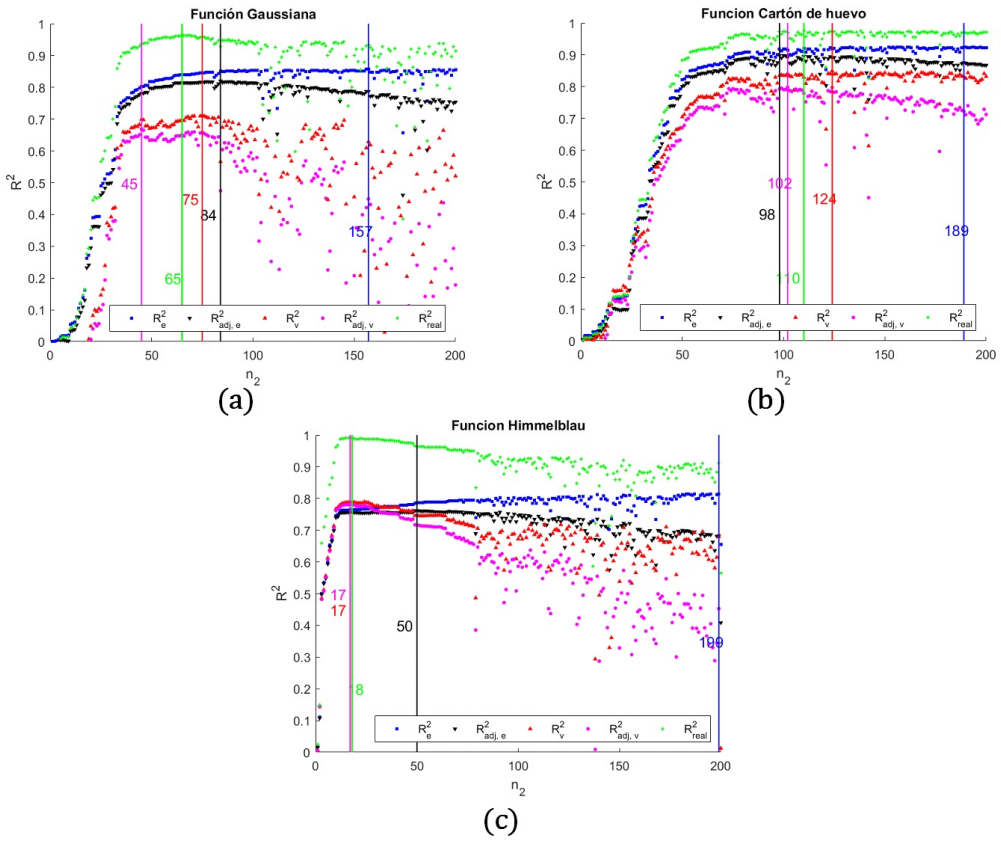


Figura 4.4: Comportamiento del coeficiente de determinación R^2 respecto del número de neuronas en la capa oculta (n_2) en una SLFN. Referente a: a) superficie Gaussiana, c) superficie de Cartón de huevo y c) superficie de Himmelblau.

3. El coeficiente de determinación del conjunto de validación, R^2_v, (curva de puntos de color rojo) comienza aumentando rápidamente hasta alcanzar un máximo y después de ahí disminuye suavemente, pero al final disminuye rápidamente con un comportamiento un tanto errático. Para los casos mostrado, sus valores máximos son alcanzados en 75, 124 y 17 neuronas ocultas respectivamente, N^*_v. El parámetro R^2_v podría proporcionar un criterio válido para obtener la estructura óptima de la RNA.

4. El coeficiente de determinación ajustado del conjunto de validación, $R^2_{adj,v}$, (curva de puntos de color magenta) comienza aumentando rápidamente hasta alcanzar un máximo y después de ahí disminuye suavemente, pero al final disminuye rápidamente de manera un tanto errática. Para los casos mostrado, sus valores máximos son alcanzados en 45, 102 y 17 neuronas ocultas respectivamente, $N^*_{adj,v}$. El parámetro $R^2_{adj,v}$ podría proporcionar un criterio válido para obtener la estructura óptima de la RNA.

5. El coeficiente de determinación de los datos reales, R^2_{real}, (curva de puntos de color verde), es un parámetro adicional que se ha usado aquí para comparar la calidad de la predicción con los datos reales. Es obtenido de acuerdo con la Ecuación 3.8, solo que el valor de la suma de cuadrados del error, SSE, es calculado entre las diferencias del valor real sin ruido y el valor predicho por la RNA. Cabe mencionar que, en un caso práctico, el valor real sin ruido es desconocido. Sin embargo, en este estudio teórico, donde se usan datos simulados, si se conoce y puede usarse para determinar el parámetro que mejor califique la evolución de la red. El parámetro R^2_{real} comienza aumentando rápidamente hasta alcanzar un máximo y después de ahí disminuye suavemente, pero al final disminuye rápidamente de manera un tanto errática. Cuando se tiene el valor máximo en R^2_{real} se alcanza la mejor estructura de la RNA, $n^*_2 = N^*_{real}$. Para los casos mostrado, sus valores máximos son alcanzados en 65, 110 y 18 neuronas ocultas respectivamente, N^*_{real}.

De manera general, se ha observado que los máximos están relacionados de la siguiente manera: $N^*_{adj,e} \geq N^*_v \geq N^*_{adj,v}$. La cuestión aquí es: ¿cuál de los parámetros $N^*_{adj,e}$, N^*_v o $N^*_{adj,v}$ es el que está más cercano a N^*_{real} y proporciona la estructura de la RNA óptima? En la siguiente sección se presenta un estudio estadístico a detalle para contestar esta pregunta.

4.3. Criterio para seleccionar la mejor RNA

Se sabe que el valor óptimo de neuronas en la capa oculta debe ser lo más próximo a N^*_{real} ya que ahí es donde las predicciones de la red aproximan mejor a la superficie real (desconocida). Desde los resultados obtenidos en la Figura 4.4, se observa que tanto $N^*_{adj,v}$ como N^*_v se encuentran cercanas a N^*_{real}, pero se encuentran o antes o después. El $N^*_{adj,e}$ no es una opción debido a que se encuentra muy por encima de N^*_{real}. La cuestión es ¿Cuál de $N^*_{adj,v}$ o N^*_v está más cercana a N^*_{real} y representa el mejor criterio para escoger el número óptimo de neuronas en la capa oculta?

Con el fin de determinar el mejor criterio para obtener una estructura óptima de la RNA se lleva a cabo una serie de 20 simulaciones a diferentes condiciones (modificando el tipo de superficie, cantidad de datos, proporción usada para el conjunto de entrenamiento, proporción de ruido respecto de la amplitud de la señal) y para cada uno de ellos se determinaron los números de neuronas en donde se encuentra el máximo en los coeficientes de determinación: de validación, N^*_v; ajustado de validación, $N^*_{adj,v}$, y real, N^*_{real}. Los resultados de las 20 simulaciones se presentan en la Tabla 4.2. Se tiene que en promedio $N^*_{adj,v} \leq N^*_v \leq N^*_{real}$ $(52.30 < 55.10 < 58.25)$, sin embargo, estos datos

tienen un cierto grado de dispersión y aunque parezcan diferentes, pueden ser estadísticamente iguales. Para comprobar la similitud de esta información, se contrastan los resultados de$N^*_{adj,v}$ y N^*_v contra N^*_{real} con pruebas t-student para medias de dos muestras emparejadas. Se encuentra que $N^*_{adj,v}$ es estadísticamente diferente de N^*_{real} ($p = 0.0034$) y que N^*_v es estadísticamente igual que N^*_{real} ($p = 0.1019$).

Tabla 4.2: Escenario experimental con 20 simulaciones a diferentes condiciones de entrenamiento de una SLFN.

Caso	Función	# Datos	% Entr	%Ruido	N^*_v	$N^*_{adj,v}$	N^*_{real}
1	B	1000	90	5	56	56	52
2	A	200	50	30	34	34	58
3	B	1000	50	5	45	45	46
4	A	1000	90	5	89	89	90
5	A	1000	50	30	65	54	71
6	B	1000	70	17.5	55	55	57
7	B	600	70	30	48	35	41
8	A	600	50	5	85	80	83
9	B	400	70	17.5	50	50	57
10	A	600	70	23.75	69	69	69
11	B	200	50	30	46	46	46
12	A	200	90	30	56	29	46
13	B	600	70	17.5	60	60	60
14	A	600	70	17.5	57	57	67
15	B	600	70	17.5	35	35	34
16	B	600	70	17.5	59	59	54
17	A	600	70	17.5	59	59	63
18	A	200	90	30	21	21	38
19	A	600	50	5	82	82	90
20	B	200	90	30	31	31	43

De esta manera se concluye que, en la práctica, el número óptimo de neuronas en la capa oculta es igual a aquel en donde se tiene un máximo en R^2_v, es decir $n^*_2 = N^*_{real}$. En general, se ha encontrado que cualquier número de neuronas ocultas, n_2, entre $N^*_{adj,v}$ y $N^*_{adj,e}$ da buenos resultados.

El método de aprendizaje SOS, por ser NO iterativo, requiere de tiempos muy cortos en el entrenamiento y alcanza un mínimo global en donde la RNA tiene capacidad de generalización. Así pues, se tiene una metodología general de entrenamiento con criterios bien definidos para determinar el número óptimo de neuronas en la capa oculta, que resuelve el inconveniente de los distintos mínimos locales y los tiempos largos de entrenamiento, lo que resulta en un incentivo muy grande para el uso de las RNAs.

4.4. Efectos de underfitting y overfitting en SOS

Tomando como base el coeficiente de determinación del conjunto de validación de la Figura 4.4 (curva de puntos rojos), se observa que hay valores muy malos de R_v^2 muchas neuronas antes y después del valor óptimo, N_v^*. Esto es debido a que antes de alcanzar N_v^* la SLFN no cuenta con suficientes neuronas en la capa oculta que logren capturar la señal real de la respuesta, dando lugar a un efecto de underfitting. Por otro lado, mucho después de alcanzar N_v^* la SLFN cuenta con un número sobrado de neuronas en la capa oculta que son capaces de capturar la señal real de la respuesta junto con el error experimental, dando lugar a un efecto de overfitting. En esta etapa la red no diferencia entre el dato original y el ruido experimental. En las Figuras 4.5, 4.6 y 4.7 se muestra el efecto que tiene el número de neuronas ocultas sobre la respuesta predicha. En las Figuras 4.5*a, b* y *c* se presenta la reconstrucción de las superficies por la red entrenada con un número óptimo de neuronas (75, 120 y 20 respectivamente por cada superficie), un número de neuronas muy por debajo del óptimo (25 neuronas por cada superficie en Figuras 4.6*a* y *b*, 10 neuronas para la Figura 4.6*c*), y un número de neuronas muy por encima del óptimo (180 por cada superficie en Figura 4.7). En la Figura 4.5 se observa que la reconstrucción es excelente debido a que se tiene un número de neuronas óptimo en la capa oculta. En la Figura 4.6 se observa una mala reconstrucción de las superficies debido a que no se tienen suficientes neuronas en la capa oculta de la SLFN, sin embargo, aunque la estructura no es suficiente, trata de capturar la forma global de la superficie. En la Figura 4.7 se observa una mala reconstrucción de las superficies debido a que se tiene un número excesivo de neuronas en la capa oculta de la SLFN, y su predicción ha capturado el error experimental involuntariamente. Esto último se puede corroborar con las gráficas de predicciones contra reales, en donde los puntos del entrenamiento (color azul) se encuentran muy bien acomodados sobre la línea de 45°, pero los puntos de validación (color rojo) comienzan a salirse de la línea de 45°. Observe que, en todos los casos, el entrenamiento SOS filtra en primera instancia el ruido de todos los datos.

Después de analizar el efecto del número excesivo de neuronas en la capa oculta, se puede ver que en las Figuras 4.1*c*, 4.2*c* y 4.3*c* el corrugado que hay en la superficie es debido más al exceso de neuronas usadas en ese entrenamiento que a un mal entrenamiento por el método iterativo que usa Matlab. Así pues, este efecto de corrugado es debido al exceso de neuronas en la capa oculta y es independiente del método de entrenamiento. Este efecto de corrugado, por exceso de neuronas ocultas, no se había mostrado en trabajos anteriores de RNA.

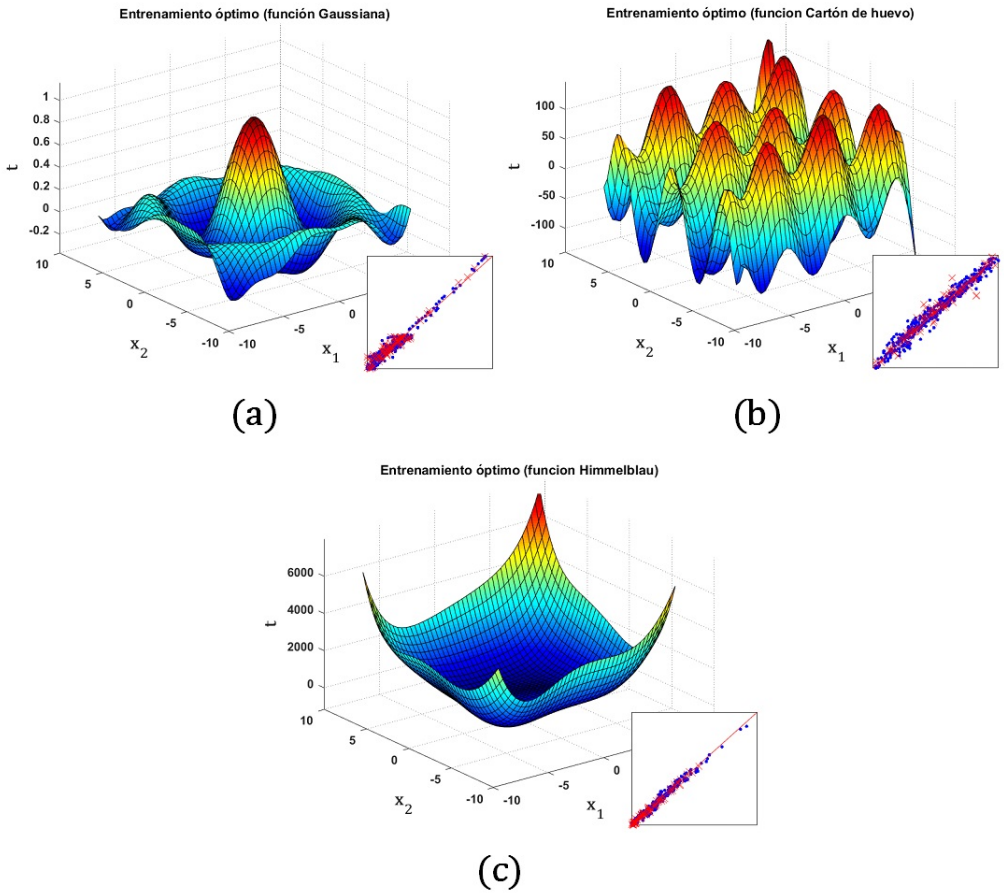


Figura 4.5: Superficies reconstruidas que muestran el efecto que tiene el número óptimo de neuronas ocultas sobre la calidad de predicción. Se usó una SLFN con topología $\{2,n_2,1\}$ donde n_2 es: a) 75, b) 120, c) 20.

4.5. Análisis de factores en el entrenamiento SOS

Hasta el momento se ha visto que el método SOS presenta ventajas con respecto a los métodos de entrenamiento típicos, que se realizan de manera iterativa, además se ha identificado el mejor criterio para determinar la estructura óptima de la SLFN, con el fin de evitar los efectos de baja generalización (underfitting) y el sobreajuste (overfitting). Dentro del desarrollo del método SOS existen algunos factores de interés que tienen la posibilidad de influir fuertemente en el desempeño de la red neuronal durante el entrenamiento. Algunos de estos factores se han analizado con anterioridad en la literatura [42, 43], pero en vista de que el método SOS no se asemeja con el entramiento tradicional (método iterativo) es necesario realizar un estudio enfocado hacia el análisis de factores que pueden mejorar o perjudicar el desempeño de la RNA. Los factores de interés que se analizan son:

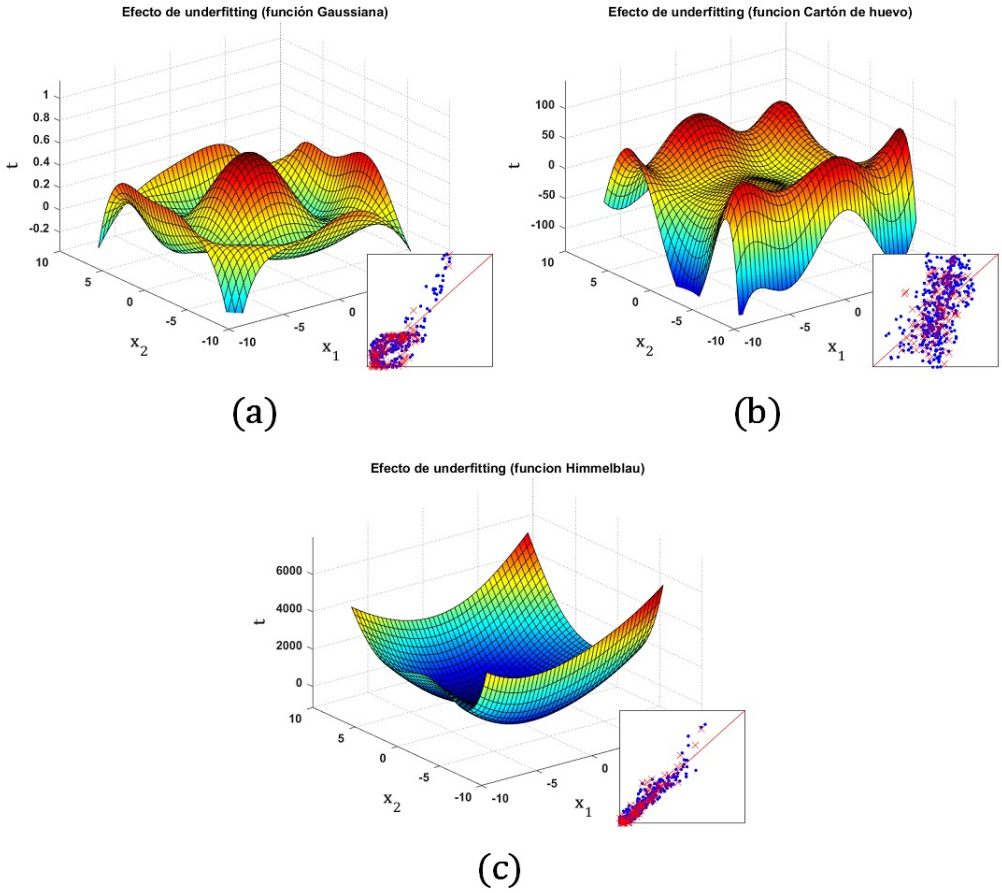


Figura 4.6: Superficies reconstruidas que muestran el efecto de underfitting sobre la calidad de predicción con un número pequeño de neuronas ocultas. Se usó una SLFN con topología {2,n_2,1} donde n_2 es: a) 25, b) 25, c) 10.

- **Porcentaje de entrenamiento**. Durante el entrenamiento de una RNA es común utilizar la validación cruzada, con la finalidad de medir la calidad de la predicción durante el ajuste de los parámetros implicados en la red. La validación cruzada implica dividir en dos subconjuntos el total de los datos con los que se cuenta (conjunto de entrenamiento y conjunto de validación). En la literatura existe un poco de ambigüedad sobre el porcentaje en el que debe ser dividido el conjunto de datos, además, no se han presentado estudios donde involucren tal parámetro. Por lo general, se deja a criterio del investigador o se utiliza la regla 80-20 %.

- **El número de datos para el entrenamiento**. En la práctica nunca se suele especificar la cantidad mínima de datos requeridos para obtener un desempeño favorable. No hay reglas para determinar el número adecuado de datos para

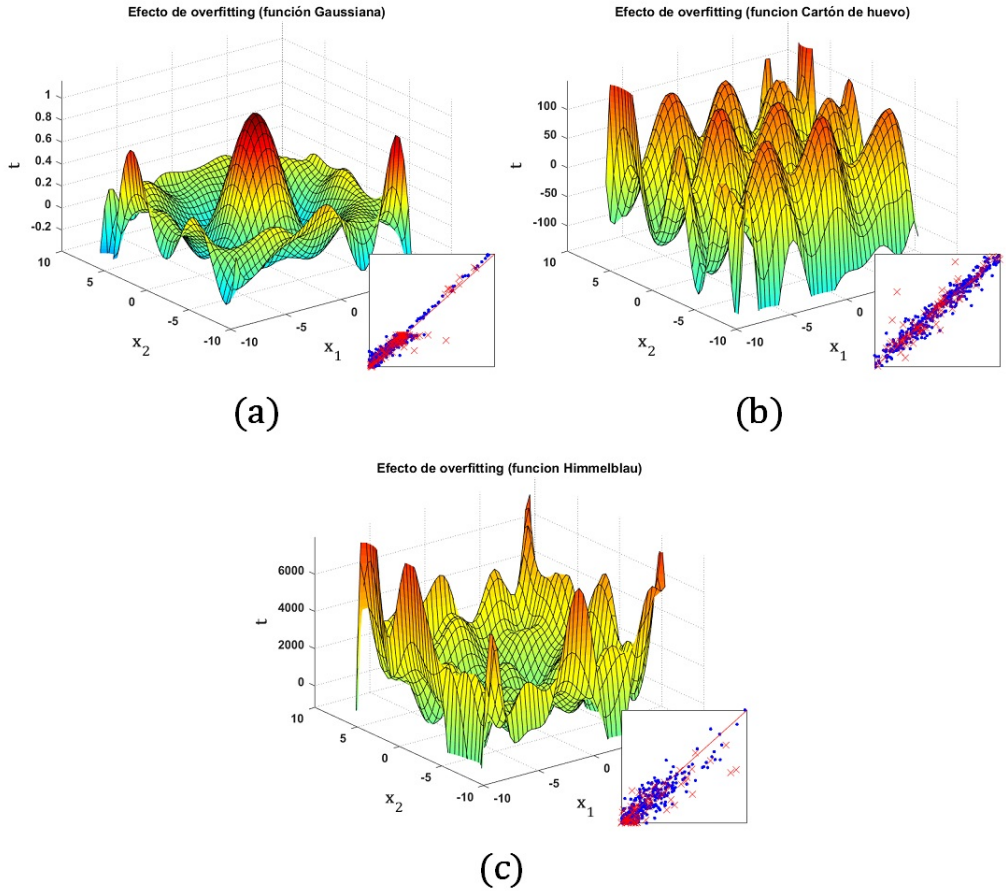


Figura 4.7: Superficies reconstruidas que muestran el efecto de overfitting sobre la calidad de predicción con un número muy grande de neuronas ocultas. Se usó una SLFN con topología $\{2,n_2,1\}$ donde n_2 es: a) 180, b) 180, c) 180.

realizar el modelamiento de un sistema mediante una RNA. Los expertos en el área solo suelen definir que entre más datos se tenga la RNA será capaz de realizar una mejor predicción. En la mayoría de los casos no se menciona en que magnitud puede influir el tamaño de la muestra sobre la calidad de la predicción.

- **El rango de inicialización de los pesos**. Cuando se generan los pesos sinápticos iniciales para un entrenamiento usualmente se hace entre ± 1, $\omega = \pm 1$. En nuestros trabajos se ha observado que el valor de inicialización de los pesos puede influir considerablemente en el rendimiento de la RNA. Por tal motivo, es de interés analizar el impacto que puede ocasionar.

- **El porcentaje de ruido que poseen los datos**. Al momento de modelar un sistema a partir de datos experimentales, estos suelen contener un nivel de ruido (error experimental). El ruido puede ser provocado por el mismo sistema o por un

error humano durante su medición. Cuando se trabaja con datos reales el nivel de ruido es desconocido para el investigador. Aquí se opta por trabajar con datos sintéticos que permitan hacer una comparación directa entre las predicciones de la RNA y los datos reales (sin ruido). Los datos sintéticos pueden ser manipulados a conveniencia, es decir, a partir de una superficie ya conocida y definida es posible crear un conjunto de datos a los cuales se les agrega un porcentaje de ruido especifico, con el objetivo de observar si el método SOS es capaz de filtrar el ruido experimental y capturar la señal.

- **Tipo de superficie**. Implica analizar el desempeño de la red para diferentes superficies, donde, cada superficie se considera un sistema diferente. Se desea analizar si el rendimiento de la red es dependiente o independiente del sistema que se esté analizando.
- **Matriz estocástica**. Existen dos formas de generar la matriz estocástica $\underline{\underline{W}}^1 \in \mathbb{R}^{(n_1+1)\times n_2}$, 1) vector: conforme se incrementa el número de neuronas en la capa oculta (p.e. una en una) es generado un vector $\underline{W}^* \in \mathbb{R}^{(n_1+1)\times 1}$ y se concatena a la matriz $\underline{\underline{W}}^1$, $W^1_{new} = \left[\underline{\underline{W}}^1 \ \underline{W}^*\right]$; 2) matriz: ocurre al momento de cambiar en número de neuronas ocultas n_2 es posible generar la matriz $\underline{\underline{W}}^1$ cambiado todos los pesos.

Para el análisis de los factores de interés se realizó un estudio de ANOVA con un nivel de significancia del 5 %. En la Tabla 4.3 se presentan los distintos niveles de los 6 factores implicados. En el Apéndice A se presenta el plan experimental que se utilizó para este análisis.

Tabla 4.3: Niveles de los distintos factores para el ANOVA.

Factores	Niveles	
Relación entre./valid. ($R_{E/V}$)	50/50	90/10
Número de datos (ND)	400	1000
Rango de los pesos (ω)	± 0.1	± 10
Porcentaje de ruido ($\%R$)	5 %	30 %
Tipo de superficie (Sup)	Cartón de huevo	Gaussiana
Matriz estocástica (G_W)	Matriz	Vector

Las variables de respuesta a analizar son: MSE entre las diferencias del valor real sin ruido (T_{real}) y el valor predicho por la red $\left(\underline{y}^3\right)$, el número de neurona en que se presenta la singularidad (N_s), para analizar dicha respuesta es necesario calcular la diferencia entre $(N_s - N_v^*) = \Delta N$. A continuación, se explica a detalle el significado de N_s y la singularidad en el método SOS. Se usa un plan experimental para superficie de respuesta con modelo cuadrático en los factores.

En la Sección 3.1 se describen varias observaciones con respecto a la Ecuación 3.6. Durante el entrenamiento de la RNA se hace uso de la inversa izquierda de $\left[\underline{1}\ \underline{\underline{y}}^2\right]$, también denominada *pseudoinversa.* Para que la pseudoinversa exista, se debe cumplir: 1) debe existir la inversa de la matriz cuadrada $\left[\underline{1}\ \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1}\ \underline{\underline{y}}^2\right]$; 2) los vectores columna de $\left[\underline{1}\ \underline{\underline{y}}^2\right]$ sean linealmente independientes. Ambas condiciones implican que $\tanh\left(\left[\underline{1}\ \underline{\underline{X}}\right] \cdot \underline{\underline{W}}^1\right)$ resulte en una matriz con vectores columna linealmente independientes o al menos se aleje de la dependencia lineal. Lo último se puede lograr si el número de observaciones experimentales es mucho mayor que el número de neuronas ocultas, $m \gg n_2$. Se ha observado, que si a pesar de que la última condición se cumpla y la matriz $\left[\underline{1}\ \underline{\underline{y}}^2\right]$ siga presentando dependencia lineal se puede cambiar el rango de los pesos estocásticos (ω) de $\underline{\underline{W}}^1$ con el objetivo de que la singularidad de la matriz $\left(\left[\underline{1}\ \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1}\ \underline{\underline{y}}^2\right]\right)^{-1}$ se presente a un número grande de neuronas ocultas, esperando que la singularidad se presente por encima de N_v^*. Es aquí donde el termino ΔN toma relevancia, si el valor de ΔN es positivo significa que la singularidad se presentó por encima de N_v^* (caso deseado), caso contrario, si ΔN es negativo significa que la singularidad se presentó antes del número óptimo de neuronas (caso no deseado). A partir del momento en que se presenta la singularidad, es imperante hacer uso del cálculo de la pseudoinversa por el método de la descomposición en valores singulares, lo cual es tardado y consume muchos recursos. Es por ello que para tener tiempos cortos, en el entrenamiento SOS, es de vital importancia mandar la singularidad muy por encima de N_v^*. El calculo de la pseudoinversa conlleva a hacer uso de más requerimiento computacional y mayor tiempo de procesamiento durante el entrenamiento, lo cual es algo no deseado en el método propuesto.

Los resultados del ANOVA para la respuesta MSE se presenta en la Tabla 4.4. Para la estabilización de la varianza, fue necesario una transformación logarítmica a la respuesta. Los coeficientes en la Tabla 4.4 muestran los términos codificados.

En la Tabla 4.4, se observa que para el MSE, la probabilidad del término constante es < 0.05, lo que hace que el análisis general sea estadísticamente significativo, es decir, los factores tienen cierta influencia sobre la respuesta (MSE). El coeficiente de determinación estimado $\left(R^2\right)$ fue de 0.9898 para este análisis, lo que indica que aproximadamente el 98 % de la variabilidad de los parámetros se contabilizan para el modelo estadístico.

Además, el factor individual G_W mostró efectos no significativos sobre la respuesta (MSE), lo cual implica que la forma de generar la matriz estocástica no influye sobre la calidad de la predicción de la RNA. En cambio, los términos individuales ND, ω, $\%R$ y Sup mostraron efectos significativos, es decir, el número de datos, el porcentaje de ruido y el tipo de superficie afectan de manera directa la calidad de la respuesta de

Tabla 4.4: Coeficientes de regresión lineal y probabilidad "P" del modelo cuadrático para MSE.

MSE $\left(R^2 = 0.9898\right)$		
Término	Coeficiente	P-valor
Constante	-0.100	< 0.0001
ND	-0.40	< 0.0001
ω	-1.27	< 0.0001
$\%R$	0.59	< 0.0001
Sup	5.92	< 0.0001
G_W	-0.14	0.0989*
ND, ω	-0.31	0.0049
ND, $\%R$	0.44	0.0001
ω, $\%R$	0.35	0.0029
ω, G_W	-0.43	< 0.0001
$\%R, Sup$	-0.20	0.0475
ω^2	1.29	< 0.0001

* factores no significativos p>0.05

la RNA.

Para el factor ω se observan efectos significativos tanto lineales como cuadráticos, lo que denota su importancia en el modelo.

Los resultados del ANOVA para la respuesta ΔN se presentan en la Tabla 4.5. Para este caso no fue necesario una transformación a la respuesta. Los coeficientes en la Tabla 4.5 muestran los términos codificados.

Se observa que para ΔN, la probabilidad del término constante es < 0.05, lo que hace que el análisis general sea estadísticamente significativo, es decir, los factores tienen cierta influencia sobre la respuesta. El coeficiente de determinación estimado $\left(R^2\right)$ fue 0.5861 para este análisis, a pesar de que su valor es bajo, no significa que el análisis sea malo, el valor bajo de R^2 indica un menor ajuste del modelo.

Además, factores individuales como ND, $\%R$, Sup y G_W mostraron efectos no significativos sobre la respuesta (ΔN), en cambio, el término individual ω mostró un efecto muy significativo, tanto lineal como cuadrático. De modo que los resultados nos llevan a determinar que el rango de inicialización de los pesos estocásticos influye fuertemente sobre la singularidad de la matriz $\left[\underline{1} \;\; \underline{\underline{y}}^2\right]$. Ahora debemos determinar los valores más apropiados de los pesos estocásticos con los cuales se debe realizar el entrenamiento de la RNA.

Tabla 4.5: Coeficientes de regresión lineal y probabilidad "P" del modelo cuadrático para ΔN.

$\Delta N \ \left(R^2 = 0.5861\right)$		
Término	Coeficiente	P-valor
Constante	-162.41	< 0.0001
ND	-37.04	$0.1090*$
ω	143.43	< 0.0001
$\%R$	-41.03	$0.0815*$
Sup	29.14	$0.1552*$
G_W	-19.80	$0.3286*$
ND, $\%R$	-54.98	0.0341
ω, Sup	-69.09	0.0049
Sup, G_W	-41.36	0.0456
ω^2	174.90	0.0008

* factores no significativos p>0.05

4.6. Rango de inicialización de los pesos y su efecto de en el método SOS

El ANOVA para ΔN proporciona información adicional sobre el rango (ω) de inicialización de los pesos estocásticos y su afecto hacia la respuesta, es decir, la presencia de singularidades en el entrenamiento de la RNA.

En la Figura 4.8 se presenta un análisis de superficie de respuesta, donde se grafica el valor de ΔN en función de ω^* y la relación entre el conjunto de entrenamiento/validación $\left(R_{E/V}\right)$.

En la sección anterior se mencionó que existe una condición para que la matriz $\left[\underline{1} \ \underline{\underline{y}}^2\right]$ sea linealmente independiente, consiste en que el número de observaciones experimentales es mucho mayor que el número de neuronas ocultas, $m \gg n_2$. Dicha condición se cumple en el entrenamiento SOS, debido a que el número máximo de neuronas ocultas (n_2) que se puede alcanzar en la red es: $m_e - 1$.

En la Figura 4.8 a y b se presenta los valores de ΔN para la función Gaussiana con $m = 400$ y $m = 1000$ respectivamente, en el plano $\omega^*, R_{E/V}$ se muestra una línea de contorno que divide los valores positivos y negativos de ΔN, se observa que ΔN es positivo cuando el valor de ω^* se encuentra cerca de 1.0. El valor de ω está codificado de manera logarítmica $\left(10^{\omega^*} = 10^{1.0} = 10, \omega = \pm 10\right)$, en el caso opuesto, el valor de ΔN es negativo cuando $\omega^* < 0.5$, es decir, $\left(10^{0.5} = 3.16, \omega = \pm 3.16\right)$.

El mismo comportamiento se observa para la función de Cartón de huevo, Figura 4.8 c y d, con $m = 400$ y $m = 1000$ respectivamente, donde los valores más altos de ΔN se logran cuando el valor de $\omega^* = 1.0$. En la Figura 4.8 también se observa que el

efecto del número de datos para entrenar la red no tiene un efecto significativo sobre el valor de ΔN, lo cual se confirma con el estudio de ANOVA, Tabla 4.5.

El rango de inicialización de los pesos estocásticos (ω) tiene un efecto no lineal sobre ΔN, es decir, que la singularidad que puede estar presente en el proceso de entrenamiento SOS se puede desplazar si se elige de manera correcta el rango de ω. Esto resulta ser contraintuitivo debido a que los métodos de entrenamiento tradicionales que hacen uso de pesos estocásticos [19, 37, 44–51] sugieren usar pesos en un rango de ± 1. Como se ha observado, el rango de ω apropiado para el entrenamiento SOS es de ± 10.

Se ha visto cómo afecta ω en la singularidad dentro del entrenamiento SOS, ahora, se describe el efecto que tiene el factor ω sobre la calidad de la respuesta, en la Figura 4.9 se presenta un análisis de superficie de respuesta donde se grafica el logaritmo de MSE en función de ω^* y el conjunto de entrenamiento/validación $\left(R_{E/V}\right)$. Se puede observar que ω^* tiene un efecto no lineal sobre MSE. Cuando el número de datos es menor, $m = 400$ Figura 4.9*a* y *c*, se observa que los valores más bajos de MSE se presentan cuando ω^*, se encuentra en un rango de $[-0.5, 0.5]$, es decir, cuando los pesos estocásticos se generan en un rango de ± 0.316 y ± 3.16, de manera similar, cuando el número de datos es mayor, $m = 1000$ Figura 4.9*b* y *d*, los valores más bajos de MSE se presentan cuando los pesos estocásticos se generan en un rango de ± 3.16.

4.7. Entrenamiento extendido a múltiples respuestas

Cuando hay varias respuestas de salida, el entrenamiento iterativo de una RNA que utiliza la regla delta se logra al minimizar la función de error total (Ecuación 4.1) que es la suma de los error de cada respuesta (Ecuación 3.1)

$$e(\underline{w}) = SSE = \sum_{i=1}^{m}\sum_{j=1}^{n_3}\left(t_{i,j} - y_{i,j}^3\right)^2 = trace\left(\left(\underline{\underline{T}} - \underline{\underline{y}}^3\right)^T \cdot \left(\underline{\underline{T}} - \underline{\underline{y}}^3\right)\right) \tag{4.1}$$

donde $trace(\bullet)$ es la función traza de la matriz (suma de los elementos de la diagonal principal) y la Ecuación 2.11 es modificada a:

$$\underline{\underline{y}}^3 = \left[\underline{1} \;\; \underline{\underline{y}}^2\right] \cdot \underline{\underline{W}}^2 \in \mathbb{R}^{m \times n_3} \tag{4.2}$$

donde $\underline{\underline{W}}^2 \in \mathbb{R}^{(n_2+1)\times n_3}$ es ahora una matriz de pesos sinápticos.

Todas las otras ecuaciones descritas en el Capitulo 3 permanecen iguales. La Ecuación 4.1 plantea una función error total, que es la contribución de todas las respuestas, al ser minimizada, asigna la misma importancia a cada respuesta. Lo cual puede ser

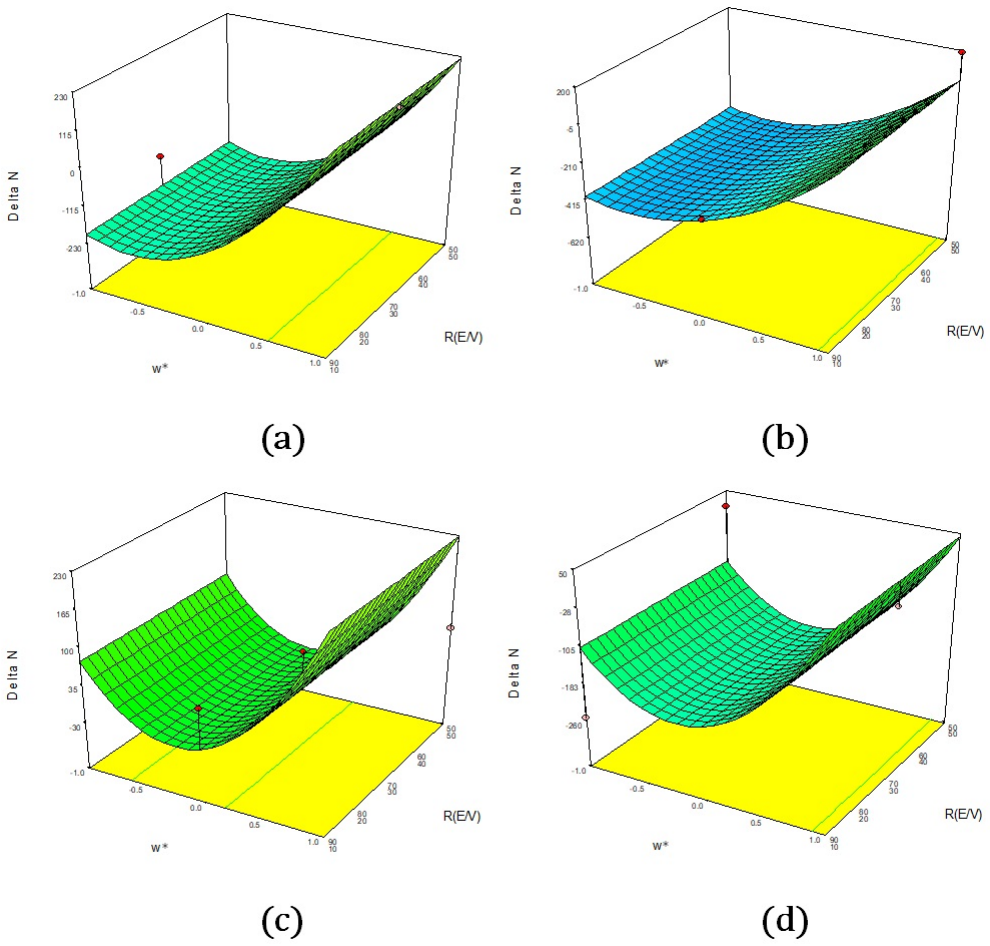


Figura 4.8: Gráficos seleccionados del análisis de superficie de respuesta: (a) Respuesta de ΔN en función de ω^* y $R_{E/V}$ para la función Gaussiana con $m = 400$, (b) Respuesta de ΔN en función de ω^* y $R_{E/V}$ para la función Gaussiana con $m = 1000$, (c) Respuesta de ΔN en función de ω^* y $R_{E/V}$ para la función Cartón de huevo con $m = 400$, (d) Respuesta de ΔN en función de ω^* y $R_{E/V}$ para la función Cartón de huevo con $m = 1000$. Todas las superficies de respuesta tienen las siguientes condiciones: $\%R = 30$ y $G_W =$ columna.

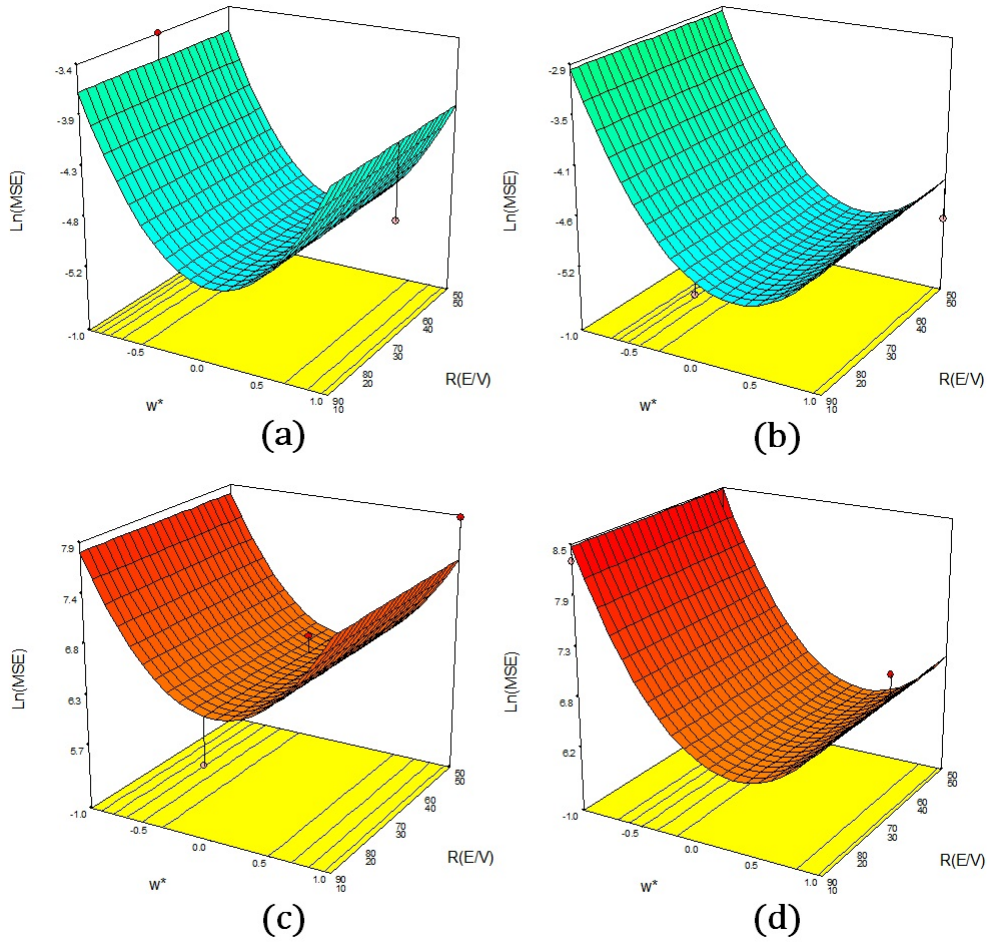


Figura 4.9: Gráficos seleccionados del análisis de superficie de respuesta: (a) Respuesta de $\ln(MSE)$ en función de ω^* y $R_{E/V}$ para la función Gaussiana con $m = 400$, (b) Respuesta de $\ln(MSE)$ en función de ω^* y $R_{E/V}$ para la función Gaussiana con $m = 1000$, (c) Respuesta de $\ln(MSE)$ en función de ω^* y $R_{E/V}$ para la función Cartón de huevo con $m = 400$, (d) Respuesta de $\ln(MSE)$ en función de ω^* y $R_{E/V}$ para la función Cartón de huevo con $m = 1000$. Todas las superficies de respuesta tienen las siguientes condiciones: $\%R = 30$ y $G_W =$ columna.

conciderado incorrecto, ya que no todas las respuestas tienen el mismo grado de complejidad estructural. En la Sección 4.2 se demostró que cada respuesta necesita diferente número de neuronas ocultas.

Una RNA entrenada con métodos tradicionales puede dar poca importancia a una respuesta estructuralmente compleja y mucha importancia a una respuesta estructuralmente sensilla, lo que resulta en una RNA deficiente.

En nuestros estudios hemos encontrado que cada respuesta tiene diferente grado de importancia que depende principalmente de la complejidad de la respuesta y por lo tanto necesita diferente número de neuronas ocultas.

Por ejemplo, una respuesta con varias crestas y valles requiere de una RNA con mayor cantidad de neuronas ocultas que una respuesta suave con una sola cresta y/o valle. Con esta idea presente, es posible entrenar una RNA por cada respuesta y medir la calidad de la predicción de manera independiente.

Esto nunca antes se ha propuesto debido a que los métodos iterativos son muy demandantes de tiempo, y sí se le agrega entrenar una RNA por cada respuesta, entonces es una propuesta muy temeraria. Sin embargo, en el contexto de entrenamiento SOS, es una propuesta muy adecuada y solo consume tiempo adicional para calcular las R_v^2 por cada respuesta. De esta manera, la Ecuación 3.6, para múltiples respuestas, se transforma en

$$\underline{\underline{\tilde{W}}}^2 = \left(\left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1} \;\; \underline{\underline{y}}^2\right]\right)^{-1} \left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \underline{\underline{T}} \in \mathbb{R}^{(n_2+1)\times n_3} \tag{4.3}$$

donde cada vector columna de la matriz $\underline{\underline{\tilde{W}}}^2$ corresponde a los parámetros de cada variable de salida como si se hubieran entrenado por separado.

Como puede verse en la Ecuación 4.3, para un número dado de neuronas ocultas, el entrenamiento no ocupa más tiempo del que se usaría para una sola respuesta.

Para determinar el número de neuronas óptimas por cada respuesta se comienza con una neurona en la capa oculta. Se asignan los parámetros $\underline{\underline{W}}^1$ de manera estocástica y con la Ecuación 4.3 se obtienen los $\underline{\underline{\tilde{W}}}^2$ optimizados. Realizando un barrido, es decir, moviendo las neuronas ocultas de manera secuencia de una en una (o con incrementos más grandes) desde 1 hasta $m_e - 1$ neuronas. En cada paso se calcula el R_v^2 por cada respuesta.

Con cada neurona que se agrega, también crece la matriz $\underline{\underline{W}}^1$ en una columna (este proceso se discutió en la Sección 4.5, "forma de generar la matriz estocástica") y la matriz $\underline{\underline{\tilde{W}}}^2$ en un renglón. Sin embargo, a la matriz $\underline{\underline{W}}^1$ se le agrega un vector columna estocástico al final (se mantienen sin cambio las columnas precedentes), mientras que la matriz $\underline{\underline{\tilde{W}}}^2$ se renueva por completo a partir de la Ecuación 4.3.

Al final, se encuentra el número de neurona ocultas en donde se tiene el valor

máximo de R_v^2 para cada respuesta por separado. Esos valores de N_v^* representan las neuronas necesarias para dicha respuesta y con esos valores se reconstruye la RNA final.

Por ejemplo, si la primera respuesta requiere 30 neuronas y la segunda requiere 50 neuronas, entonces se asigna estocásticamente $\underline{\underline{W}}^1$ con 30 columnas y con la Ecuación 3.6 se calcula el vector columna $\underline{\tilde{W}}^{2,1} \in \mathbb{R}^{(31\times1)}$ con los datos de la primera respuesta y luego se aumenta con 20 ceros al final para completar $\underline{\tilde{W}}^{2,1} \in \mathbb{R}^{(51\times1)}$.

Después se agregan 20 vectores columna estocásticos al final de la matriz $\underline{\underline{W}}^1$ y con la Ecuación 3.6 se ajusta el vector columna $\underline{\tilde{W}}^{2,2} \in \mathbb{R}^{(51\times1)}$. Al final se reconstruye la matriz $\underline{\underline{W}}^2 = \left[\begin{array}{cc} \underline{\tilde{W}}^{2,1} & \underline{\tilde{W}}^{2,2} \end{array}\right] \in \mathbb{R}^{(51\times1)}$ y junto con $\underline{\underline{W}}^1$ se tiene la estructura optimizada de una RNA, esto se ilustra en la Figura 4.10.

Cuando se simulan múltiples respuestas con las superficies estudiadas aquí (Figura 3.2), se obtienen prácticamente los mismos resultados que se presentan en la Figura 4.5.

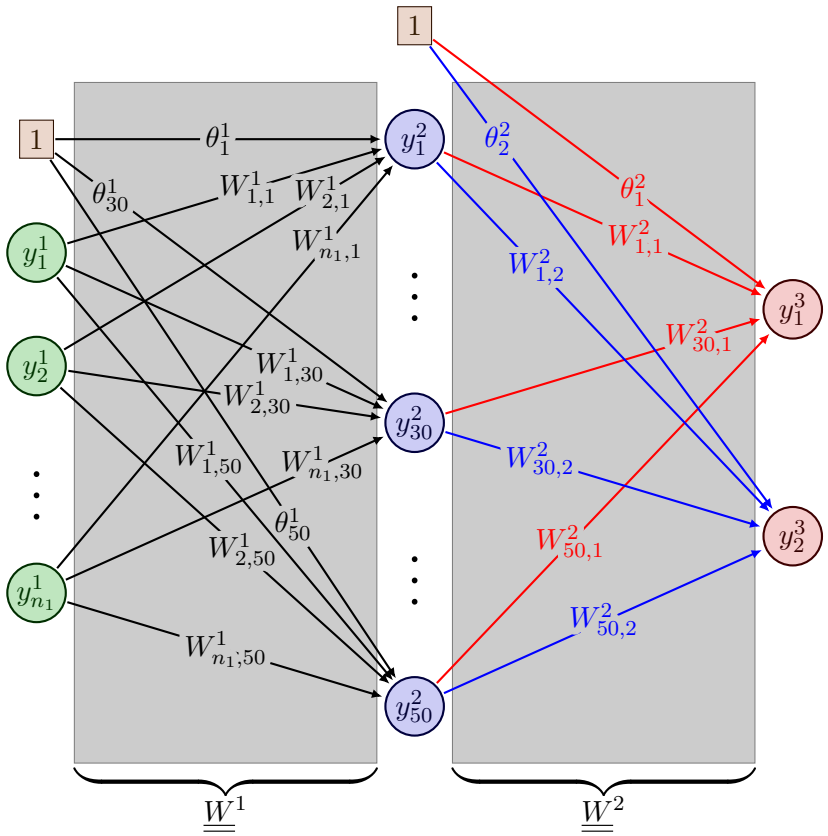
Hemos aplicado el procedimiento SOS a los 60000 datos de la base de datos MNIST ("Modified National Institute of Standards and Technology") [52] y se han obtenido buenos resultados. Se usaron 36000 datos escogidos aleatoriamente para el conjunto de entrenamiento y el resto para la validación y se obtuvo una SLFN {784,7451,10} que falla solamente en el 2.72 % de los 10000 datos del conjunto test. Los resultados principales del entrenamiento se presentan en la Tabla 4.6

Tabla 4.6: Resultados de una SLFN $\{784, 7451, 10\}$ entrenada con el método SOS y usando 36000 datos de la base MNIST.

Dígito	0	1	2	3	4	5	6	7	8	9
R_e^2	0.950	0.957	0.910	0.893	0.911	0.892	0.945	0.902	0.878	0.846
$R_{adj,e}^2$	0.936	0.943	0.887	0.864	0.882	0.862	0.928	0.879	0.844	0.812
R_v^2	0.902	0.923	0.829	0.798	0.834	0.802	0.882	0.844	0.770	0.760
$R_{adj,v}^2$	0.874	0.897	0.784	0.743	0.780	0.747	0.845	0.808	0.706	0.706
N_v^*	6801	7451	6301	6401	7301	6501	7201	5601	6601	5501
Fallas	8	10	40	26	26	24	20	44	31	43
Casos	980	1135	1032	1010	982	892	958	1028	974	1009

4.8. Selección del número de componentes principales y entrenamiento SOS

Se aplican los distintos criterios a las bases de datos, estos criterios otorgan una información a priori del número apropiado de CPs antes de introducirlos a un modelo, aunque no aseguran tener la mejor calidad en la respuesta. Por lo tanto, se presentan



$$\underline{\underline{W}}^1 = \begin{bmatrix} \theta^1_1 & \theta^1_2 & \cdots & \theta^1_{50} \\ W^1_{1,1} & W^1_{1,2} & \cdots & W^1_{1,50} \\ W^1_{2,1} & W^1_{2,2} & \cdots & W^1_{2,50} \\ \vdots & \vdots & \ddots & \vdots \\ W^1_{n_1,1} & W^1_{n_1,2} & \cdots & W^1_{n_1,50} \end{bmatrix} \quad \underline{\underline{W}}^2 = \begin{bmatrix} \theta^2_1 & \theta^2_2 \\ W^2_{1,1} & W^2_{1,2} \\ W^2_{2,1} & W^2_{2,2} \\ \vdots & \vdots \\ W^2_{30,1} & W^2_{30,2} \\ 0 & W^2_{31,2} \\ \vdots & \vdots \\ 0 & W^2_{50,2} \end{bmatrix}$$

Figura 4.10: Reconstrucción de una RNA mediante el método SOS con más de una neurona en la capa de salida, mediante el uso de una matriz amputada. Donde la primera respuesta solo necesita 30 neuronas ocultas y la segunda respuesta necesita las 50 neuronas ocultas.

la información obtenida por estos criterios y es comparada con la búsqueda de mallado proveniente de usar una RNA variando el número de CPs. El número de CPs que otorga el mayor coeficiente de determinación se toma como el CP óptimo.

Se utilizan las siguientes bases de datos. La base de datos MNIST ("Modified National Institute of Standards and Technology") es una base de datos estándar que contiene 70´000 imágenes de dígitos escritos a mano, de los cuales 60'000 son utilizados para entrenar y ajustar el modelo y 10'000 datos para prueba, esta base de datos es comúnmente usada en modelos de aprendizaje automático, además, se tiene un ranking en [52] donde es posible ver los diferentes modelos que la han utilizado y sus respectivos rendimientos en precisión.

Como observación en la mayoría de los modelos reportados que hacen uso de esta base de datos utilizan las 784 variables de entrada, cada variable representa un píxel de una imagen de 28×28 pixeles.

La base de datos Candy-Process fue recolectada de un proceso de confitería donde se reportan tanto variables de calidad de las materias primas como variables de proceso, dando como resultado 20 variables de entrada y dos variables de salida que identifican la calidad del producto final. La base se compone de datos historiográficos con 300 observaciones para entrenar y 100 para prueba.

En la Tabla 4.7 se presenta un resumen de las distintas bases de datos utilizadas, donde se identifican el número de atributos (variables de entrada), la cantidad de datos, tanto de entrenamiento, validación y prueba, y sus variables de respuesta.

Tabla 4.7: Base de datos para el análisis de componentes principales.

Nombre	Datos de			Variables	Variables
	Entrenamiento	Validación	Prueba	de entrada	de salida
MNIST*	24'000	8'000	10'000	784	10
Candy-Process	240	60	100	20	2

*Solo se utilizó la mitad de los datos de la cantidad original.

4.8.1. Criterios heurísticos y estadísticos

Se presentan los resultados de los distintos criterios, donde se determina el número de componentes principales para la base de datos de MNIST en la Figura 4.11.

En la Figura 4.12 se presenta los resultados para la base de datos de Candy-Process.

Los distintos criterios tratan de reducir la dimensionalidad del espacio original tomando solo la información del espacio de atributo (variables explicativas), se observa que cada criterio proporciona distinta información.

Es interesante remarcar que ninguno de los criterios presentados toma en cuenta

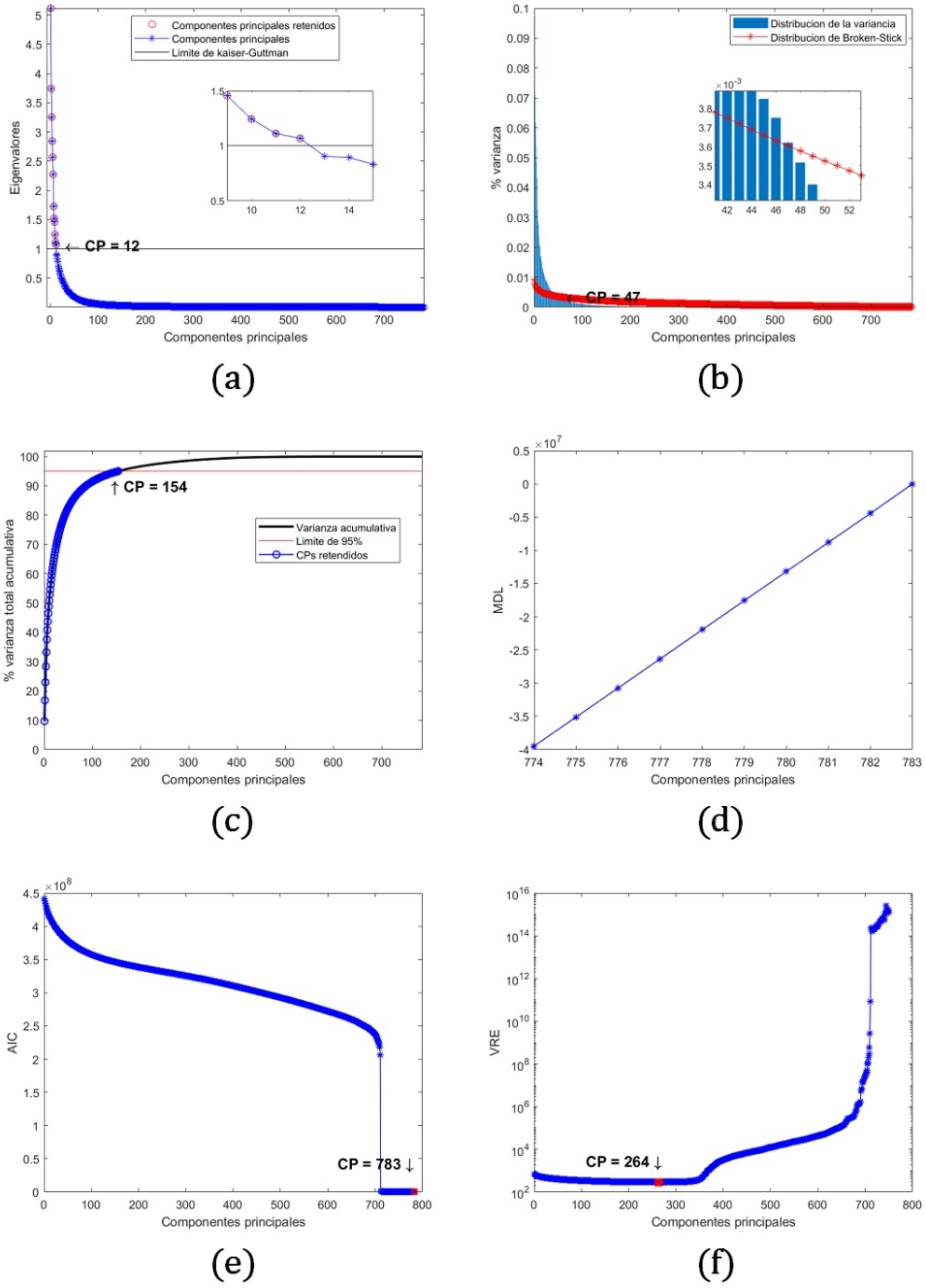


Figura 4.11: Criterios para determinar el número de componentes principales para MNIST: a) criterio de Kaiser-Guttman, b) criterio de Broken-Stick, c) porcentaje de varianza total acumulada, d) MDL, e) AIC, f) VRE.

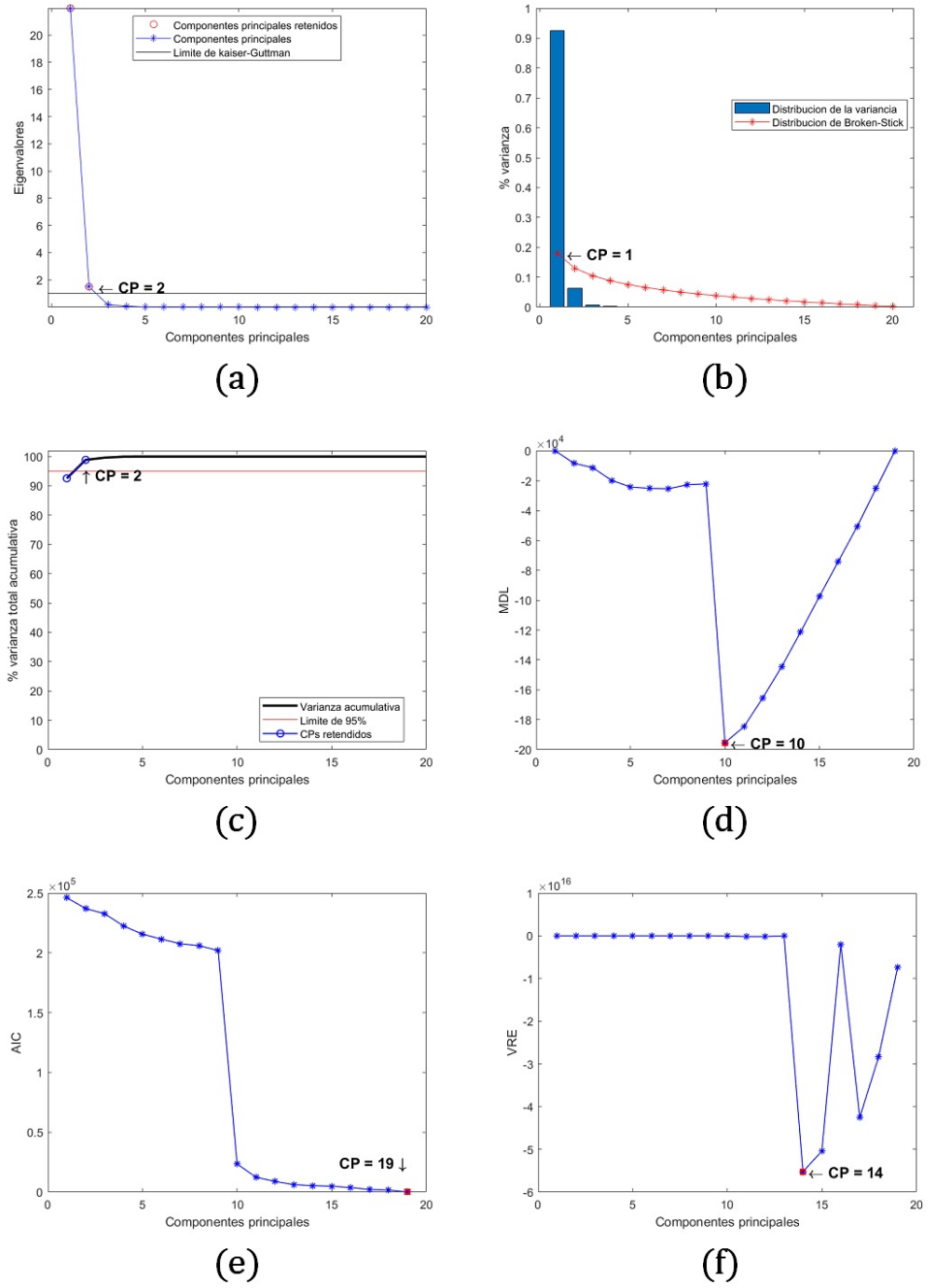


Figura 4.12: Criterios para determinar el número de componentes principales para Candy-Process: a) criterio de Kaiser-Guttman, b) criterio de Broken-Stick, c) porcentaje de varianza total acumulada, d) MDL, e) AIC, f) VRE.

la proyección del espacio de las variables explicativas tiene hacia el espacio de las respuesta. El implementar un modelo a partir de cada criterio puede llevar a distintos resultados.

4.8.2. Método riguroso

El implementar una red neuronal con un número determinado de CPs bajo un entrenamiento tradicional puede llevar un tiempo considerable, ahora si se desea analizar la calidad de la predicción de la red variando el número de variables de entrada puede ser aún más tardado, una alternativa es utilizar un método de entrenamiento eficiente que proporcione resultados confiables en un corto tiempo. Por tal razón se implementa el método SOS para solucionar dicho problema.

En la Figura 4.13 se presenta el análisis de la variación de la calidad de predicción en base al coeficiente de determinación variando el número de CPs, este proceso otorga el número de CPs apropiados para obtener la mejor calidad proveniente de una red neuronal.

Como se observa, al momento de variar el número de CPs, el coeficiente de determinación $\left(R^2\right)$ alcanza un valor máximo, este valor indica el número óptimo de CPs que garantiza la mejor calidad en la respuesta. Para la base de datos Candy-Process, se tienen dos variables de respuesta (fracción de solidos disueltos y pH, respectivamente). En la Figura 4.13a se observa que el R_v^2 alcanza un máximo en el décimo CP, así mismo, en este CP se logra un mínimo en el MSE.

En la Figura 4.13b se observa que el máximo valor de R_v^2 se alcanza en el 14° CP, mientras que el MSE logra un mínimo en el décimo CP. En base a lo anterior, se establece que el número óptimo de CPs para este caso en particular es de 10, aunque para ambas respuestas el valor máximo de R_v^2 se encuentra en distintos puntos, lo mejor es tomar solo los 10 primeros CPs.

Para la base de datos MNIST, Figura 4.13c, se presenta la calidad de la respuesta contra el número de CPs. Este caso en particular es un problema de clasificación, donde se tiene una imagen de 28×28 pixeles, esto genera un vector de entrada de 784 variables. Cada imagen está asociada a un dígito entre el 0 y 9, el objetivo es clasificar de forma correcta cada imagen con su respectiva categoria. Por lo tanto, la RNA debe tener 10 neuronas en la capa de salida, cada una asociada a cada dígito (0 al 9).

Con cada CP que se retiene se entrena una RNA, en la Tabla 4.8 se reportan los R^2 asociados a cada dígito (variable de salida), además, se presenta el valor de la precisión que se obtiene al retener cierta cantidad de CPs.

Por lo tanto, al retener cada vez más CPs la precisión de la red cambia, el valor de la precisión inicia en 76.39 % utilizando solo los primeros cinco CPs, este valor alcanza

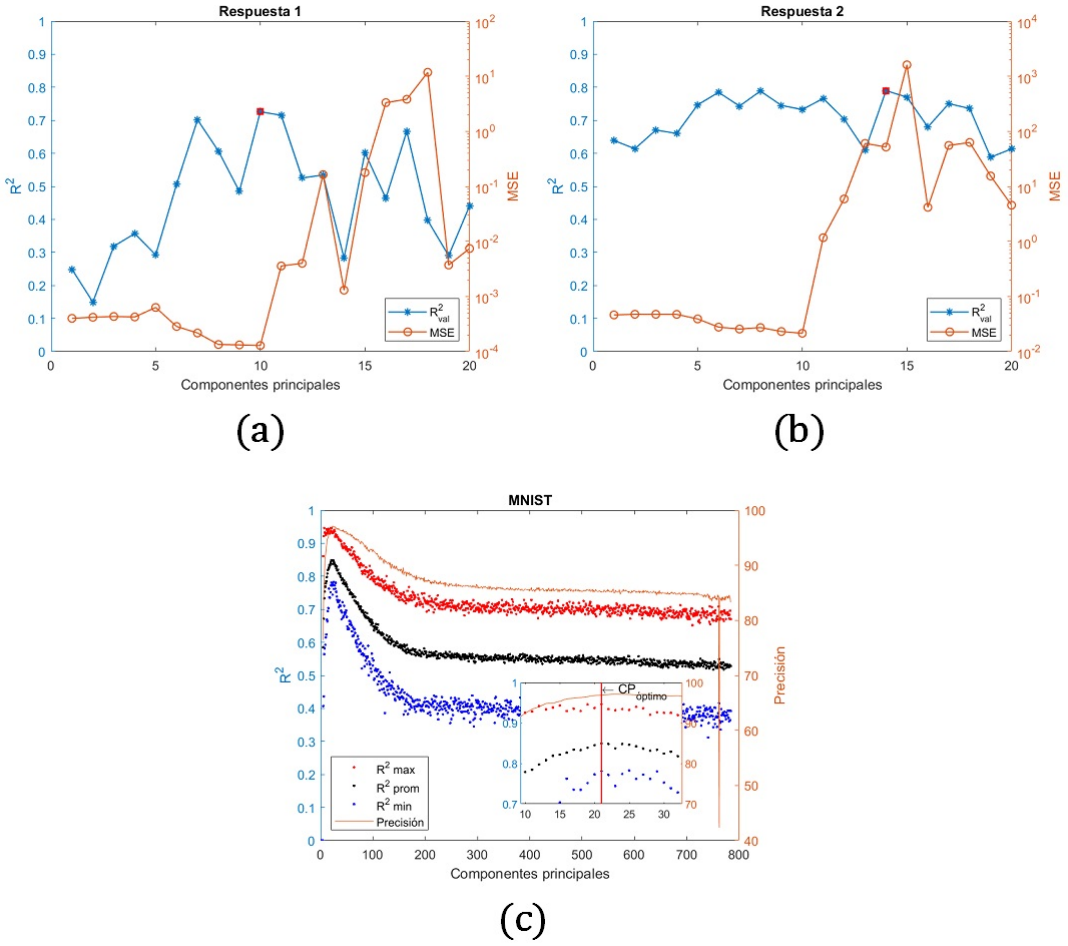


Figura 4.13: Análisis de la variación de la calidad de la respuesta en términos de (R^2) contra el número de componentes principales: a) Candy-Process respuesta 1, b) Candy-Process respuesta 2 y c) MNIST.

Tabla 4.8: Coeficientes de determinación para cada dígito y su precisión en función de los componentes principales retenidos.

CPs	R_v^2 para cada dígito										R_v^2	R_v^2	R_v^2	Precisión
	0	1	2	3	4	5	6	7	8	9	max	min	prom	%
1-5	0.82	0.86	0.55	0.56	0.44	0.53	0.50	0.66	0.49	0.40	0.86	0.4	0.58	76.3
1-10	0.84	0.92	0.81	0.74	0.72	0.72	0.85	0.83	0.69	0.61	0.92	0.61	0.77	92.3
1-20	0.91	0.93	0.84	0.81	0.83	0.79	0.87	0.86	0.79	0.77	0.93	0.77	0.84	96.7
1-21	0.89	0.94	0.83	0.82	0.85	0.79	0.87	0.87	0.81	0.78	0.94	0.78	0.84	96.8
1-23	0.91	0.93	0.80	0.83	0.83	0.79	0.88	0.84	0.78	0.74	0.93	0.74	0.83	**97.1**
1-30	0.90	0.92	0.80	0.82	0.80	0.77	0.88	0.83	0.75	0.75	0.92	0.75	0.82	96.8
1-100	0.76	0.79	0.66	0.60	0.62	0.52	0.75	0.70	0.59	0.53	0.79	0.52	0.65	92.7
1-200	0.68	0.72	0.57	0.52	0.56	0.43	0.68	0.61	0.48	0.43	0.72	0.43	0.57	87.2

un máximo de 97.12 % cuando el número de CPs retenidos es de 23. Posteriormente, la precisión decae conforme se retienen más CPs. La evolución de la precisión se aprecia en la Figura 4.13c (línea naranja).

Es interesante observar que después de 200 CP valor de la precisión permanece asintótico hasta llegar a los 784 CPs. La mayor variabilidad de la precisión ocurre en los primeros 100 CPs, por lo tanto, es posible elegir desde el CP 20 hasta el CP 30, para obtener el mayor porcentaje de precisión se opta por seleccionar el 23° CP.

En la Tabla 4.9 se resumen los criterios y su correspondiente número de componentes principales.

Tabla 4.9: Criterios estadísticos y heurísticos para las distintas bases de datos.

Criterio	Base de datos			
	MNIST	Precisión (%)	Candy-Process	MSE $\times 10^3$
Kaiser-Guttman	12	93.97	2	4.1915
Broken-Stick	47	96.12	1	4.0061
PVTA	154	89.32	2	4.1915
Esferacidad de Bartlett	784	83.30	20	74.2638
MDL	774	83.67	10	**1.2837**
AIC	783	83.90	19	37.2446
VRE	264	86.19	14	13.0180
Riguroso (propuesto)	23	**97.12**	10	**1.2837**

Los criterios de evaluación muestran una discrepancia entre cada uno, si tomamos el criterio basado en la búsqueda de mallado como el adecuado para obtener los mejores resultados podemos apreciar que el criterio de Kaiser-Guttman está por debajo del óptimo para ambas bases de datos, el criterio de Broken-Stick está por encima, casi el doble para MNIST y muy por debajo para Candy-Process, el PVT sobre estima el valor de los CPs para MNIST y para Candy-Process no proporciona valores adecuados, encontrándose por debajo del óptimo.

La prueba de Bartlett nos indica que debemos retener todos los CPs dando una mala estimación. Los criterios MDL, AIC y VRE para MNIST muestran que los CPs no triviales son más del 774 dando una sobre estimación del valor óptimo.

El criterio que más se acerca al valor óptimo es el de Kaiser-Guttman, esto pude deberse principalmente al gran número de variables de entrada. Por otro lado, el criterio MDL y VRE también se aproximan al óptimo para Candy-Process, estos criterios pueden ser utilizados como una primera aproximación al valor óptimo siempre y cuando no se cuenten con bastantes variables de entrada. Además, el criterio de AIC sobre estima el número de CPs no triviales para ambos casos.

4.9. Pesos estocásticos vs variables de entrada

En la literatura [42, 53–55] se ha observado que el rango de los pesos estocásticos (ω) se puede establecer de manera a priori antes de llevar a cabo el entrenamiento de la red neuronal. Hemos observado que el número de variables de entrada n_1 afecta manera directa la capa oculta de la red. La función de activación sigmoidal ($\tanh(x)$) que se ha elegido presenta algunas peculiaridades. En cualquier neurona oculta se puede observar lo siguiente:

$$\underline{\underline{y}}^2 = \tanh\left(\left[\underline{1}\ \underline{\underline{X}}\right] \cdot \underline{\underline{W}}^1\right) \in \mathbb{R}^{m \times n_2} \tag{4.4}$$

donde $\underline{\underline{y}}^2$ representa la salida de la capa oculta, sí de esta matriz se elige un vector renglón cualquiera entre $[1, m]$, este vector se puede representar como:

$$\underline{y}_i^2 = \tanh\left(\left[\begin{array}{ccccc} 1 & y_1 & y_2 & ... & y_{n_1} \end{array}\right] \cdot \underline{\underline{W}}^1\right) \in \mathbb{R}^{n_2} \tag{4.5}$$

$$\underline{\widetilde{y}}_i^2 = \left[\begin{array}{ccccc} 1 & y_1 & y_2 & ... & y_{n_1} \end{array}\right] \cdot \underline{\underline{W}}^1 \in \mathbb{R}^{n_2} \tag{4.6}$$

donde $\underline{\widetilde{y}}_i^2$ es el vector renglón antes de aplicar la función de activación.

Para analizar el efecto que tiene el número de variables de entrada n_1 y el rango de inicalización de los pesos estocásticos se presentan dos casos particulares, donde n_1 tiene distintos valores (2 y 500) y para ambos casos se tienen el mismo número de neuronas ocultas $n_2 = 50$.

- Para $n_1 = 2$. La salida de la capa oculta es:

$$\underline{y}^2 = \tanh\left(\left[\begin{array}{ccc} 1 & y_1 & y_2 \end{array}\right] \cdot \underline{\underline{W}}^1\right) \in \mathbb{R}^{50} \tag{4.7}$$

  donde $\underline{\underline{W}}^1 \in \mathbb{R}^{3 \times 50}$. En la Figura 4.14 se grafica la salida de la capa oculta contra los datos antes de aplicar la función de activación, también se varía el rango en el que se genera la matriz $\underline{\underline{W}}^1$, es decir, cambiando el rango ω de los pesos estocásticos.

Se observa que el rango $\omega = [-1, 1]$ presenta con mayor suavidad la función sigmoidal (linea roja), en cambio, para valores grandes o pequeños de ω presentan comportamientos distintos a la función sigmoidal, en el caso de $4.14a$ se presenta un comportamiento lineal y para $4.14b$ se presenta un comportamiento de tipo escalón. Estos comportamientos no son deseados dentro de una RNA. Una de las principales características de las RNAs es que describan comportamientos no lineales, esto no se presenta cuando $\omega = [-0.1, 0.1]$ y $\omega = [-10, 10]$. Posiblemente un rango más adecuado sería entre ± 1 y ± 10, para $n_1 = 2$.

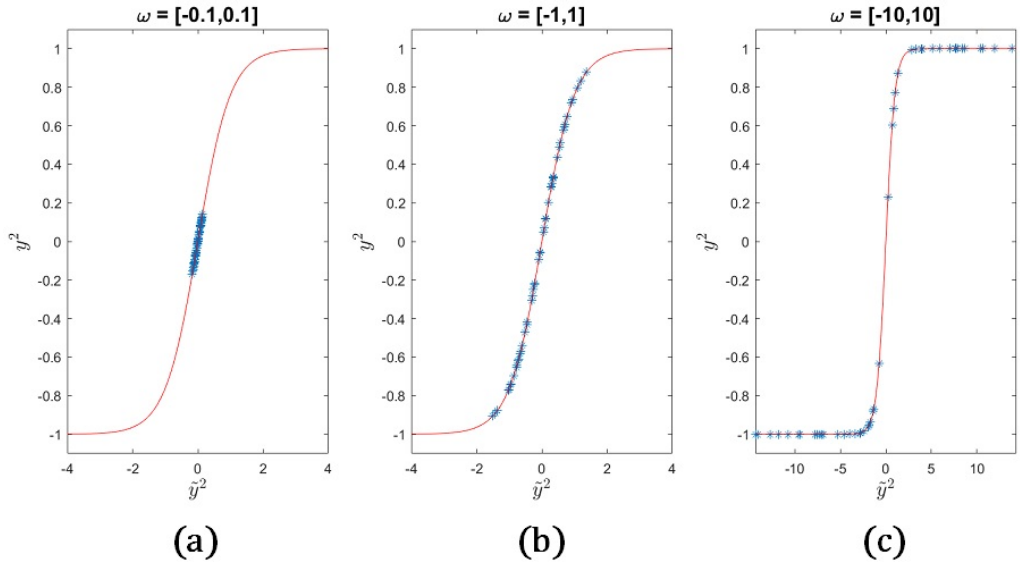


Figura 4.14: Comportamiento de las neuronas ocultas al modificar el rango ω: a) $[\pm 0.1]$, b) $[\pm 1]$ y c) $[\pm 10]$ para $n_1 = 2$ y $n_2 = 50$.

- Para $n_1 = 500$. La salida de la capa oculta es:

$$\underline{y}^2 = \tanh\left(\begin{bmatrix} 1 & y_1 & y_2 & \dots & y_{500} \end{bmatrix} \cdot \underline{\underline{W}}^1\right) \in \mathbb{R}^{50} \tag{4.8}$$

  donde $\underline{\underline{W}}^1 \in \mathbb{R}^{501\times 50}$. En la Figura 4.15 se grafica la salida de la capa oculta contra los datos antes de aplicar la función de activación, de la misma forma que el caso anterior, se varía el rango en el que se genera la matriz $\underline{\underline{W}}^1$, es decir, cambiando el rango ω de los pesos estocásticos.

se observa que el rango $\omega = [-0.1, 0.1]$ presenta con mayor suavidad la función sigmoidal, en cambio para valores de $\omega = [-1, 1]$ y $\omega = [-10, 10]$ se observan comportamientos de tipo escalón, nuevamente, este tipo de comportamiento no se desea dentro de la RNA.

Cuando el valor de n_1 es pequeño, la red se comporta mejor con ω entre ± 1 y ± 10. En cambio, cuando la cantidad de variables de entrada es grande, la red se comporta mejor con ω entre ± 0.1. Por lo tanto, se puede concluir que el número de variables de entrada afecta de manera significativa el comportamiento no lineal de la red, en consecuencia, es necesario establecer una regla o función empírica que nos indique el rango apropiado de ω en función de n_1.

Por el momento hemos encontrado la siguiente relación en función con el número de variables de entrada.

$$r_\omega = \frac{12(\ln(n1) + n1)}{(n_1)^2} \tag{4.9}$$

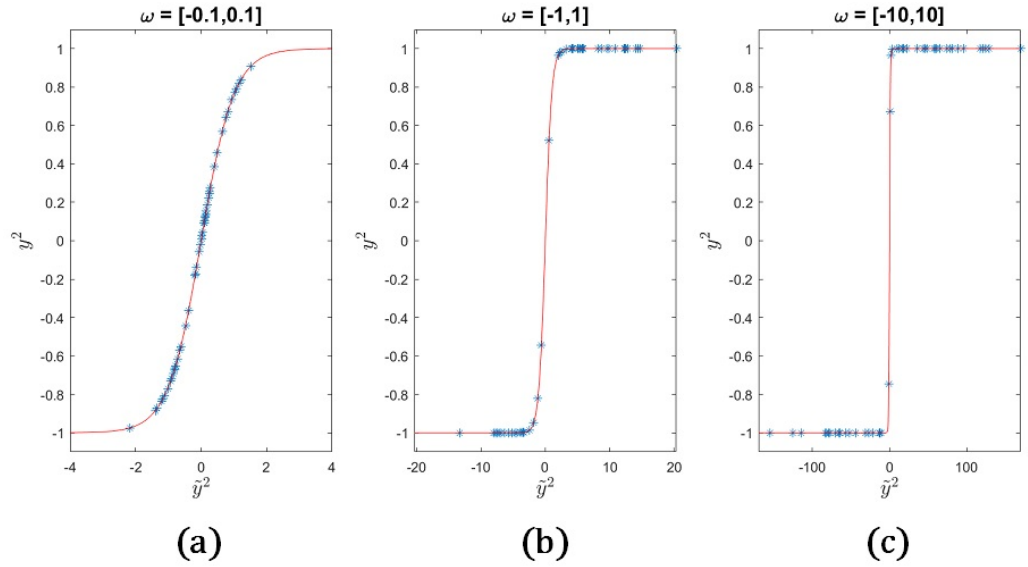


Figura 4.15: Comportamiento de las neuronas ocultas al modificar el rango ω: a) $[\pm 0.1]$, b) $[\pm 1]$ y c) $[\pm 10]$ para $n_1 = 500$ y $n_2 = 50$.

Se debe realizar más estudios referente a esta función con el objetivo de encontrar el rango apropiado de $\omega = [-r_\omega, r_\omega]$ para un determinado número de variables de entrada.

4.10. Procedimiento final para el entrenamiento SOS

El procedimiento SOS propuesto para el entrenamiento y determinación de la arquitectura óptima de una SLFN con estructura $\{n_1, n_2, n_3\}$, es como sigue:

1. Se escalan las variables de los datos experimentales en el rango de $[-1, +1]$.
2. Se asigna un número inicial de neuronas ocultas $n_2 \geq 1$.
3. Se inicializan los parámetros $\underline{\underline{W}}^1$ de manera aleatoria en un rango adecuado, de tal manera que la matriz inversa izquierda de $\left[\underline{1} \;\; \underline{\underline{y}}^2\right]$ se aleje de contener columnas linealmente independientes. De manera natural, los parámetros se inicializan en un rango de $[-1, +1]$, pero eso no garantiza lo anterior, por lo tanto se puede elegir ω entre ± 1 y ± 10. Hemos encontrado que se logran buenos resultados en la independencia de las columnas cuando la amplitud de los pesos se escoge en un rango inversamente proporcional al número de entradas (n_1), Ecuación 4.9, pero esto es solo un resultado preliminar.
4. Se calculan las salidas de las neuronas de la capa oculta del conjunto de entre-

namiento a partir de la Ecuación 3.3

$$\underline{\underline{y}}^2 = \tanh\left(\left[\underline{1} \;\; \underline{\underline{X}}\right] \cdot \underline{\underline{W}}^1\right) \in \mathbb{R}^{m \times n_2}$$

5. Se calculan los parámetros optimizados hacia la última capa del conjunto de entrenamiento a partir de la Ecuación 4.3.

$$\underline{\underline{\tilde{W}}}^2 = \left(\left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \left[\underline{1} \;\; \underline{\underline{y}}^2\right]\right)^{-1} \left[\underline{1} \;\; \underline{\underline{y}}^2\right]^T \cdot \underline{\underline{T}} \in \mathbb{R}^{(n_2+1) \times n_3}$$

6. Se calcula el coeficiente de determinación del conjunto de validación para cada respuesta a partir de la Ecuación 3.8.

$$R_v^2 = 1 - \frac{SSE_v}{SST_v}$$

   La SSE_v se obtiene, cuando se evalúa la RNA con el conjunto de validación, sumando los elementos de la diagonal principal de $\left(\underline{\underline{T}}_v - \underline{\underline{y}}^3\right)^T \cdot \left(\underline{\underline{T}}_v - \underline{\underline{y}}^3\right)$

7. Guardar los pares $\{n_2, R_v^2\}$ de cada una de las respuestas.

8. Cambiar el número de neuronas de la capa oculta, n_2, y repetir el proceso desde el paso 3. El número de neuronas ocultas se puede mover desde 1 hasta $m_e - 1$. El cambio puede hacerse secuencialmente de uno en uno o por bloques o como sea, pero es necesario ir aumentándolo.

9. Buscar el número de neuronas óptimo de la capa oculta, $n_2 = N_v^*$, que corresponda al valor más alto de R_v^2 para cada una de las respuestas.

10. Al terminar de hacer el barrido, si se tiene más de una respuesta, reconstruir la matriz $\underline{\underline{\tilde{W}}}^2$, de manera amputada, tal como se indicó en la sección 4.7.

Cuando se tienen varias capas ocultas, todos los parámetros entre ellas se asignan de manera estocástica y solamente los pesos de la última capa se calculan con la Ecuación 4.3. Se pueden aplicar diferentes estrategias para mover el número de neuronas de cada capa oculta, lo más común es moverlos de igual manera en cada capa.

Capítulo 5

Conclusiones

Finalmente, se ha presentado un algoritmo de entrenamiento con criterios bien definidos para determinar el número óptimo de neuronas en la capa oculta, que resuelve el inconveniente de los distintos mínimos locales y los tiempos largos de entrenamiento, lo que resulta en un incentivo muy grande para el uso de las RNA. Para una gran cantidad de problemas, el método se puede implementar en cualquier ordenador personal, mientras que cuente con la capacidad de procesamiento para evaluar la matriz inversa que aparece en la Ecuación 4.3, la cual se vuelve tediosa cuando se tiene un conjunto de datos muy grande.

El método SOS, comparado con varios métodos iterativos (Sección 4.1), es prácticamente instantáneo, arroja resultados de calidad muy aceptables, y en un solo paso se logran resultados con capacidades de generalización. Se ha encontrado que el número óptimo de neuronas en la capa oculta es igual a aquel en donde se tiene un máximo en R_v^2, para una respuesta de salida en particular (Sección 4.2). En general se ha encontrado que cualquier número de neuronas ocultas, entre $N_{adj,v}^*$ y $N_{adj,e}^*$ da buenos resultados (Sección 4.3). Y aun cuando el número de neurona se encuentre por encima o por debajo del número óptimo requerido, el método SOS siempre tiende a generalizar (filtra el ruido).

El número de neuronas en la capa oculta puede buscarse haciendo un barrido desde 1 hasta el número de datos menos uno con incrementos de uno o mayores de uno (Sección 4.4).

Se estudiaron los distintos factores que afectan el rendimiento del método SOS, se encontró que el rango de los pesos estocásticos tiene un impacto directo al rendimiento de la red, en cuestión de independencia lineal y calidad de la predicción (Sección 4.5). Con el fin evitar la dependencia lineal de los vectores columna en la matriz respuesta de la capa oculta, se recomienda asignar los pesos aleatorios hacia la capa oculta en un rango de amplitud inversamente proporcional al número de entradas en la red,

Ecuación 4.9, es importante mencionar que es necesario analizar a detalle la relación que existe entre las variables de entrada (n_1) y ω.

Para problemas con múltiples respuestas, el barrido va proporcionando el número óptimo de neuronas requeridas por cada respuesta (de acuerdo con su complejidad estructural), generando una matriz amputada de pesos hacia las respuestas, sin detrimento de la rapidez del método (Sección 4.7).

El método SOS arroja un número óptimo de neuronas, en la capa oculta, mayor que aquel que se encontraría con un método iterativo tradicional. Sin embargo, con el método iterativo no es práctico implementar una búsqueda por barrido. El número excesivo de neuronas en la capa oculta provoca un corrugado en la calidad de la predicción y es independiente del método de entrenamiento. Este efecto de corrugado, por exceso de neuronas ocultas, no se había mostrado en trabajos anteriores de RNA.

Como se demostró a lo largo de este trabajo, el método de entrenamiento SOS es rápido en comparación con métodos de entrenamiento tradicionales (métodos iterativos). A pesar de que hoy en día existan métodos semejantes de entrenamiento (ELM, máquina de aprendizaje extremo), la propuesta de este trabajo intenta mejorar y reconstruir el concepto de muchas de estas técnicas en especial cuando se modelan sistemas MIMO. Como se observó en la sección de resultados es interesante observar que cada respuesta necesita su propia topología, este enfoque no se ha presentado antes debido a los conflictos durante el entrenamiento tradicional. A pesar de que ELM tiene bastantes variantes no se ha reportado el monitoreo de cada respuesta y mucho menos el uso de una matriz de parámetros amputada como se presenta en este trabajo.

Por lo tanto, utilizar el método SOS con una matriz amputada puede generar resultados competentes con otras técnicas, además, se ha comprobado que es sin duda un método de entrenamiento rápido y determina la mejor estructura de la RNA, esto garantiza que el usuario no deba intervenir durante el entrenamiento de la red.

Además, la correcta selección de variables de entrada no es un tema que debe tomarse a la ligera, en especial cuando la base de datos cuenta con bastantes variables, hemos observado que el método riguroso proporciona un número adecuado de CPs de tal forma que los resultados obtenidos de su modelación son los mejores. Podemos observar que algunos criterios descritos en la literatura no proporcionan la correcta información de cómo debemos elegir el número de CPs, aunque en el caso de Candy-Process los criterios VRE y MDL presentaron valores cercanos al óptimo, en cambio para la base de datos MNIST los criterios de Kaiser-Guttman y Broken-Stick estuvieron más cerca que los demás, aunque para la base de datos Candy-Process esto no fue igual, dando valores muy por debajo del óptimo. Esto nos indica que el número de variables de entrada afecta directamente a estos criterios y su correcta selección.

En la practica el realizar una búsqueda de mallado suele ser agotador si no se cuenta

con las herramientas adecuadas para este análisis, es por ello que la implementación del método SOS favorece con creces este tipo de análisis. Por lo tanto, el realizar este sondeo de CPs favorece significativamente hacia la obtención de un modelo eficiente que proporcione resultados confiables.

Apéndice A

Diseño Experimental

Para observar el rendimiento del método SOS se implementó un diseño experimental con el objetivo de analizar factores que pueden afectar o mejor el desempeño del método propuesto.

Se obtuvo como resultado un diseño experimental que generó 67 puntos. La descripción completa se muestra en la Tabla A.1.

Tabla A.1: Plan experimental para el método SOS.

RUN	% Entre	% Valid	ND	Rango pesos	% Ruido	Sup	GW
1	70	30	200	1	5	Gauss	Columna
2	70	30	200	1	30	Gauss	Columna
3	70	30	200	-1	30	CH	Matriz
4	50	50	1000	-1	17.5	Gauss	Columna
5	90	10	200	-1	17.5	Gauss	Matriz
6	50	50	1000	-1	30	CH	Columna
7	90	10	1000	-1	5	CH	Matriz
8	50	50	200	1	17.5	Gauss	Columna
9	70	30	600	0.5	23.75	CH	Columna
10	70	30	600	-1	30	CH	Columna
11	90	10	1000	-1	5	Gauss	Matriz
12	50	50	1000	-1	5	CH	Matriz
13	70	30	200	0	30	Gauss	Matriz
14	50	50	200	-1	30	Gauss	Matriz
15	50	50	600	1	5	Gauss	Matriz
16	60	40	1000	1	30	Gauss	Columna
17	70	30	1000	1	30	CH	Matriz

18	90	10	600	1	5	CH	Columna
19	90	10	200	-1	5	CH	Matriz
20	90	10	600	-1	17.5	Gauss	Columna
21	50	50	600	0	17.5	Gauss	Columna
22	90	10	1000	1	5	CH	Matriz
23	70	30	600	-1	5	Gauss	Matriz
24	70	30	1000	-1	17.5	CH	Matriz
25	50	50	600	1	17.5	CH	Matriz
26	70	30	600	-1	5	Gauss	Matriz
27	50	50	1000	-1	30	Gauss	Matriz
28	50	50	800	0.5	11.25	Gauss	Columna
29	90	10	1000	1	17.5	CH	Columna
30	90	10	1000	0	30	Gauss	Columna
31	90	10	1000	1	5	Gauss	Columna
32	50	50	200	0	5	CH	Matriz
33	50	50	200	-1	30	Gauss	Matriz
34	70	30	200	-1	17.5	Gauss	Matriz
35	50	50	200	-1	5	Gauss	Columna
36	50	50	600	1	5	CH	Columna
37	50	50	1000	1	30	Gauss	Columna
38	50	50	600	1	30	Gauss	Matriz
39	90	10	1000	-1	30	CH	Columna
40	90	10	200	0	30	CH	Columna
41	50	50	200	1	17.5	Gauss	Columna
42	80	20	200	0	17.5	CH	Matriz
43	70	30	200	1	30	Gauss	Columna
44	90	10	200	1	5	Gauss	Matriz
45	50	50	200	-1	5	Gauss	Matriz
46	90	10	1000	1	30	CH	Matriz
47	50	50	600	-1	17.5	CH	Matriz
48	90	10	200	0	5	Gauss	Columna
49	90	10	600	-1	30	CH	Matriz
50	70	30	1000	1	5	CH	Matriz
51	50	50	1000	1	30	Gauss	Columna
52	90	10	200	-1	17.5	Gauss	Columna
53	70	30	200	1	5	CH	Matriz

54	90	10	600	-0.5	11.25	CH	Columna
55	50	50	400	0	17.5	CH	Columna
56	90	10	200	1	17.5	CH	Columna
57	90	10	200	1	30	CH	Matriz
58	50	50	1000	0	17.5	CH	Matriz
59	60	40	600	-1	30	Gauss	Columna
60	60	40	800	-0.5	11.25	CH	Columna
61	90	10	800	0.5	17.5	Gauss	Matriz
62	70	30	200	-1	5	CH	Columna
63	70	30	1000	0	30	CH	Columna
64	70	30	1000	-1	5	Gauss	Matriz
65	50	50	200	1	30	CH	Matriz
66	90	10	600	1	30	CH	Columna
67	80	20	1000	-1	30	Gauss	Matriz

Bibliografía

[1] L. Leonardo Hurtado, "Modelamiento teórico y modelamiento empírico de procesos, una síntesis," *Scientia et Technica*, vol. XII, no. 31, pp. 103–108, 2006.

[2] D. R. Baughman and Y. A. Liu, *Neural Networks in Bioprocessing and Chemical Engineering*. Elsevier, jul 1995.

[3] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering*. Cambridge University Press, jan 2019.

[4] M. Fernandez, H. Barron, and A. S. Barnard, "Artificial neural network analysis of the catalytic efficiency of platinum nanoparticles," *RSC Adv.*, vol. 7, no. 77, pp. 48962–48971, 2017.

[5] B. R. Goldsmith, J. Esterhuizen, J.-X. Liu, C. J. Bartel, and C. Sutton, "Machine learning for heterogeneous catalyst design and discovery," *AIChE Journal*, vol. 64, pp. 2311–2323, jul 2018.

[6] R. González-García, R. Rico-Martinez, W. Wolf, M. Lübke, and M. Eiswirth, "Caracterización de Sistemas No-Lineales Mediante Redes Neuronales: Estudio de una Reacción Electroquímica," *Ingenieria Quimica*, vol. 30, no. 345, pp. 173–184, 1998.

[7] G. L. Gutiérrez Urueta, D. Colorado Garrido, J. A. Hernandez, P. A. Rodriguez Aumente, and W. Rivera Gomez Franco, "Performance estimation and optimization of an adiabatic H2 O-Libr absorption system using artificial neural networks," *Ingeniería Investigación y Tecnología*, vol. 20, pp. 1–13, jan 2019.

[8] R. A. González-Grimaldo and J. C. Cuevas-Tello, "Analysis of Time Series with Artificial Neural Networks," in *2008 Seventh Mexican International Conference on Artificial Intelligence*, pp. 131–137, IEEE, oct 2008.

[9] E. Herrera-Hernández, R. Ocampo-Perez, C. Aguilar-Madera, and J. Flores-Cano, "Adsorption-diffusion model with neural network-based equilibrium relationship,"

DESALINATION AND WATER TREATMENT, vol. 132, no. March 2019, pp. 42–51, 2018.

[10] D. M. Himmelblau, "Applications of artificial neural networks in chemical engineering," *Korean Journal of Chemical Engineering*, vol. 17, pp. 373–392, jul 2000.

[11] R. González-Garcia, R. Rico-Martinez, W. Wolf, M. Lübke, M. Eiswirth, J. Anderson, and I. Kevrekidis, "Characterization of a two-parameter mixed-mode electrochemical behavior regime using neural networks," *Physica D: Nonlinear Phenomena*, vol. 151, pp. 27–43, apr 2001.

[12] R. Gonzalez-Garcia, R. Rico-Martinez, and I. Kevrekidis, "Identification of distributed parameter systems: A neural net based approach," *Computers and Chemical Engineering*, vol. 22, pp. S965–S968, mar 1998.

[13] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, dec 1943.

[14] B. Widrow and M. E. Hoff, "Associative storage and retrieval of digital information in networks of adaptive "neurons".," *Biological Prototypes and Synthetic Systems.*, pp. 160–160, 1962.

[15] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," tech. rep., US Dept of the Navy, sep 1985.

[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks.," *Science (New York, N.Y.)*, vol. 313, pp. 504–7, jul 2006.

[18] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, jan 2015.

[19] M. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, pp. 989–993, apr 1994.

[20] S. He, N. Sepehri, and R. Unbehauen, "Modifying Weights Layer-By-Layer with Levenberg-Marquardt Backpropagation Algorithm," *Intelligent Automation & Soft Computing*, vol. 7, pp. 233–247, jan 2001.

[21] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, oct 1986.

[23] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, pp. 559–572, nov 1901.

[24] K. A. Yeomans and P. A. Golder, "The Guttman-Kaiser Criterion as a Predictor of the Number of Common Factors," *The Statistician*, vol. 31, p. 221, sep 1982.

[25] D. A. Jackson, "Stopping Rules in Principal Components Analysis: A Comparison of Heuristical and Statistical Approaches," *Ecology*, vol. 74, pp. 2204–2214, dec 1993.

[26] S. Valle, W. Li, and S. J. Qin, "Selection of the Number of Principal Components: The Variance of the Reconstruction Error Criterion with a Comparison to Other Methods," *Industrial & Engineering Chemistry Research*, vol. 38, pp. 4389–4401, nov 1999.

[27] M. Wax and T. Kallath, "Detection of signals by information theoretic criteria," *Adaptive Antennas for Wireless Communications*, pp. 184–189, 2009.

[28] S. J. Qin and R. Dunia, "Determining the number of principal components for best reconstruction," *Journal of Process Control*, vol. 10, no. 2, pp. 245–250, 2000.

[29] M. J. Masnan, A. Zakaria, A. Y. Shakaff, N. I. Mahat, H. Hamid, and N. Subari, "Principal Component Analysis - A Realization of Classification Success in Multi Sensor Data Fusion," 2007.

[30] B. Erkmen and T. Yildirim, "Improving classification performance of sonar targets by applying general regression neural network with PCA," *Expert Systems with Applications*, vol. 35, no. 1-2, pp. 472–475, 2008.

[31] M. Wagih, F. Abou-Chadi, H. El-Din, and N. Mekky, "Classification of Ultrasound Kidney Images using PCA and Neural Networks," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 4, pp. 53–57, 2015.

[32] R. Milewski, D. Jankowska, U. Cwalina, A. J. Milewska, D. Citko, T. Więsak, A. Morgan, and S. Wołczyński, "Application of Artificial Neural Networks and Principal Component Analysis to Predict Results of Infertility Treatment Using the IVF Method," *Studies in Logic, Grammar and Rhetoric*, vol. 47, pp. 33–46, dec 2016.

[33] A. Mostaar, M. R. Sattari, S. Hosseini, and M. R. Deevband, "Use of artificial neural networks and pca to predict results of infertility treatment in the icsi method," *Journal of Biomedical Physics and Engineering*, vol. 9, no. 6, pp. 679–686, 2019.

[34] J. Zhang, "Machine learning with feature selection using principal component analysis for malware detection: A case study," *arXiv*, 2019.

[35] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, dec 1989.

[36] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[37] W. Schmidt, M. Kraaijveld, and R. Duin, "Feedforward neural networks with random weights," in *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, pp. 1–4, IEEE Comput. Soc. Press, 1992.

[38] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, pp. 985–990, IEEE, 2004.

[39] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, pp. 163–180, apr 1994.

[40] P. G. Anderson, M. Ge, S. Raghavendra, M.-L. Lung, and R. S. Gaborski, "Using Quasirandom Weights in Neural Networks," *Intelligent Automation & Soft Computing*, vol. 4, pp. 61–71, jan 1998.

[41] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, dec 2006.

[42] W. Wang and X. Liu, "The selection of input weights of extreme learning machine: A sample structure preserving point of view," *Neurocomputing*, vol. 261, pp. 28–36, oct 2017.

[43] C. K. Chui, X. Li, and H. N. Mhaskar, "Limitations of the approximation capabilities of neural networks with one hidden layer," *Advances in Computational Mathematics*, vol. 5, pp. 233–243, dec 1996.

[44] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, pp. 76–79, may 1992.

[45] E. Wong, "Stochastic neural networks," *Algorithmica*, vol. 6, pp. 466–478, jun 1991.

[46] B. Igelnik and Yoh-Han Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1320–1329, 1995.

[47] Minghu Jiang, Beixing Deng, Bin Wang, and Bo Zhang, "A fast learning algorithm of neural networks by changing error functions," in *International Conference on Neural Networks and Signal Processing, 2003. Proceedings of the 2003*, vol. 1, pp. 249–252 Vol.1, IEEE, 2003.

[48] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, pp. 985–990, IEEE, 2004.

[49] M. Georgiopoulos, C. Li, and T. Kocak, "Learning in the feed-forward random neural network: A critical review," *Performance Evaluation*, vol. 68, pp. 361–384, apr 2011.

[50] R. Zhang, Y. Lan, G.-b. Huang, and Y. C. Soh, "Extreme Learning Machine with Adaptive Growth of Hidden Nodes and Incremental Updating of Output Weights," in *Proceedings of the Second International Conference on Autonomous and Intelligent Systems*, vol. 08, pp. 253–262, Burnaby, BC, Canada: Springer, Berlin, Heidelberg, 2011.

[51] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, pp. 107–122, jun 2011.

[52] Y. LeCun, C. Cortes, and C. J. Burges, ""MNIST handwritten digit database".."

[53] D. Wang and M. Li, "Stochastic Configuration Networks: Fundamentals and Algorithms," *IEEE Transactions on Cybernetics*, vol. 47, pp. 3466–3479, oct 2017.

[54] D. Wang and C. Cui, "Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics," *Information Sciences*, vol. 417, pp. 55–71, nov 2017.

[55] M. Li and D. Wang, "Insights into randomized algorithms for neural networks: Practical issues and common pitfalls," *Information Sciences*, vol. 382-383, pp. 170–178, mar 2017.