

UCUENCA

Facultad de Ingeniería

Maestría en Electricidad Mención Redes Eléctricas Inteligentes

Evaluación de un algoritmo basado en Machine Learning para un flujo de potencia óptima de corriente alterna ACOPF

Trabajo de titulación previo a la obtención del título de Magíster en Electricidad Mención Redes Eléctricas Inteligentes

Autor:

Ing. Walter Ramiro Astudillo Astudillo

CI: 0105356810

Correo electrónico: waltheraastudillo@outlook.com

Director:

Ing. Darwin Fabián Astudillo Salinas, PhD

CI: 0103907036

Cuenca, Ecuador

18 de octubre de 2022

Resumen

En este trabajo, se analiza la factibilidad de usar el aprendizaje automático (*Machine Learning (ML)*) para obtener soluciones al problema del flujo de potencia óptimo de corriente alterna (*Alternating Current Optimal Power Flow (ACOPF)*). Debido a que *ACOPF* es un problema no convexo y con una alta no linealidad, se han realizado numerosos esfuerzos para encontrar métodos eficientes de optimización que puedan reducir sustancialmente los tiempos de resolución. Los problemas de *Optimal Power Flow (OPF)* generalmente se resuelven mediante métodos de punto interior [1], también conocidos como métodos de barrera. Uno de los enfoques más utilizados es la técnica del punto interior dual primario con una búsqueda de línea de filtro [2]. Estos métodos son robustos pero costosos, ya que requieren el cálculo de la segunda derivada del Lagrangiano en cada iteración. Una nueva y fructífera dirección de investigación consiste en utilizar técnicas de *ML* para resolver problemas de operación y control de las redes eléctricas. *ML* ha mostrado reducir significativamente el uso de recursos computacionales en muchos problemas del mundo real. Se han utilizado varios métodos de solución entre ellos se destacan bosque aleatorio, árbol de decisiones de objetivos múltiples y máquina de aprendizaje extrema [3, 4]. El funcionamiento de *ML* en este caso se aplica como un método que predice primero magnitudes y ángulos de voltaje en cada bus. Empleando ecuaciones de red basadas en la física para calcular la inyección de potencia en diferentes buses. Para el aprendizaje en general de *ML* los datos se dividen en tres conjuntos: uno de entrenamiento, otro de validación y finalmente, uno de pruebas. Estos algoritmos se centran en minimizar su función objetivo y el costo de operación de una red de transmisión de corriente alterna.

Palabras clave : Aprendizaje automático. Flujo de potencia óptimo. Flujo de potencia óptimo de corriente alterna.

Abstract

In this work, the feasibility of using machine learning (ML) to obtain solutions to the alternating current optimal power flow problem ACOPF (Alternating Current Optimal Power Flow) is analyzed. Because ACOPF is a nonconvex problem with high nonlinearity, numerous efforts have been made to find efficient optimization methods that can substantially reduce resolution times. OPF (Optimal Power Flow) problems are usually solved by interior point methods [1], also known as barrier methods. One of the most widely used approaches is the primary dual interior point technique with a filter line search [2]. These methods are robust but expensive, since they require the calculation of the second derivative of the Lagrangian in each iteration. A new and fruitful research direction is to use ML techniques to solve problems of operation and control of electrical networks. ML has been shown to significantly reduce the use of computational resources in many real-world problems. Several solution methods have been used, among them random forest, multi-objective decision tree and extreme learning machine [3, 4]. The ML operation in this case is applied as a method that first predicts voltage magnitudes and angles on each bus. Using network equations based on physics to calculate the injection of power in different buses. For general ML learning, the data is divided into three sets: one for training, one for validation, and finally, one for testing. These algorithms focus on minimizing their objective function and the cost of operating an AC transmission network.

Keywords: Machine learning. Optimal power flow. Alternating current optimal power flow.

Índice general

Resumen	II
Abstract	III
Índice general	IV
Índice de figuras	VI
Índice de tablas	VII
Cláusula de Propiedad Intelectual	VIII
Cláusula de licencia y autorización para publicación en el Repositorio Institucional	IX
Certifico	X
Dedicatoria	XI
Agradecimientos	XII
Abreviaciones y acrónimos	1
1. Introducción	1
1.1. Descripción del problema	2
1.2. Alcance	2
1.3. Objetivos	3
1.3.1. Objetivo general	3
1.3.2. Objetivo específico	3
1.4. Motivación y objetivo del trabajo	3
2. Marco teórico	4
2.1. Flujo óptimo de potencia	4
2.2. Redes neuronales artificiales	6
2.2.1. Clasificación de redes neuronales artificiales	7
2.2.1.1. Redes de propagación hacia Delante	7
2.2.1.2. Redes de propagación hacia atrás	7
2.3. Principales tipos de entrenamiento para redes neuronales	8
2.4. Aprendizaje automático	10

2.4.1.	Modelos de aprendizaje automático	11
2.4.1.1.	Modelo de bosque aleatorio <i>Random Forest</i> (RF)	11
2.4.1.2.	Árbol de decisiones de objetivos múltiple <i>Multi-Target Decision Tree</i> (DT)	12
2.4.1.3.	Máquina de aprendizaje extrema <i>Extreme Learning Machine</i> (ELM)	12
2.5.	Teoría de aprendizaje de ELM	13
2.5.1.	Máquina de aprendizaje extremo ELM	14
2.6.	Arquitectura y funcionamiento de ELM	15
2.6.1.	Matriz pseudoinversa de Moore-Penrose	15
2.7.	Función de activación	16
3.	Trabajos relacionados	19
3.1.	Modelo de bosque aleatorio RF	20
3.2.	Árbol de decisiones de objetivos múltiples DT	20
3.3.	Máquina de aprendizaje extrema ELM	21
4.	Diseño e implementación	23
4.1.	Metodología OPF con aprendizaje automático	23
4.2.	Selección del algoritmo de machine learning	24
4.3.	Generación de conjuntos de datos de entrada	25
4.4.	Implementación del algoritmo ELM en ACOPF	26
4.5.	Análisis de datos de salida	27
5.	Resultados y discusión	28
5.1.	Rendimiento de ELM	28
5.2.	Evaluación de ELM sobre ACOPF	30
5.3.	Comparación de tiempo computacional	32
6.	Conclusiones	34
7.	A1	36
7.1.	Código de Matlab	36
8.	A2	37
8.1.	Gráficas y Resultados	37
	Bibliografía	41

Índice de figuras

2.1. Esquema de una red neuronal artificial y una biológica.	6
2.2. Red Neuronal de Tipo feedforward	7
2.3. Red Monocapa (izquierda), Red multicapa (derecha)	8
2.4. Red de tipo <i>feedforward</i> multicapa.	9
2.5. Red de tipo feedforward monocapa.	10
2.6. Sistema poco entrenado, correctamente entrenado y sobreentrenado [28].	13
2.7. Arquitectura utilizada de ELM [28]	16
2.8. Funciones de activación [16]	18
4.1. Esquema de Metodología OPF con aprendizaje automático ML	23
4.2. Generación de conjuntos de datos de entrada	26
4.3. Implementación del algoritmo ELM	27
4.4. Análisis de datos de salida del algoritmo ELM	27
5.1. Error absoluto case300	29
5.2. Representación gráfica de valores de voltaje y ángulo con ruido correspondiente al Escenario 1 y 2.	30
8.1. Error absoluto case500	38
8.2. Error absoluto case3375wp	39
8.3. Iteraciones de Voltaje predico Vs Voltaje real	40

Índice de tablas

4.1. Comparación principal entre RF y DT , ELM	25
4.2. Características de redes de prueba seleccionadas	26
5.1. Errores de prueba y entrenamiento del case300	30
5.2. Detalle del rendimiento estadístico para la predicción de potencia activa y reactiva, y magnitud del voltaje y ángulo.	31
5.3. Detalle del rendimiento estadístico para la predicción de potencia activa y reactiva, y magnitud del voltaje y ángulo con ruido del 5 %.	31
5.4. Costo de producción de la función objetivo del flujo óptimo	32
5.5. Comparación de tiempo computacional entre el solucionador OPF con aprendizaje aumentado y Matpower Interior Point Solver (MIPS)	33
5.6. Resultados de optimización de los enfoques OPF de ELM	33
8.1. Errores de prueba y entrenamiento del case500	37
8.2. Errores de prueba y entrenamiento del case3375wp	37

Cláusula de Propiedad Intelectual

Yo, Walter Ramiro Astudillo Astudillo, autor de la tesis "Evaluación de un algoritmo basado en Machine Learning para un flujo de potencia óptima de corriente alterna ACOPF", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 21 de octubre de 2022



Walter Ramiro Astudillo Astudillo

010535681-0

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Walter Ramiro Astudillo Astudillo en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Evaluación de un algoritmo basado en Machine Learning para un flujo de potencia óptima de corriente alterna ACOF", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 21 de octubre de 2022



Walter Ramiro Astudillo Astudillo

010535681-0

Certifico

Que el presente proyecto de tesis: Evaluación de un algoritmo basado en Machine Learning para un flujo de potencia óptima de corriente alterna ACOFF, fue dirigido y revisado por mi persona.



Ing. Darwin Fabián Astudillo Salinas, Ph.D.
Director

Dedicatoria

A Dios por darme salud y fuerzas en salir adelante en todo momento, por haberme dado la vida y permitir el haber llegado hasta este momento tan importante de mi formación académica.

A mis padres Ramiro y Ligia quienes han sido un apoyo incondicional que sin su ayuda nada de esto sería posible, y mis hermanos Catalina, Eduardo y Byron que siempre confiaron en mí en todo momento.

A mi esposa Ana Cristina, por estar a mi lado y darme ánimos en salir adelante.

A mis suegros Hugo y Sara por el apoyo a lo largo de este trabajo, y a toda su familia.

A familiares y amigos cuyos consejos siempre fueron para continuar y no ceder cuando se presentaba alguna dificultad.

Walter Ramiro Astudillo Astudillo

Agradecimientos

Agradecerle a mi Madre Ligia Lucila quien siempre ha estado a mi lado apoyándome, dándome fuerza y ánimos para seguir en los momentos difíciles, con quien siempre he podido contar y quien me ha demostrado todo su amor incondicional.

A la Universidad de Cuenca y todos los profesores que fueron parte de mi preparación universitaria, en especial al Ing. Fabián Astudillo, Phd por su apoyo y ayuda a lo largo de este trabajo.

A mi Ecuador...

Walter Ramiro Astudillo Astudillo

Abreviaciones y Acrónimos

ACOPF *Alternating Current Optimal Power Flow*. 1–4, 11, 19, 20, 23, 25, 26, 28, 30, 32

CA Corriente Alterna. 4, 5, 34

CC Corriente Continua. 1

DNN Deep Neural Networks. 14, 15

DT *Multi-Target Decision Tree*. 2, 12, 19–21, 24, 25, 28, 34

ELM *Extreme Learning Machine*. 2, 10, 12–16, 19, 21, 22, 24–35, 38

IEEE Institute of Electrical and Electronics Engineers. 2, 25, 28, 29, 32, 34

LARS Least Angle Regression. 26

MAE Error medio absoluto. 28, 29, 31, 37

MIMO *Multiple input multiple output*. 19

MIPS Matpower Interior Point Solver. 23, 24, 26, 27, 31–33

ML *Machine Learning*. 1–3, 5, 11, 12, 14, 21, 23, 24, 26, 28, 30, 32, 34

MLP *Multi-Layer Perceptron*. 11, 12, 27

OPF *Optimal Power Flow*. 1–4, 20–26, 32–34

RBF Redes de Funciones de Base Radial. 9, 13

RF *Random Forest*. 2, 11, 19, 20, 24, 25, 28, 34

RMSE Raíz del Error Cuadrático Medio. 28, 29, 31

RNA Redes Neuronales Artificiales. 11

RSS Suma residual de cuadrados. 12, 20

SLFN *Single Layer Feedforward Neural network*. 2, 9, 10, 13–15, 19, 21, 25

SVM Máquinas de vectores de soporte. 8, 19

Introducción

El problema de flujo óptimo de potencia (OPF por sus siglas en inglés) consiste en determinar los niveles óptimos de operación para diferentes generadores dentro de una red de transmisión a fin de satisfacer la demanda que va cambiando en el espacio y el tiempo [5]. En la actualidad los operadores del sistema requieren tomar decisiones precisas de despacho y mercado para utilizar mejor los recursos disponibles [5]. Los operadores del sistema ejecutan algoritmos de OPF para obtener las decisiones de despacho en un menor tiempo, por ejemplo en California se ejecuta cada 5 minutos [2]. Debido a la naturaleza incierta y estocástica de los recursos renovables, los parámetros de decisión de OPF deben determinarse con prontitud para evitar inconvenientes al momento de tomar alguna decisión futura. Sin embargo, ACOPF sigue siendo un problema desafiante; debido a su no linealidad. Se han propuesto muchos enfoques para resolverlo y se han implementado métodos como: la relajación lineal [3], la relajación convexa [6], los algoritmos heurísticos [7] y el uso de aprendizaje automático ML [4]. OPF con la implementación de ML es un tema de investigación establecida tanto en sistemas de energía como en operaciones; OPF se aplica en la gestión y regulación de redes eléctricas. En este trabajo, se espera obtener soluciones aproximadas en tiempo real al problema de OPF utilizando ML.

ML ha mostrado mejorar significativamente la eficiencia computacional de muchos problemas complejos, incluido el OPF. La implementación de enfoques ML para resolver OPF se puede dividir en dos categorías: (1) enfoques de un extremo a otro que se basan puramente en ML, (2) enfoques híbridos de ML y soluciones basados en la física. En la literatura se han desarrollado varios métodos de ML de extremo a extremo para resolver la OPF. Por ejemplo, se adoptó una colección de algoritmos de aprendizaje supervisado para resolver solo el valor de la función objetivo de OPF; entre los algoritmos se encuentran Bayes ingenuo de Gauss, regresión logística, árbol de decisión, bosque aleatorio [8]. En este trabajo, se busca verificar la factibilidad del uso de ML para obtener soluciones aproximadas al problema de OPF. La formulación clásica de ACOPF es un problema desafiante no convexo; además de minimizar los costos del generador, las soluciones deben cumplir con las leyes físicas que rigen el flujo de energía y cumplir con la ley de voltaje de Kirchhoff respetando los límites de ingeniería de la red [9, 10]. Otro método que brinda una solución viable es aproximar el sistema como un flujo de energía de Corriente Continua (CC). Esto crea un problema de optimización lineal que es más fácil de resolver y computacionalmente más eficiente, pero las soluciones de este método pueden ser subóptimas e inducir pérdidas financieras sustanciales o pueden fallar en satisfacer las limitaciones físicas y de ingeniería [11].

El uso de ML presenta varias ventajas. Las redes neuronales han mostrado la capacidad de modelar funciones

no convexas extremadamente complicadas, lo que las hace atractivas para este entorno [11]. Un modelo podría entrenarse fuera de línea con datos históricos y usarse en tiempo real para hacer predicciones en una configuración de energía óptima. Esto es beneficioso tanto para las empresas de servicios públicos como para los investigadores en el campo [2]. Este tema se ha trabajado poco, ya que el reciente aumento de la penetración de las energías renovables está comenzando a hacer que esta estrategia sea atractiva. En la literatura el uso de ML ha mostrado que los algoritmos de vecinos más cercanos y basados en árboles funcionan bien en el mapeo de múltiples objetivos, pero los métodos de los vecinos más cercanos son notoriamente lentos e inadecuados. Por lo tanto, se estudian dos algoritmos ML basados en árboles para realizar la tarea de aprendizaje, que son: RF y DT [11]. También se adopta una red neuronal de retroalimentación *Single Layer Feedforward Neural network* (SLFN) de una sola capa extremadamente rápida, llamada ELM. En [12] se demostró que ELM es capaz de reducir significativamente el tiempo de entrenamiento y predicción, en comparación con los algoritmos clásicos de ML.

1.1. Descripción del problema

El cálculo del flujo de potencia óptimo para sistemas pequeños, medianos y grandes juega un papel importante en la optimización, el cual busca minimizar el costo de operación de una red de transmisión [4]. El principal problema al que se enfrenta este tipo de optimización es la cantidad de tiempo que se necesita para realizar el cálculo y por ende los recursos computacionales hasta conseguir un resultado que se ajuste a la necesidad [1]. El principal inconveniente de usar ML para obtener soluciones directas de un sistema de extremo a extremo es que puede existir una gran cantidad de violaciones con referencia a restricciones físicas relacionadas con la ley de Kirchhoff [4, 6, 11]. Esto se debe al hecho de que tanto la magnitud del voltaje como el ángulo determinan la cantidad de flujo de energía en las ramas, y las pérdidas de energía de las líneas también dependen de esos parámetros de voltaje. Para afrontar los desafíos de un sistema de extremo a extremo basado en ML, este trabajo evalúa métodos de solución de puntos interiores duales usando la librería MATPOWER de Matlab para resolver ACOPF, que no depende de ningún solucionador de OPF basado en la física. Para abordar los problemas de violación de restricciones relacionadas con la ley física, el modelo de ML propuesto resuelve las magnitudes y ángulos de voltaje en lugar de la generación de energía real o reactiva. La magnitud de la tensión generada y los ángulos no se contradicen entre sí según las leyes físicas, lo que garantiza la viabilidad de las soluciones OPF con ML.

1.2. Alcance

Este trabajo comprende el análisis del uso de ML, para evaluar y optimizar el costo mínimo de producción de una red de transmisión. Para ello se utilizan algoritmos de ML aplicados a los casos de estudio de red *Institute of Electrical and Electronics Engineers* (IEEE) como modelos de prueba para realizar el entrenamiento. Una vez realizado el entrenamiento del algoritmo, se evalúa el tiempo de convergencia y la eficiencia del proceso de entrenamiento del algoritmo ML implementado en un sistema eléctrico.

1.3. Objetivos

1.3.1. Objetivo general

Evaluar y desarrollar un algoritmo basado en *Machine Learning* (ML) para predecir la convergencia y/o el valor de costo de producción de la función objetivo del flujo óptimo.

1.3.2. Objetivo específico

El presente trabajo tiene los siguientes objetivos específicos:

- Revisar el estado del arte de algoritmos de *Machine Learning* (ML) y seleccionar el más adecuado para aplicar al ACOPF.
- Definir y caracterizar las variables e indicadores de entrada y salida del sistema para verificar su convergencia en base a los casos de la IEEE.
- Implementar el algoritmo de ML para el uso de la ACOPF.
- Realizar una comparación del algoritmo seleccionado de *Machine Learning* (ML) con respecto al algoritmo ACOPF implementado en MATPOWER.

1.4. Motivación y objetivo del trabajo

El principal inconveniente de usar ML para obtener soluciones directas de un extremo a otro en un sistema eléctrico es que puede existir una gran cantidad de violaciones físicas relacionadas con la ley de Kirchhoff [4, 6, 11]. Para abordar los desafíos en los sistemas eléctricos y dar solución de un extremo a extremo basados en ML, este documento desarrolla un enfoque de ML para resolver ACOPF, que no depende de ningún solucionador de OPF basado en la física [4]. Para afrontar los problemas de violación de restricciones relacionadas con la ley física, el modelo propuesto busca dar una solución implementando ML a una red eléctrica mediante la manipulación de las magnitudes y ángulos de voltaje en lugar de la generación de energía real o reactiva ya que estas no se contradicen entre sí según las leyes físicas, lo que garantiza la viabilidad de las soluciones OPF [4, 9, 13, 14]. Ahora para entrenar y evaluar la red del algoritmo propuesto se obtienen los parámetros de entrada y se normalizan para aplicar ecuaciones de flujo de potencia basadas en la física para calcular otras variables operativas. Se espera que el enfoque propuesto reduzca la dimensión de mapeo de entrada-salida para el algoritmo ML y así disminuir el gasto computacional y cumplir en los resultados con la referencia a la ley física [9, 15, 16]. El documento está organizado de la siguiente manera. El capítulo 2, trata sobre el marco teórico. En el capítulo 3, se describe los trabajos relacionados acerca del algoritmo seleccionado. En el capítulo 4 trata acerca de la configuración del algoritmo seleccionado y su aplicación. El capítulo 5 presenta resultados detallados y discusión de estudios de caso.

Marco teórico

2.1. Flujo óptimo de potencia

El problema de flujo óptimo de potencia (**OPF** por sus siglas en inglés) se planteó a principios de los años sesenta como una extensión del problema de despacho económico, en su formulación general se trata de un problema de optimización [15], que cuenta con una función objetivo y restricciones no lineales, representando la operación en estado estacionario del sistema [17]. El flujo óptimo de potencia es un problema matemático grande y complejo, que tiene como objetivo principal la minimización de generación de potencia [14]. El flujo óptimo de potencia debe cumplir con dos objetivos principales que son [17]:

- Mantener una operación segura.
- Encontrar un punto de operación económico.

Las principales áreas donde se utiliza **OPF** son: operación y planificación [4, 8], cuya solución determina los valores óptimos de un conjunto de variables tanto de control como de estado, y está sujeta a restricciones que gobiernan los sistemas eléctricos de potencia [9]. A diferencia de los flujos de potencia convencionales el mismo que tiene por objetivo determinar el estado del sistema teniendo como datos de partida la potencia generada y consumidas en todos los nodos, así como el estado de los equipos de control, un **OPF** permite resolver las ecuaciones del sistema eléctrico y obtener el valor de determinadas variables de control que optimizan un objeto concreto, cuantificando de esta forma una función escalar de las variables del problema [10]. Los flujos de potencia **Corriente Alterna (CA)** tienen un grado de precisión en los cálculos muy alta pero la velocidad en la resolución es baja, dentro del análisis del despacho de sistemas eléctricos de potencia no se requiere de una alta precisión en los cálculos, debido a su complejidad y no linealidad, los resultados pueden opacar los parámetros de correlación [7]. La velocidad de resolución es de mayor preocupación dentro de un sistema eléctrico a gran escala [11, 13]. Esto se logra mediante la minimización de una función objetivo y cambiando diferentes sistemas de control, tomando en cuenta las restricciones de igualdad y desigualdad, usadas para modelar las restricciones de balance de potencia activa y varios límites operativos. Por lo tanto, el problema **ACOPF** se formula para minimizar el costo de producción, planificación y minimización de pérdidas, la formulación del problema se detalla en la Ecuación (2.1) [17].

$$\frac{\min}{v, \theta, p, q, b^{cs}} \sum_{g=1}^G C_g \quad (2.1)$$

sujeto a

$$\begin{aligned} \dot{v}_i &\leq v_i \leq \bar{v}_i \forall i \in N \\ \dot{\theta}_i &\leq \theta_i \leq \bar{\theta}_i \forall i \in N \\ \dot{p}_g &\leq p_g \leq \bar{p}_g \forall g \in G \\ \dot{q}_g &\leq q_g \leq \bar{q}_g \forall g \in G \\ \dot{b}_k^{cs} &\leq b_k^{cs} \leq \bar{b}_k^{cs} \forall k \in B \\ p_{inj}^i - p_d^i &= \sum_{(i,j) \in \mathcal{E}} p_{i,j}(v, \theta), \forall i \in N, \forall (i, j) \in \mathcal{E} \\ q_{inj}^i - q_d^i &= \sum_{(i,j) \in \mathcal{E}} q_{i,j}(v, \theta), \forall i \in N, \forall (i, j) \in \mathcal{E} \\ \sqrt{p_{i,j}^2 + q_{i,j}^2} &\leq \bar{s}_{i,j} \forall (i, j) \in \mathcal{E} \end{aligned}$$

Donde N , \mathcal{E} , B y G representan el conjunto de bus, rama (línea y transformador), derivación conmutada y generador. El parámetro C_g representa el costo de generación del generador g , que se calcula a partir de una función de costo lineal individual por partes. Los parámetros v_i y θ_i son la magnitud del voltaje y el ángulo en el bus i , respectivamente. La potencia activa y reactiva del generador se indica mediante p_g y q_g . La susceptancia de derivación se indica mediante b^{cs} y está limitada por sus límites MVAR. Los parámetros p_{inj}^i y q_{inj}^i , representan la inyección de potencia real y reactiva en el bus i , mientras que p_d^i y q_d^i son la demanda de potencia real y reactiva en ese bus, respectivamente. La inyección de potencia neta real y reactiva se representa como $p_{i,j}$, $q_{i,j}$, en una rama se puede representar por la diferencia entre la inyección y la demanda en los buses i y j conectadas. Para una rama que conecta los buses i y j , el flujo de potencia $S_{i,j}$ está limitado por su clasificación $s_{i,j}$. Una solución viable garantiza que no se infrinjan estas limitaciones. Las restricciones derivadas de leyes físicas como la ley de Ohm y la ley de Kirchhoff [4]. Las limitaciones relacionadas con la práctica de la ingeniería podrían relajarse en momentos de contingencia. En este trabajo, con el propósito de generar datos, el límite de flujo de la rama se cambia a un límite blando cuando el caso no se puede resolver tratando el límite de flujo de la rama como un límite estricto. Esta relajación de la restricción permite una mayor variación de fase en los casos necesarios. En una formulación matemática, esta relajación de la restricción se describe en la Ecuación (2.2).

$$-s_{i,j} - \lambda \leq s_{i,j} \leq s_{i,j} + \lambda \quad (2.2)$$

Donde λ es una variable de holgura que permite un margen para que la solución mejore en su espacio de solución factible. Aunque, los flujos de potencia en CA tienen un grado de precisión muy alto en los cálculos, la velocidad en la resolución es baja. Dentro del análisis del despacho de sistemas eléctricos de potencia no se requiere de una alta precisión en los cálculos, debido a su complejidad y no linealidad, los resultados pueden opacar los parámetros de correlación [14]. La velocidad de resolución es de mayor preocupación dentro de un sistema eléctrico a gran escala [11, 13]. Por otro para la resolución de los flujos de potencia en este trabajo se usará aprendizaje automático ML.

2.2. Redes neuronales artificiales

Las redes neuronales artificiales simulan la forma del procesamiento de la información de un sistema nervioso biológico [18]. Cada neurona se representa como una unidad de proceso que forma parte de una entidad mayor, la red neuronal. En este sentido una red neuronal es un procesador de información, de distribución altamente paralela, constituido por muchas unidades sencillas de procesamiento llamadas neuronas las cuales se caracterizan principalmente por [18, 19]:

- Tener una inclinación natural a adquirir conocimiento a través de la experiencia, el cual es almacenado, al igual que en el cerebro humano, en el peso relativo de las conexiones inter-neuronales.
- Tener alta plasticidad y adaptabilidad; son capaces de cambiar dinámicamente junto con el medio.
- Poseer un alto nivel de tolerancia a fallos.
- Tener un comportamiento no lineal puede permitirles procesar información procedente de otros fenómenos no lineales.

En la Figura (2.1), se observa una comparación entre una red neuronal biológica y una red neuronal artificial. Este esquema muestra la similitud entre ellas, en donde se tiene una serie de entradas X_i que equivalen a las dendritas (donde reciben la estimulación las neuronas biológicas de otras neuronas), que ponderadas por un peso W_i (que representan como son evaluados los impulsos entrantes) y combinadas con la función de red o ponderación, se obtendrá el nivel de potencial de la neurona. La salida de la función de red es evaluada con la función de activación que da lugar a la salida de la unidad de proceso. Por tanto, se puede decir que una red neuronal artificial se comporta como una neurona biológica pero de una forma simplificada [16].

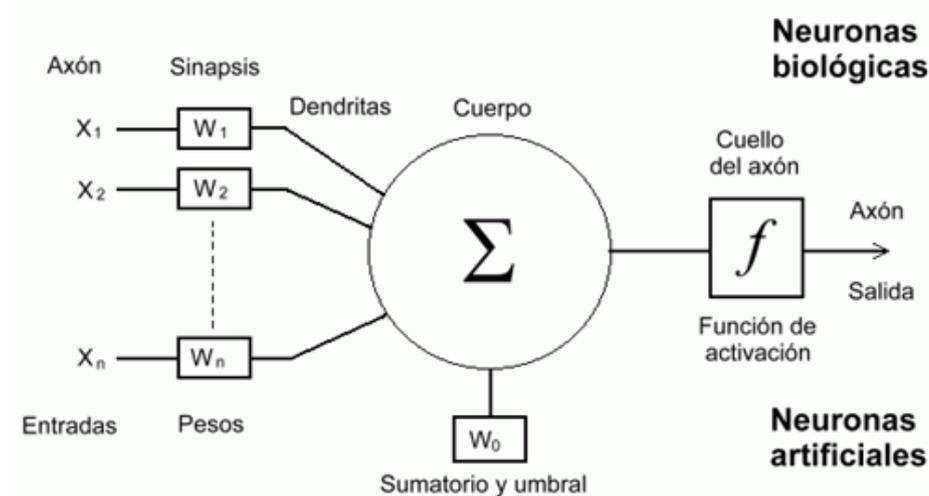


Figura 2.1: Esquema de una red neuronal artificial y una biológica.

Una representación vectorial del funcionamiento básico de una neurona artificial se formula como indica la Ecuación (2.3).

$$O = f(X * W) \quad (2.3)$$

Donde la función f puede ser una función lineal, una función umbral o una función no lineal que simule con mayor exactitud las características de transferencia no lineales de las neuronas biológicas.

2.2.1. Clasificación de redes neuronales artificiales

Las redes de neuronas artificiales se clasifican según su función del patrón de conexiones que presenta. Así se definen dos tipos básicos de redes.

2.2.1.1. Redes de propagación hacia Delante

Las redes neuronales alimentadas hacia delante, generalmente conocidas como redes *feedforward*, son aquellas en las que la información se mueve en un único sentido, desde la entrada hacia la salida. Estas redes están organizadas en “capas”. Cada capa agrupa a un conjunto de neuronas que reciben sinapsis de las neuronas de la capa anterior y emiten salidas hacia las neuronas de la capa siguiente. Entre las neuronas de una misma capa no hay sinapsis [19].

Como se observa en la Figura (2.2), en este tipo de redes existe una capa de entrada, formada por las neuronas que reciben las señales de entrada, y una capa de salida formada por una o más neuronas que emiten la respuesta de la red al exterior. Entre la capa de entrada y las de salida existen una o más capas intermedias, esto se define en dos tipos de redes *feedforward*: monocapa o multicapa.

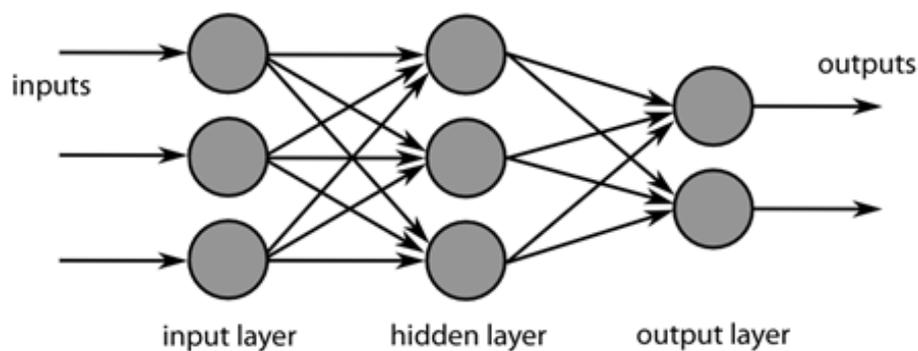


Figura 2.2: Red Neuronal de Tipo feedforward

- **Redes Monocapa:** En este tipo se tienen las redes neuronales: Perceptrón y la red Adaline.
- **Redes Multicapa:** En este tipo se tienen las redes neuronales: Perceptrón Multicapa.

En la Figura (2.3) se ilustra la estructura de cada tipo de red *feedforward* monocapa y multicapa. En redes de este tipo es evidente que la información sólo puede moverse en un sentido: desde la capa de entrada hasta la de salida, atravesando las capas intermedias una sola vez. La no conexión entre las neuronas de una misma capa hace que no existan tiempos de espera; y por lo tanto, que la capa adquiera un estado estable rápidamente. Se trata por tanto de redes rápidas en sus cálculos.

2.2.1.2. Redes de propagación hacia atrás

Las redes neuronales hacia atrás o conocidas como *backpropagation* es un tipo de red de aprendizaje supervisado, que emplea un ciclo de propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas [20]. Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia que todas las neuronas de la capa oculta contribuyen directamente a la salida. Sin embargo, las

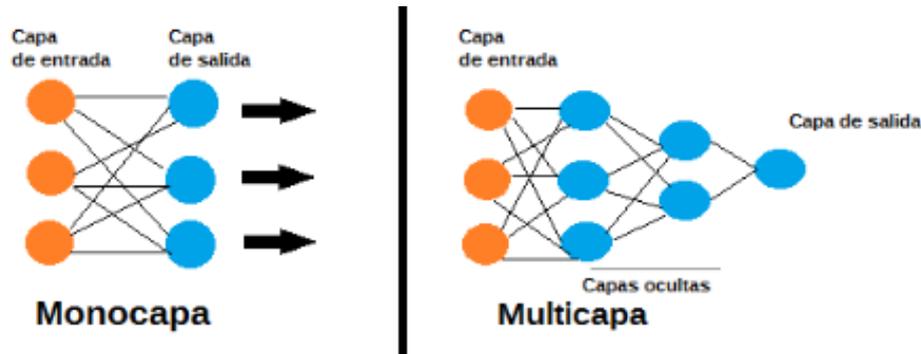


Figura 2.3: Red Monocapa (izquierda), Red multicapa (derecha)

neuronas de la capa oculta solo reciben una fracción de la señal total del error basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total. Basándose en la señal de error percibida, donde se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente los patrones de entrenamiento [19] [20].

2.3. Principales tipos de entrenamiento para redes neuronales

Existen muchos tipos de redes neuronales como se mencionó en la sección anterior. Sin embargo, las redes de tipo *feedforward* son de las más populares. A modo de resumen, consisten en una capa de entrada que recibe el estímulo procedente del ambiente externo, una o más capas ocultas, y una capa de salida que envía la salida al exterior. Para este caso se usan tres acciones principales para el entrenamiento de este tipo de redes [16]:

- **Método basado en descenso por gradiente:** Como por ejemplo *Backpropagation* para redes multicapa hacia adelante. En estos casos se suelen usar nodos ocultos de tipo aditivo, con funciones de activación $g(x) : R \rightarrow R$, tal como se describe en la Ecuación (2.4).

$$g(x) = \frac{1}{1 + \exp(-x)} \quad (2.4)$$

La función de salida del nodo i en la capa oculta l se detalla en la Ecuación (2.5).

$$G(a_i^l, b_i^l, x_i^l) = g(a_i^l \cdot x_i^l + b_i^l), b_i^l \in R \quad (2.5)$$

Donde a_i^l es el vector de pesos que conecta la capa $(l - 1)$ del nodo i con la capa l y b_i^l es el sesgo o *bias* del nodo i en la capa l . Los algoritmos de aprendizaje basados en descenso por gradiente son en general mucho más lentos de lo esperado.

- **Método basado en optimización estándar:** Se utiliza en [Máquinas de vectores de soporte \(SVM\)](#) para un tipo específico de redes monocapa hacia adelante o también llamadas redes de vectores de soporte. Este caso es un mecanismo de aprendizaje donde sólo se ajustaban los pesos de la última capa oculta y la capa de salida. Después de haber fijado todos los pesos los datos de entrada son transformados a un espacio de características Z de la última capa oculta, véase la Figura (2.4). En este espacio de características se construye una función de decisión lineal, Ecuación (2.6).

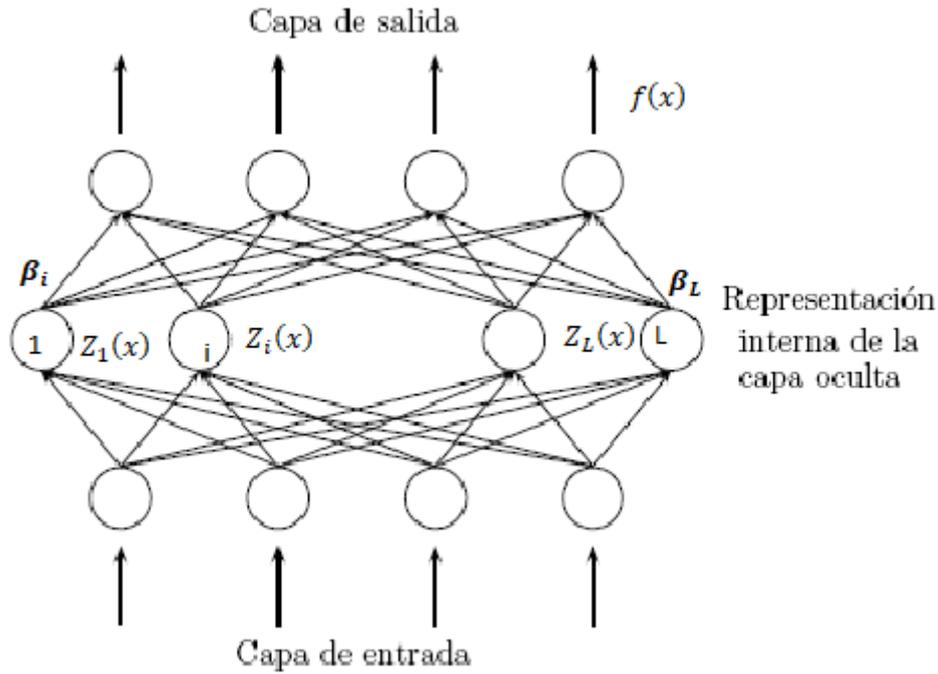


Figura 2.4: Red de tipo *feedforward* multicapa.

$$f(x) = \text{sign}\left(\sum_{i=1}^L \beta_i Z_i(x)\right) \tag{2.6}$$

Donde β_i representa los pesos de la capa de salida entre el nodo de salida y la neurona i en la última capa oculta de un Perceptron, y $Z_i(x)$ es la salida de la neurona i en la última capa oculta del Perceptron.

Para encontrar una solución alternativa de (x) , en [21] propusieron la máquina de soporte de vectores que mapea los datos del espacio de entrada a un espacio de características Z de mayor dimensión, a través de un mapeo no lineal escogido a priori. Los métodos de optimización son usados para encontrar el hiperplano que maximiza los márgenes de separación de las dos clases en el espacio de características.

• **Método basado en mínimos cuadrados:**

Este método es utilizado de la misma forma que las redes **Redes de Funciones de Base Radial (RBF)**, cuyos nodos ocultos tienen una función de activación $g(x) : \mathbb{R} \rightarrow \mathbb{R}$, como la función Gaussiana, véase la Ecuación (2.7).

$$g(x) = \exp(-x^2) \tag{2.7}$$

Donde la función Gaussiana se detalla en la Ecuación (2.8).

$$G(a_i, b_i, x) = g(b_i \|x - a_i\|), b_i \in \mathbb{R}^+ \tag{2.8}$$

Donde a_i y b_i son el centro y factor de impacto del nodo oculto i . \mathbb{R}^+ indica el conjunto de todos los números reales positivos. La red **RBF** es un caso especial de las redes **SLFN** con nodos **RBF** en su capa oculta, véase la Figura (2.5). Cada nodo tiene sus propios parámetros y su salida viene dada por una función simétrica en función de la distancia entre la entrada y el centro.

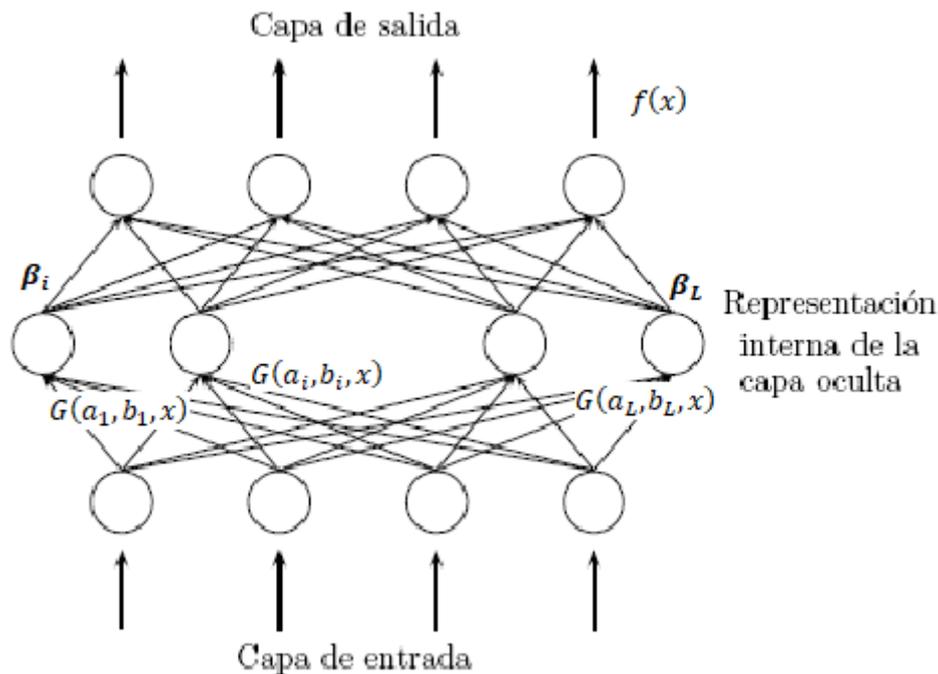


Figura 2.5: Red de tipo feedforward monocapa.

En la implementación de estas redes neuronales, según [22], los centros de los nodos ocultos pueden ser seleccionados aleatoriamente de los datos de entrada o de la región de entrenamiento en lugar de ser entrenados, y todos los factores de impacto b_i se establecen con el mismo valor. Después de fijar estos parámetros, el vector de pesos de salida β_i es el único parámetro desconocido que se puede resolver a través del método de mínimos cuadrados.

ELM fue desarrollada para redes hacia adelante de una sola capa oculta y después se extendieron a redes “generalizadas” hacia adelante monocapa (SLFN). La esencia del ELM es que los pesos de la capa oculta no necesitan ser calculados, es decir, dicha capa no necesita ser entrenada [16]. Una de las implementaciones propias del ELM consiste en usar nodos calculados aleatoriamente en la capa oculta, que puede ser independiente de los datos de entrenamiento. Acorde con la teoría de redes neuronales, en redes con propagación hacia adelante se llega al menor error de entrenamiento cuanto menor sea la norma de los pesos de salida, y la red tiende a tener mayor generalización. Por tanto, los nodos ocultos no necesitan ser entrenados y los parámetros de la capa oculta pueden fijarse para después calcular los pesos de salida mediante el método de mínimos cuadrados.

2.4. Aprendizaje automático

El aprendizaje automático es un método que aprende en base a la información que se le suministra. Estos algoritmos tratan de generalizar y reconocer patrones, a partir de tal información proporcionada. Así como los seres humanos aprenden día a día nuevos conocimientos y adquieren experiencia durante la vida. Los algoritmos de aprendizaje automático aprenden en base a ejemplos que se les muestra, sin ser explícitamente programados para este fin. Para que el algoritmo sea eficiente debe contener una gran cantidad de datos para su entrenamiento y no tener inconvenientes al momento de ponerlo a prueba. El tipo de aprendizaje que se utiliza en los métodos de aprendizaje automático, puede ser clasificado de la siguiente manera [19]:

- **Aprendizaje supervisado:** Los algoritmos que se utilizan en este aprendizaje, generan una función que establece una correspondencia entre la entrada y la salida, donde la base del conocimiento está formada por ejemplos etiquetados.
- **Aprendizaje no supervisado:** Este tipo de aprendizaje se usa cuando la información con la que se entrena el algoritmo, no cuenta con una etiqueta, por lo que el algoritmo busca características similares de los datos de entrada para poder agruparlos de acuerdo a la similitud entre sí.
- **Aprendizaje por refuerzo:** Los algoritmos que usan este tipo de aprendizaje interactúan con su ambiente, realizando un proceso de ensayo y error, reforzando o recompensando aquellas acciones que reciben una respuesta positiva. Este aprendizaje permite determinar automáticamente el comportamiento ideal en un contexto específico, con el fin de minimizar su desempeño.

Estas son las categorías principales en las que se clasifican un tipo de aprendizaje automático. Aunque, el aprendizaje automático mediante el uso de **Redes Neuronales Artificiales (RNA)** ha sido empleado con éxito en múltiples aplicaciones durante las últimas décadas [4]; surgen diversos inconvenientes durante el proceso de optimización de los hiper-parámetros tales como: pesos y número de neuronas ya que requiere de un tiempo elevado de cálculo para obtener su convergencia y determinar los mínimos locales. Esto sucede debido a la no-unicidad de la solución para un determinado problema al usar métodos de entrenamiento basados en gradiente [1]. Varios trabajos han sido desarrollados para obtener algoritmos de entrenamiento supervisado rápidos y precisos para esquemas del tipo *feed-forward*, siendo eficientes al momento de determinar una solución.

Las **RNA** del tipo *feed-forward*, o perceptrones multicapa **Multi-Layer Perceptron (MLP)**, constan de múltiples capas de neuronas interconectadas entre sí; cualquier función continua se puede aproximar seleccionando los hiper-parámetros adecuadamente [1]. Un **MLP** estándar está compuesto de una capa oculta de H neuronas. Los pesos de la capa de entrada conectan las n variables de entrada con las H neuronas, mientras que una segunda capa de pesos conecta las salidas de las H neuronas con las m unidades de salida del **MLP**. Al considerar un vector de entrada $x = [x_1, x_2, x_3, \dots, x_n]$ con *target* $t = [t_1, t_2, t_3, \dots, t_m]$ la salida del **MLP** viene detallada en la Ecuación (2.9) [18].

$$y = \sum_{j=1}^H \beta_j f(w_j * x + b_j) \quad (2.9)$$

Donde, $y = \{y_k\}_{k=1}^m$, $f(\cdot)$, son las funciones de activación de las neuronas ocultas, $\beta_j = [\beta_{j1}, \beta_{j2} \dots \beta_{jm}]^T$ son los pesos de la salida asociada a la neurona oculta, mientras que w y b_j representan el sesgo de entrada asociado a dicha neurona. El uso de la función de activación varía de la solución para la de la función objetivo, las funciones de activación se describen en la Sección 2.7.

2.4.1. Modelos de aprendizaje automático

Existen muchos modelos de aprendizaje automático dentro de **ML**; a continuación se enumeran los que están más relacionados con este trabajo y su aplicación con respecto a **ACOPF**.

2.4.1.1. Modelo de bosque aleatorio **RF**

El modelo de bosque aleatorio o **RF**, es un conjunto de árboles de decisión que combina las predicciones de distintos árboles y puede utilizarse como método de clasificación [22]. El bosque aleatorio utiliza una variación del “*boosting*”, llamada “*boosting aggregation*” [23]. Este método primero elabora un conjunto de árboles como base, para luego construir con las muestras un conjunto *bootstrap* aleatorio y se escoge la moda de las clases

predichas como pronóstico. Los árboles resultantes tienden a ser profundos, i.e., de múltiples niveles, y tienden a sobre-ajustar los datos, es decir, a ser poco generalizables, pero el procedimiento de hallar la moda de los pronósticos de cada uno de los árboles de decisión generados mejora el rendimiento de la predicción [23]. De hecho, el error de generalización de un bosque converge al aumentar el número de árboles y depende, a su vez, de cada árbol de decisión y de la correlación existente entre estos [24].

En principio, los árboles que resultan del procedimiento anterior suelen ser correlacionados, lo que empeora el pronóstico general, pero una técnica denominada “*dropout*” permite descorrelacionar los árboles al considerar solo un subconjunto de los predictores que han sido establecidos aleatoriamente al momento de dividir una rama potencial del árbol [23]. Esta técnica permite la reducción de la correlación promedio entre las predicciones y mejora el comportamiento de la predicción global en comparación con el “*bagging*” estándar. Para modelar un bosque aleatorio, se suelen optimizar parámetros de ajuste como la profundidad de los árboles y el número de muestras “*bootstrap*” mediante la validación cruzada; esto es, usando una muestra de validación no vista previamente para garantizar la generalización del método [23, 24].

2.4.1.2. Árbol de decisiones de objetivos múltiple DT

Los árboles de decisión o DT pueden ser utilizados tanto para problemas de clasificación de variable de respuesta categórica y para problemas de regresión de variable de respuesta continua [22, 25]. Este modelo consiste en segmentar el espacio de las variables independientes predictoras en un número de regiones simples. Para esto, se utiliza una serie de reglas binarias que permiten generar la partición del espacio y las cuales pueden ser representadas en forma de árbol. Este tipo de modelos se basan en la estructura de árboles y tienen la ventaja de que son simples e interpretables, lo cual puede llegar a ser muy útil dependiendo de los objetivos que se quieran cumplir con el modelo. Por otra parte, tienen la desventaja de que por lo general no tienen tan buen desempeño al momento de compararlos con otros modelos de ML [20, 25].

El método comienza con todas las observaciones contenidas dentro de un primer rectángulo. Este espacio se divide en dos partes procurando obtener la mejor partición posible independientemente de las posibles particiones futuras. En caso de ser necesario, las nuevas regiones conformadas se vuelven a dividir de forma binaria y se continúa así, sucesivamente, hasta obtener un resultado satisfactorio. Para realizar la partición de cada región, se selecciona un predictor X_j y un punto de corte s tal que el espacio de predicciones sea dividido en regiones de forma que se minimice el **Suma residual de cuadrados (RSS)**; es decir, se consideran todas las posibles predicciones y todos los posibles puntos de corte s para cada predicción y se escoge la región cuyo árbol resultante tenga el mejor RSS. Entonces, el proceso se repite buscando una nueva región de mejor predicción y un nuevo punto de corte, que dividan una de las dos regiones creadas anteriormente. Este proceso se repite hasta que se alcance algún criterio de tolerancia [25, 26].

2.4.1.3. Máquina de aprendizaje extrema ELM

El algoritmo de máquina de aprendizaje extrema ELM se basa en la estructura de un MLP compuesto por H neuronas, cuyos pesos de entrada se inicializan aleatoriamente. Estos pueden “aprender” N distintos casos de entrenamiento produciendo un error cero, siendo $N \geq H$, y aproximar cualquier tipo de función continua [27]. ELM inicializa de manera aleatoria los pesos de entrada; los pesos de salida pueden obtenerse de manera analítica mediante un cálculo simple de la matriz pseudo-inversa de las salidas de las H neuronas ocultas para un determinado conjunto de entrenamiento [10, 11, 27]. Así, dado un conjunto de N vectores de entrada, un MLP con H neuronas ocultas puede aproximar estos N casos a un error nulo. Esto se puede expresar como $H\beta = T$ donde H es la matriz de salidas de la capa oculta de neuronas del MLP, β es la matriz de pesos de salida y T es

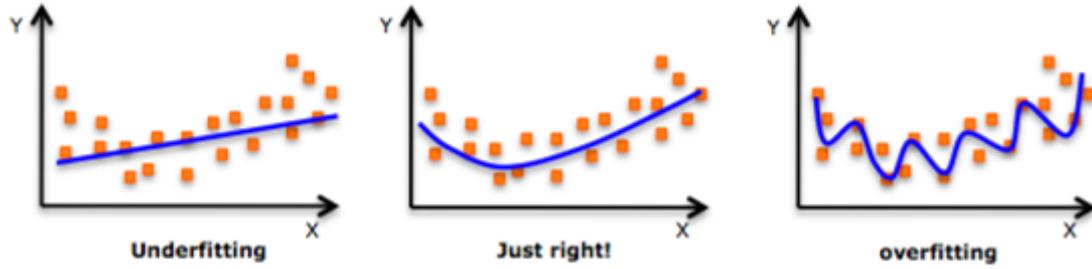


Figura 2.6: Sistema poco entrenado, correctamente entrenado y sobreentrenado [28].

la matriz de “targets” de los N casos de entrenamiento [10, 11]. Por otra parte, a pesar de su rapidez, al iniciar los pesos de la capa de entrada aleatoriamente, se necesita muchas neuronas en la capa oculta para entrenar bien el sistema. Por tanto, se debe tener muchos parámetros (pesos) a optimizar y la red sobre-entrenará produciendo un gran error de generalización. En la Figura (2.6) se detalla los tres casos de respuesta que se puede obtener.

2.5. Teoría de aprendizaje de ELM

Las capacidades de interpolación y de aproximación universal de ELM han sido principalmente investigadas por Huang en [16, 19, 21]. La función de salida de una red SLFN con L nodos ocultos se presenta en la Ecuación (2.10).

$$f_L(x) = \sum_{i=1}^L \beta_i g_i(x) = \sum_{i=1}^L \beta_i G(a_i, b_i, x), x \in \mathbb{R}^d, \beta_i \in \mathbb{R}^m \quad (2.10)$$

Donde g_i denota la función de salida del nodo oculto $G(a_i, b_i, x)$ Para nodos aditivos con función de activación g , g_i y esto se define en la Ecuación (2.11):

$$g_i = G(a_i, b_i, x) = g(a_i \cdot x + b_i), a_i \in \mathbb{R}^d, b_i \in \mathbb{R} \quad (2.11)$$

Y para nodos RBF con función de activación g , g_i se define en la Ecuación (2.12):

$$g_i = G(a_i, b_i, x) = g(b_i \|x - a_i\|), a_i \in \mathbb{R}^d, b_i \in \mathbb{R}^+ \quad (2.12)$$

La capacidad de interpolación y aproximación universal de las redes SLFN han sido investigadas profundamente en [11]. Se probó que N muestras distintas pueden ser aprendidas por este tipo de redes con exactamente N nodos ocultos. Además, este estudio dio una respuesta más completa sobre la capacidad de interpolación de las SLFN y probó que para una red con muchos nodos ocultos (N) y con una función de activación arbitraria no lineal y acotada, puede aprender cualquier conjunto de N muestras distintas sin error. Tales funciones de activación incluyen la escalón, rampa y la sigmoideal así como las de base radial, funciones coseno cuadrado y muchas no regulares [10, 11].

En [16, 19, 21] se ha demostrado rigurosamente que dada una función de activación $g(x)$ que satisface ciertas condiciones, existe una secuencia de funciones de red $\{f_L\}$ que aproximan cualquier función continua f dada como función objetivo con un error esperado $\epsilon > 0$. En todas estas teorías convencionales, todos los parámetros de cualquier f_L de la secuencia de red; por ejemplo, los parámetros de la capa oculta (a_i, b_i) y los pesos de salida β_i necesitan ser ajustados libremente. De acuerdo con estas teorías convencionales, los parámetros (a_i, b_i)

necesitan ser entrenados para encontrar los valores apropiados para la función objetivo f [11].

Los parámetros de la capa oculta en aquellos modelos convencionales de ML necesitan ser ajustados al menos una vez basándose en las muestras de entrenamiento. En contraste, todos los parámetros de ELM no necesitan ser entrenados y pueden ser independientes de las muestras de entrenamiento.

2.5.1. Máquina de aprendizaje extremo ELM

La esencia y las ventajas de implementar ELM reside en:

- La capa oculta de la red no necesita ser calculada iterativamente.
- Acorde con la teoría sobre redes hacia delante, el error de entrenamiento $\|H\beta - T\|$ y la norma de los pesos $\|\beta\|$ necesitan ser minimizados.
- Corresponde a una red neuronal pequeña

Acorde con lo que se describe en [11] los pesos de los nodos ocultos pueden ser generados aleatoriamente, de tal forma que los únicos parámetros desconocidos serán los vectores de pesos de salida β_i entre la capa oculta y la de salida, que pueden ser calculados directamente mediante mínimos cuadrados. Los parámetros de los nodos (a_i, b_i) permanecen fijos una vez que se han establecido aleatoriamente. Entrenar la red SLFN simplemente es equivalente a encontrar la solución por mínimos cuadrados del sistema lineal $H\beta = T$, lo cual se detalla en la Ecuación (2.13):

$$\|H\hat{\beta} - T\| = \min_{\beta} \|H\beta - T\| \quad (2.13)$$

Si el número L de nodos ocultos es igual al número de muestras de entrenamiento N , $L = N$, la matriz H es cuadrada e invertible cuando los parámetros (a_i, b_i) son elegidos aleatoriamente, y así las redes SLFN pueden aproximar dichas muestras de entrenamiento con un error nulo. Sin embargo, en la mayoría de los casos el número de nodos es mucho menor que el número de muestras de entrenamiento, $L \ll N$ por lo que H no es cuadrada y puede no existir los parámetros $a_i, b_i, \beta_i (i = 1, \dots, L)$ tal que $H\beta = T$. La solución que minimiza por mínimos cuadrados el sistema de ecuaciones se detalla en la Ecuación (2.14).

$$\hat{\beta} = H^+ T \quad (2.14)$$

Donde H^+ es la inversa generalizada Moore-Penrose de la matriz H . Así, el algoritmo ELM se puede resumir como: dado un conjunto de entrenamiento $N = \{(x_i, t_i) \mid x_i \in \mathbb{R}^d, t_i \in \mathbb{R}^m, i = 1, \dots, N\}$, una función de salida de los nodos ocultos $G(a_i, b_i, x)$ y un número de nodos ocultos L , entonces:

- Generar de forma aleatoria los parámetros de los nodos ocultos (a_i, b_i) .
- Calcular la matriz de salida de la capa oculta H .
- Calcular los pesos de la capa de salida como se detalla en la Ecuación (2.14).

La ventaja de utilizar el algoritmo ELM es que se puede trabajar con muchos tipos de funciones de activación. Muchos algoritmos de ML no están preparados para trabajar con redes umbral directamente y en su lugar usan redes analógicas para aproximar las redes umbrales, tales como el método de descenso por gradiente. Sin embargo, ELM puede ser usado para entrenar este tipo de redes directamente sin necesidad de recurrir a otros algoritmos. Según estudios recientes expuestos en [29] hablan acerca de la implementación de Redes Neuronales Profundas (Deep Neural Networks (DNN)) y Máquinas de Aprendizaje Extremo (ELM) siendo métodos prometedores para encontrar soluciones y predicciones precisas. En [29] se describe que la arquitectura

de [DNN](#) presenta un buen rendimiento de predicción y esta arquitectura supera a [ELM](#). Aunque estos dos modelos [DNN](#) y [ELM](#) cada uno con su arquitectura y funcionalidad son adecuados para dar solución al problema presentado, pero la principal contribución de este trabajo se enfoca en encontrar una solución implementando el método [ELM](#).

2.6. Arquitectura y funcionamiento de [ELM](#)

El algoritmo [ELM](#), inicia aleatoriamente los pesos que unen la capa de entrada y la capa oculta. De esta manera, sólo será necesario optimizar los pesos que están entre la capa oculta y la capa de salida. La arquitectura del algoritmo [ELM](#) se muestra en la Figura (2.7) en donde se observa que la red puede tener N patrones de entradas y la salida deseada. Esta red corresponde a una red neuronal pequeña pero cumple con el objetivo planteado en este trabajo. En la capa oculta y la salida para optimizar estos pesos se utiliza la matriz pseudoinversa de Moore-Penrose [28]. [ELM](#) disminuye notablemente el tiempo computacional empleado para ajustar los pesos debido a que no utiliza ningún método de búsqueda para los coeficientes ocultos. En cambio, como se detallará a continuación, el método de la matriz pseudoinversa de Moore-Penrose es un simple y rápido cálculo algebraico.

2.6.1. Matriz pseudoinversa de Moore-Penrose

Dado un conjunto de N patrones, $D = (x_i, o_i)$, $i = 1 \cdots N$, donde las entradas son $x_i \in R^{d_1}$ y la salida deseada $o_i \in R^{d_2}$. El objetivo es encontrar la relación entre x_i y o_i . Siendo M el número de neuronas en la capa oculta y y_i la salida j -ésima del [SLFN](#), lo cual se formula en la Ecuación (2.15).

$$y_i = \sum_{k=1}^M h_k \cdot f(w_k, x_j) \quad (2.15)$$

Donde $1 \leq j \leq N$, w_k representa los parámetros del k -ésimo elemento de la capa de entrada, h_k es el k -ésimo peso que conecta la capa de entrada con la capa de salida y f es una función de activación aplicada al producto escalar del vector de entrada y los pesos de la capa de salida. La Ecuación (2.15) también puede ser expresada como $y = G \cdot h$ donde h es el vector de pesos de la capa de salida, y es el vector de salida y G se calcula como se muestra en la Ecuación (2.16).

$$G = \begin{pmatrix} f(w_1, x_1) & \cdots & f(w_M, x_1) \\ \vdots & \ddots & \vdots \\ f(w_1, x_N) & \cdots & f(w_M, x_N) \end{pmatrix} \quad (2.16)$$

$$h = \begin{pmatrix} h_1 \\ \vdots \\ h_M \end{pmatrix} \quad (2.17)$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \quad (2.18)$$

Como se comentó anteriormente, se inicializan aleatoriamente los pesos de la capa de entrada. Los pesos de la capa de salida se obtienen mediante la matriz pseudoinversa de Moore-Penrose G^+ , la cual se detalla la formulación en la Ecuación (2.19).

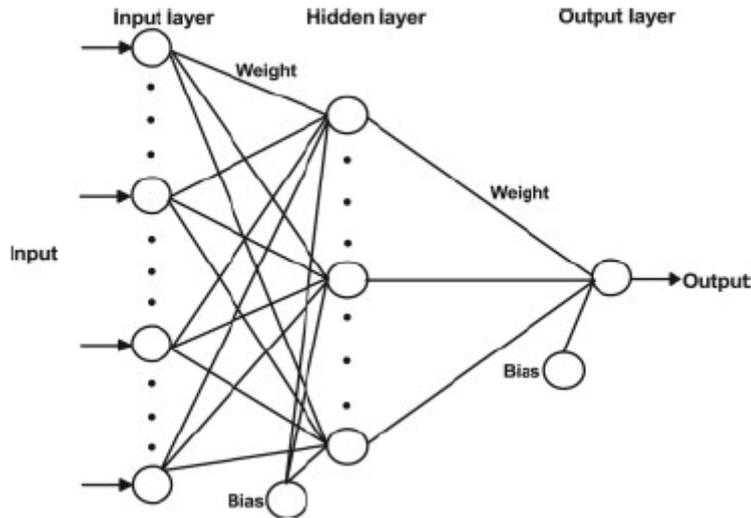


Figura 2.7: Arquitectura utilizada de ELM [28]

$$h = G^+ \cdot o \tag{2.19}$$

Donde la matriz pseudoinversa G^+ , se calcula mediante la Ecuación (2.20).

$$G^+ = (G^T \cdot G)^{-1} \cdot G^T \tag{2.20}$$

Por lo tanto, o es el vector de salida deseado definido como se muestra en la Ecuación (2.21).

$$o = \begin{pmatrix} o_1 \\ \vdots \\ o_N \end{pmatrix} \tag{2.21}$$

Como puede observar en la Ecuación (2.19), una vez calculada G^+ , basta con multiplicarla con la salida deseada (o) para obtener, automáticamente, el vector de pesos de la capa de salida (h).

2.7. Función de activación

La función de activación se encarga de devolver una salida a partir de un valor de entrada; normalmente el conjunto de valores de salida en un rango determinado como $(0, 1)$ o $(-1, 1)$. La función de activación determina el estado de activación actual de la neurona en base al potencial resultante net_i y al estado de activación anterior de la neurona $ai(t - 1)$ para un determinado instante de tiempo t , véase la Ecuación (2.22) [16].

$$a_i(t) = f(a_i(t - 1), net_i(t)) \tag{2.22}$$

Sin embargo, en la mayoría de los modelos se suele ignorar el estado anterior de la neurona, definiéndose el estado de activación en función del potencial resultante a_i Ecuación (2.23) [16].

$$a_i(t) = f(net_i(t)) \tag{2.23}$$

Se busca que las funciones de activación sean derivadas simples, para minimizar con ello el coste computacional. Algunas de las funciones de activación más utilizadas en los distintos modelos de redes neuronales se detallan en la Figura (2.8) [16]. En el caso de estudio a aplicar, uno de los parámetros de entrada del sistema eléctrico es la carga reactiva q_d y a menudo toma un valor negativo, por lo cual se selecciona una función de activación “sigmoidea”. El beneficio de utilizar este tipo de función de activación es que actúa como una especie de función “aplastadora”, comprimiendo nuestra salida a un rango de 0 a 1, un valor negativo lo aproxima a 0 y un valor positivo lo aproxima a 1.

La función de salida convierte el estado de la neurona en la salida hacia la siguiente neurona que se transmite por la sinapsis. Normalmente no se usa y se toma la identidad de manera que la salida es el propio estado de activación de la neurona en base al estado de activación de la neurona, lo cual se detalla en la Ecuación (2.24).

$$y_i(t) = f(\text{net}_i(t)) \quad (2.24)$$

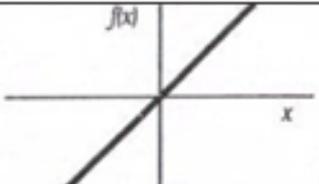
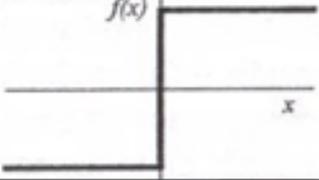
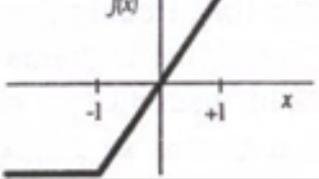
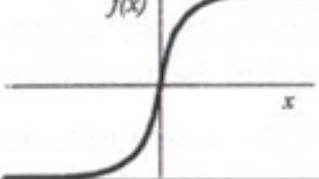
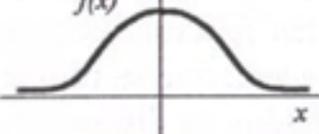
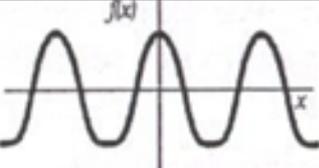
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, \infty]$	
Escalón	$y = \begin{cases} 1, & \text{si } x \geq 0 \\ 0, & \text{si } x < 0 \end{cases}$ $y = \begin{cases} 1, & \text{si } x \geq 0 \\ -1, & \text{si } x < 0 \end{cases}$	$[0, 1]$ $[-1, 1]$	
Lineal a tramos	$y = \begin{cases} 1, & \text{si } x > 1 \\ x, & \text{si } -1 \leq x \leq 1 \\ -1, & \text{si } x < -1 \end{cases}$	$[-1, 1]$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = \tanh(x)$	$[0, 1]$ $[-1, 1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, 1]$	
Sinusoidal	$y = A \sin(wx + \varphi)$	$[-1, 1]$	

Figura 2.8: Funciones de activación [16]

Trabajos relacionados

Las técnicas de aprendizaje automático han sido usadas en una gran variedad de aplicaciones. Entre las numerosas técnicas, las redes neuronales y las máquinas de vectores de soporte (**SVM** por sus siglas en inglés) han jugado un papel dominante. Sin embargo, ambos se enfrentan a algunas cuestiones difíciles de resolver tales como su velocidad de aprendizaje lenta[27].

Recientemente, **ELM** ha atraído la atención de los investigadores como una técnica emergente que resuelve algunos problemas a los que se enfrentan otras técnicas. **ELM** fue propuesto por Huang [30]. Este algoritmo se usa en una estructura multicapa con una sola capa de neuronas oculta (**SLFN**, *Single Layer Feedforward Network*) [12]. La esencia del **ELM** es que la capa oculta de la red no necesita ser entrenada. **ELM** mantiene la capacidad de generalización y un entrenamiento mucho más rápido con menor intervención humana comparado con las técnicas de inteligencia computacional tradicionales [4, 16, 31].

Los modelos de aprendizaje automático desarrollados para resolver **ACOPF** deben tener una capacidad de mapeo de múltiples entradas y múltiples salidas *Multiple input multiple output (MIMO)* [4], con una alta velocidad de cálculo y precisión. La mayoría de los algoritmos de aprendizaje automático existentes no tienen la capacidad de manejo de **MIMO** y tienen que depender de técnicas de conjunto para realizar tareas **MIMO**, especialmente para problemas de regresión. Sin embargo, la eficiencia computacional puede verse afectada significativamente por los métodos de conjunto, y el rendimiento puede deteriorarse a través del proceso de ajuste de la red neuronal [32, 33].

Se ha demostrado que los algoritmos de vecinos más cercanos y basados en árboles funcionan bien en el mapeo de múltiples objetivos; sin embargo, los métodos del vecino más cercano son notoriamente lentos e inadecuados[15]. Entonces se adopta y compara dos algoritmos de aprendizaje automático basados en árboles para realizar la tarea de aprendizaje, que son **RF** y **DT**. También se adopta una red neuronal de retroalimentación **SLFN** de una sola capa extremadamente rápida, llamada **ELM**, para fines de comparación, debido a su capacidad de selección automática de características y entrenamiento rápido. En [14, 31] se probaron varios algoritmos de aprendizaje profundo, como la red neuronal de alimentación directa profunda y la red neuronal convolucional profunda unidimensional, debido a su capacidad de selección automática de características. Sin embargo, las precisiones de los algoritmos de aprendizaje profundo probados no son satisfactorias y sus resultados no se detallan a profundidad en este trabajo.

3.1. Modelo de bosque aleatorio RF

El modelo de bosque aleatorio (RF) es una combinación de árboles predictivos (clasificadores débiles); este trabaja con una colección de árboles incorrelacionados y los promedia [34]; cada árbol depende de los valores de un vector aleatorio de la muestra independiente y con la misma distribución de todos los árboles en el bosque. Aunque este modelo tiene una capacidad superior de mapeo de múltiples entradas y múltiples salidas; se adapta perfectamente al propósito de estimar múltiples variables en ACOPF.

Existe una colección de árboles de decisión en un modelo de RF para hacer sus predicciones por separado. Luego, estas predicciones se promedian para minimizar el error. Los árboles son aprendices básicos y casi independientes, lo que reduce el riesgo de decisiones sesgadas o sobreajuste. Durante el entrenamiento del modelo de RF, las muestras de arranque se extraen del conjunto de datos de entrada (es decir, la carga del sistema P_d y Q_d), y para cada muestra, se hace crecer un árbol de regresión sin podar. En cada nodo, las variables de entrada se muestrean aleatoriamente y la división se realiza en el espacio de variables. La mejor división se asegura minimizando la RSS, dada por la Ecuación (3.1) [34].

$$RSS = \sum_{l=1}^L \sum_{m \in R_l} (y_m - \check{y}_{R_l})^2 \quad (3.1)$$

Donde, \check{y}_{R_l} es el promedio de las observaciones en la l -ésima región no superpuesta, L es el número total de regiones del espacio predictor y y_m es el valor predicho. El proceso de entrenamiento continúa hasta que el número de árboles crecidos es igual a un número predefinido. Los resultados del modelo, es decir, v y θ se generan agregando la predicción de todos los árboles cultivados. La solución promedio de todos los árboles se considera como la solución final, lo que asegura cero errores de predicción aleatorios de los árboles y preserva la verdadera relación entre predictores y respuestas. El pseudocódigo para la OPF basado en RF se detalla en algoritmo 1 [4].

Algorithm 1 Algoritmo RF

Requiere: Conjunto de datos de prueba y entrenamiento de RF: $((X_{tr}, Y_{tr}), (X_{test}, Y_{test}))$ Asegúrese de que las variables OPF estén determinadas: 1. Bosque aleatorio $((X_{tr}, Y_{tr}), B)$

2. Inicializar $Q = \varphi$

3. for $e = 1$ to B do

4. $I_e \leftarrow$ una muestra del conjunto de entrenamiento

5. $q_e \leftarrow$ obtenga el árbol aprendido de la muestra con un punto de corte para minimizar la Ecuación (3.1)

6. $Q \leftarrow Q \cup q_e$

7. end for

8. return Q

9. Predecir con Q en X_{test} y v y θ

3.2. Árbol de decisiones de objetivos múltiples DT

El método de decisiones de objetivos múltiples DT es capaz de realizar una regresión multiobjetivo a la vez, mediante un solo árbol en lugar de construir un árbol para cada variable objetivo. El DT multiobjetivo puede aprovechar los beneficios de la correlación entre las variables de entrada. En comparación con un árbol de regresión de un solo objetivo, el DT de varios objetivos es significativamente más pequeño y requiere menos tiempo para entrenar. A diferencia de almacenar un único valor numérico en cada nodo hoja, cada hoja DT de

objetivos múltiples almacena un vector. El aprendizaje comienza en el nodo raíz de un único DT con un conjunto de datos de entrenamiento, y el conjunto se divide de forma recursiva en subconjuntos más pequeños mediante una función heurística [35]. En comparación al método de árbol de un solo objetivo, una función heurística selecciona la variable de entrada establecida en cada nodo interno de un DT de varios objetivos sobre la base de la variación intra-grupo, véase la Ecuación (3.2) [4, 35].

(3.2)

$$N \sum_{t=1}^T var[y_t] \quad (3.2)$$

Donde T es el número de variables objetivo, N es el número de muestras y $var[y_t]$ es la varianza de la variable objetivo y_t , en el grupo (*cluster*) [4, 35]. Un valor más pequeño de la varianza intra-subconjunto representa una mayor precisión en la predicción. El proceso de partición se detiene cuando se cumple un criterio de parada predefinido, es decir, la profundidad máxima del árbol [35]. El pseudocódigo para la OPF basado en DT se detalla en 2 [4]:

Algorithm 2 Algoritmo DT

Requiere: Conjunto de datos de prueba y entrenamiento de DT: $((X_{tr}, Y_{tr}), (X_{test}, Y_{test}))$

Asegúrese de que las variables OPF estén determinadas:

1. DT $((X_{tr}, Y_{tr}), B)$
 2. Inicializar árbol ($tree$) = φ
 3. Inicializar k conjunto de clústeres $C_k = (X_{trk}, Y_{trk})$
 4. $C^* \leftarrow$ Obtenga el mejor conjunto de datos minimizando la varianza dentro del conjunto
 5. $y^* \leftarrow$ obtener el valor promedio de los objetivos en el nodo hoja
 6. if $y^* = \varphi$ then
 7. for each $C_k \in C^*$ do
 8. $tree \leftarrow DT((X_{trk}, Y_{trk}))$
 9. end for
 10. return $node(y^*, tree)$
 11. else
 12. return $leaf(y^*)$
 13. end if
 14. return tree
 15. Predecir con árbol en X_{test} para obtener v y θ
-

3.3. Máquina de aprendizaje extrema ELM

La máquina de aprendizaje extremo o ELM es un modelo de red neuronal SLFN en el que los nodos de la capa oculta son seleccionados aleatoriamente y los nodos de la capa de salida se determinan analíticamente. ELM ha mostrado ser capaz de reducir significativamente el tiempo de entrenamiento y predicción, en comparación con los algoritmos clásicos de ML. La baja eficiencia computacional de los algoritmos clásicos de ML se atribuye principalmente a: i) algoritmos de aprendizaje basados en gradientes lentos, y ii) el ajuste iterativo agitado de los parámetros del modelo para optimizar su rendimiento [16]. ELM aborda los desafíos antes mencionados eligiendo aleatoriamente los pesos de entrada y determinando analíticamente los pesos de salida. Los pesos de entrada y los sesgos de capa oculta se asignan al inicio del proceso de aprendizaje. La matriz de salida de la capa oculta (H) permanece inalterada si el número de nodos ocultos es igual al número de muestras de entrada distintas. Si ocurre una excepción, las ponderaciones de salida se calculan usando la inversa generalizada de

Moore Penrose de la matriz H en lugar de usar H directamente [27, 31]. Para una sola variable objetivo, un ELM con N neuronas ocultas se describe en la Ecuación (3.3).

$$f_{\tilde{N}}(x) = \sum_{i=1}^{\tilde{N}} \beta_i h_i(x) = h(x)\beta \quad (3.3)$$

Donde x es la entrada, β es el vector de ponderación de salida y $h(x)$ es el vector de salida de la capa oculta. Para un problema de regresión de objetivos múltiples, $f_{\tilde{N}}$ es una matriz $M \times \tilde{N}$ donde M es el número de variables objetivo. La función de costo E se minimiza con N muestras la cual se describe en la Ecuación (3.4).

$$E = \sum_{j=1}^N \left(\sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) - y_j \right)^2 = \|H\beta - y\|^2 \quad (3.4)$$

Donde $g(\cdot)$ es una función de activación que se supone infinitamente diferenciable. El parámetro w_i es un vector de peso que conecta el nodo de entrada y el i -ésimo nodo oculto. El parámetro b_i es un umbral del i -ésimo nodo oculto. $w_i \cdot x_j$ denota el producto interno entre w_i y x_j , y_j representa un vector objetivo y y representa una matriz objetivo. El pseudocódigo para OPF basado en ELM se ilustra en la tabla 3 [4]:

Algorithm 3 Algoritmo ELM

Requiere: Conjunto de datos de prueba y entrenamiento de ELM: $((X_{tr}, Y_{tr}), (X_{test}, Y_{test}))$

(No de neuronas ocultas \tilde{N} , función de activación $g(x)$ Asegúrese de que las variables OPF estén determinadas:

1. ELM $((X_{tr}, Y_{tr}), \tilde{N})$
 2. Inicializar árbol $\beta = \varphi$ Y $H = \varphi$
 3. for $j = 1$ to N do
 4. for $i = 1$ to \tilde{N} do
 5. Inicializar w_i and b_i
 6. $h_i(x) \leftarrow$ calcular $h_i(x)$ evaluando cada $g(x)$
 7. $\beta_i \leftarrow$ calcular β_i
 8. end for
 9. return $h(x_j), \beta$
 10. $H \leftarrow H \cup h(x_j)$
 11. end for
 12. Actualizar $\beta \leftarrow \beta$ con la Ecuación (3.4)
 13. return H, β
 14. return tree
 15. Predecir con árbol en X_{test} para obtener v y θ
-

Diseño e implementación

4.1. Metodología OPF con aprendizaje automático

En la Figura (4.1), se observa el marco general del enfoque de aprendizaje automático para resolver ACOPT [4, 11]. Con condiciones de carga dadas (es decir, carga de potencia real y reactiva), las magnitudes y ángulos de voltaje correspondientes se resuelven utilizando un solucionador OPF convencional, es decir, el solucionador de puntos interiores MATPOWER (MIPS), que genera un gran conjunto de datos para el modelo ML. El uso de métodos de aprendizaje automático podría usarse en una serie de escenarios cuando se necesiten soluciones OPF en tiempo real, como:

- Decisión de despacho en tiempo real ante contingencias, desastres o ciberataques.
- Potenciales redes eléctricas futuras con altas penetraciones de generación renovable incierta.

Además, en condiciones normales, la solución de ML también podría usarse como un inicio en caliente para la solución consecutiva, ya que los estados del sistema varían poco en un período corto de tiempo sin interrupciones importantes [33].

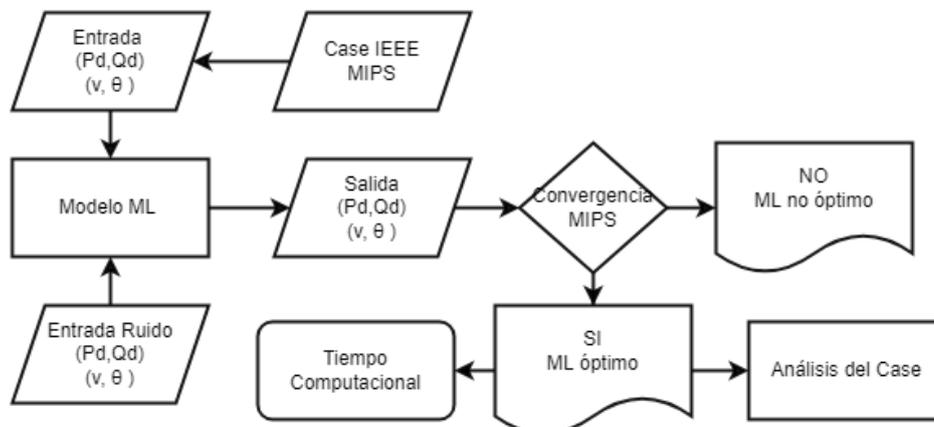


Figura 4.1: Esquema de Metodología OPF con aprendizaje automático ML

Por lo tanto, en el proceso de entrenamiento del algoritmo **ELM** se ingresa el conjunto de datos tomados de la salida del solucionador **MIPS** convencional. El conjunto de datos generado se divide en datos de entrenamiento (80 %) y de validación (20 %). Una vez entrenado el algoritmo se utiliza los pesos calculados para realizar predicciones posteriormente. Después de resolver los parámetros de salida, se toma el valor del voltaje, y se aplican ecuaciones de red para determinar la inyección de corriente en diferentes buses, con base en una matriz de admitancia de bus Y que generalmente se calcula a partir de parámetros de red dados, Ecuación (4.1).

$$I^{inj} = Y \times v \quad (4.1)$$

Una vez que se conocen las inyecciones de corriente, la potencia aparente S se calcula a partir del producto de la tensión del bus y la corriente conjugada, Ecuación (4.2).

$$S = v \times I^{inj*} \quad (4.2)$$

Finalmente, la inyección de potencia de bus compleja S_{inj} se calcula restando la demanda de potencia de bus S_d de la potencia aparente, Ecuación (4.2). Esto se inspira en el ajuste repetitivo de los parámetros de voltaje del sistema para producir un desajuste cero en la inyección de energía de bus compleja, que generalmente se usa en algoritmos de flujo de energía basados en la física.

$$S_{inj} = S - S_d \quad (4.3)$$

La parte real de la compleja inyección de energía del bus en el bus del generador alimenta luego a las funciones de costo lineal por partes de los generadores individuales para calcular el costo total de producción. El solucionador de **OPF** con aprendizaje aumentado, es decir, la integración de **ML** y ecuaciones de red, garantiza que se satisfagan las restricciones relacionadas con las leyes físicas. En cuanto al costo o valor de la función objetivo, el modelo de aprendizaje aumentado produce la solución **OPF** basada en el modelo entrenado. Dado que la capacitación se realiza con las soluciones **OPF** generadas que minimizan el costo, también se espera que el solucionador de aprendizaje aumentado minimice el costo.

4.2. Selección del algoritmo de machine learning

Los algoritmos de **ML** son esenciales para llegar a una solución a cualquier programa o problema informático. Cuanto mayor sea la eficiencia del algoritmo, mayor será la velocidad de ejecución. Los algoritmos se desarrollan en base a los enfoques matemáticos conocidos. **RF** y **DT** son algoritmos utilizados para problemas relacionados con la clasificación y la regresión. Ayudan a manejar grandes porciones de datos que requieren algoritmos rigurosos para ayudar a tomar mejores decisiones. La diferencia fundamental entre **RF** y **DT** es que; **DT** son gráficos que ilustran todos los resultados posibles de una decisión utilizando un enfoque de ramificación. Mientras que, la salida del algoritmo **RF** es un conjunto de árboles de decisión que funcionan de acuerdo con la salida. La desventaja de implementar el algoritmo de **RF** es que no se puede visualizar el modelo final, y si no tiene suficiente memoria de procesamiento o el conjunto de datos con el que se está trabajando es muy grande, pueden tomar mucho tiempo para llegar a una solución. Por otro lado, el algoritmo de **DT** es más complicado porque son combinaciones de árboles de decisión. Al construir un modelo de algoritmo de **DT**, se tiene que definir cuántos árboles hacer y cuántas variables se necesitan para cada nodo. En general, **DT** mejorará el rendimiento y hará que las predicciones sean más estables, pero también ralentizarán la velocidad de cálculo haciendo un modelo más lento. Según lo estudiado en [4, 14, 15, 31], y detallado en el Capítulo 2 en

comparación con los modelos **RF** y **DT**, **ELM** tiene menos parámetros para ajustar su red. El algoritmo usa una estructura multicapa con una sola capa de neuronas oculta **SLFN**, e inicia aleatoriamente los pesos que unen la capa de entrada y la capa oculta. De esta manera, sólo será necesario optimizar los pesos que unen a estas dos capas. Para optimizar estos últimos pesos se utiliza la matriz pseudoinversa de Moore-Penrose. **ELM** es un método más rápido que **RF**, **DT** y el método de descenso por gradiente que se usa de forma clásica [16, 28]. **ELM** disminuye notablemente el tiempo computacional empleado para ajustar los pesos debido a que no utiliza ningún método de búsqueda para los coeficientes ocultos. La Tabla 4.1 detalla algunas características de cada algoritmo para tener una mejor perspectiva sobre la funcionalidad de cada modelo.

RF	DT	ELM
Al construir un bosque aleatorio, el número de filas se selecciona al azar.	Al construir varios árboles de decisión el resultado se descubre por un diagrama.	Usa una estructura multicapa con una sola capa de neuronas oculta. SLFN
Combina dos o más árboles de decisión juntos.	La decisión es una colección de variables o conjunto de datos o atributos.	Inicia aleatoriamente una sola vez los pesos que unen la capa de entrada y la capa oculta.
Resultados precisos	Resultados menos precisos	Resultados precisos
Reduce las posibilidades de sobreajuste	Tiene la posibilidad de sobreajuste	Reduce el sobreajuste
Es más complicado de interpretar.	Es simple, por lo que es fácil de leer y comprender.	Es fácil de interpretar
Demasiado lento	Lento	Muy Rápido
Se necesita generar, procesar y analizar árboles, por lo que este proceso es lento, puede demorar una hora o incluso días.	No es preciso, pero se procesa rápido, lo que significa que es rápido de procesar.	Necesita menos uso computacional y es preciso
Usa método de búsqueda para encontrar su solución	Usa método de búsqueda para encontrar su solución	Usa método analítico, la matriz pseudoinversa de Moore-Penrose.

Tabla 4.1: Comparación principal entre **RF** y **DT**, **ELM**

4.3. Generación de conjuntos de datos de entrada

Como se explicó en la Figura (4.1), los conjuntos de datos de entrada se obtienen de los casos de prueba **IEEE** que incluyen una distribución de carga pre-calculada dadas (es decir, carga de potencia real P_d y reactiva Q_d y magnitud de ángulo θ y voltaje V), Figura (4.2) [4]. Se consideró estos conjunto de datos de entrada ya que el problema **ACOPF** es una tarea de regresión donde los puntos de ajuste del generador se estiman a partir del perfil de demanda de la red. La fijación de la inyección de potencia activa en todos los generadores, afecta drásticamente la viabilidad de la solución. De igual manera se seleccionó el conjunto de datos de voltaje y ángulo, ya que están relacionadas con la ley física y no se contradicen entre sí, lo que garantiza la viabilidad de las soluciones **OPF**. Cada conjunto de datos se convierte a valor por unidad [p.u] y luego se normalizan como se detalla en la ecuación (4.4), esto con la finalidad de que el entrenamiento de la red neuronal sea más eficiente y así reducir su redundancia. Con respecto al valor del ángulo se convierte primero a radianes para una mejor precisión de la predicción, y luego se traza en grados para una mejor observabilidad. Los casos de la **IEEE** que

se van a utilizar en este trabajo son: case300, case500 y case3375wp, en la Tabla 4.2 se detalla sus características. Cada *case* representa un sistema eléctrico diferente con una estructura diferente.

$$vectorN = \frac{vector}{norm(vector)} \quad (4.4)$$

Donde, se divide el *vector* por su norma *norm()* y el resultado se asigna como el nuevo valor del vector ya normalizado *vectorN*.

Red	Num de Generadores	Num de buses	Num de ramas
Case300	69	300	411
Case500	90	500	597
Case3375wp	596	3374	4161

Tabla 4.2: Características de redes de prueba seleccionadas

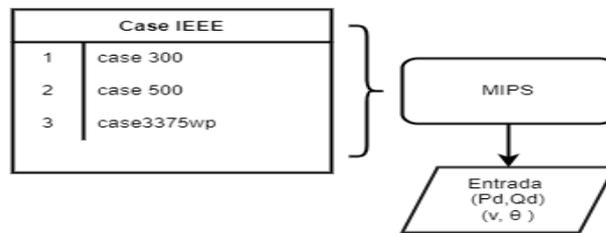


Figura 4.2: Generación de conjuntos de datos de entrada

El algoritmo de aprendizaje automático **ELM** no tiene la capacidad de discriminar entre casos factibles e inviables. Entre todas las entradas, no todas podrían proporcionar una solución viable y solo las soluciones de casos convergentes se utilizan para el entrenamiento y las pruebas del modelo **ML**. También se excluyen los casos en los que el solucionador **MIPS** no produce una solución factible. Para reducir aún más la tarea de mapeo, se excluye el ángulo de holgura del bus, ya que se toma como referencia al resolver **OPF**. En algunos casos, la demanda de algunos buses de carga permanece constante, lo que no afecta el rendimiento de **ML** y, por lo tanto, se excluyen del conjunto de variables de entrada. Todos los parámetros de los conjuntos de entrada fueron adquiridos tras una simulación **OPF**, con el fin de tener valores aceptables para el entrenamiento del algoritmo **ELM**. Dado que los parámetros de los casos de prueba se encuentran en escalas de magnitud diferente, se realiza una normalización de datos. Para evaluar el rendimiento de la **ML** se realiza dos escenarios:

- Escenario 1: Los perfiles de carga de bus escalados se utilizan para probar el método **OPF** de aprendizaje aumentado.
- Escenario 2: la carga del bus escalada se agrega mediante ruidos gaussianos con una media de cero y una desviación estándar del 5 %.

4.4. Implementación del algoritmo **ELM** en **ACOPF**

En esta sección se implementa el pseudocódigo [4], en donde la matriz con los datos de entrada se divide en dos partes una de entrenamiento (X_{tr}, Y_{tr}) y otra de validación (X_{test}, Y_{test}) que corresponde al 80 % y 20 % respectivamente. Para determinar el número de neuronas \tilde{N} en la capa oculta se establece inicialmente un número elevado de neuronas ($H \geq N$) y mediante el algoritmo **Least Angle Regression (LARS)** descrito en

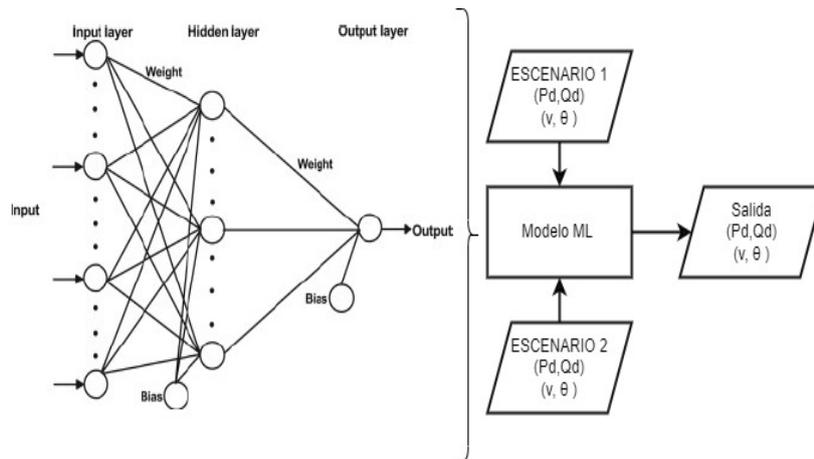


Figura 4.3: Implementación del algoritmo ELM

[25], se elimina aquellas neuronas que no son útiles para resolver el problema de mínimos cuadrados dejando un número aceptable para empezar el entrenamiento. La generación de los pesos para la capa de entrada se obtiene aleatoriamente en relación al número de neuronas de entrada H , el cual está fundamentado en MLP. Tras inicializar de manera aleatoria los pesos de la capa de entrada, los pesos de salida se obtienen de manera analítica mediante el cálculo de la matriz pseudo-inversa Moore-Penrose descrita en la sección (2.6.1). Para la función de activación de entrada, y la función de activación de salida (en este trabajo se utiliza la misma función de activación). Dado que uno de los conjuntos de parámetros de entrada es la carga reactiva Q_d y a menudo toma un valor negativo, se selecciona una función de activación "sigmoidea". Finalmente, Cuando la red ya es entrenada con los datos que corresponden al escenario 1 es validado y puesto a prueba con los datos del escenario 2, el resultado es ingresado en un solucionador MIPS para establecer si el entrenamiento es óptimo y que tiene convergencia y realizar el análisis de resultados, figura (4.3).

4.5. Análisis de datos de salida

El análisis de los datos de salida de la red se detallan de una forma más amplia en el siguiente capítulo (5), pero la idea general del análisis y la obtención de resultados se muestra en la Figura (4.4). Para interpretar las gráficas, cada dato fue convertido en su valor base sea magnitud de voltaje en [p.u] o ángulo de voltaje en grados, de igual manera la potencia activa y reactiva.

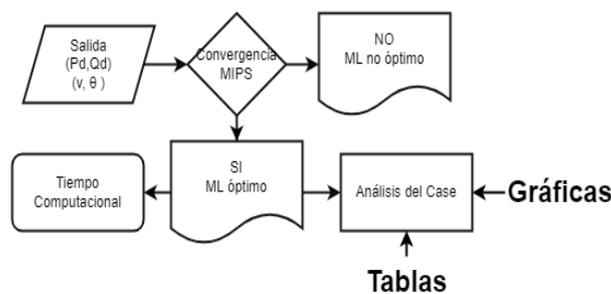


Figura 4.4: Análisis de datos de salida del algoritmo ELM

Resultados y discusión

Este capítulo muestra el resultado del análisis y estudio de los objetivos planteado desde un principio sobre el desarrollo de un algoritmo basado en **ML** para predecir la convergencia y el valor de costo de producción de la función objetivo de un sistema eléctrico. Como primer y principal objetivo planteado, es estudiar y revisar el estado del arte con la finalidad de lograr evaluar y desarrollar el algoritmo **ELM** basado en **ML**, este algoritmo se comparó con otros métodos de aprendizaje automático que son **RF** y **DT**. Siendo el algoritmo **ELM** el más adecuado para este trabajo.

Antes de entrar en detalle la definición del conjunto de datos de entrada para el entrenamiento del algoritmo **ELM** se basó en las variables que más representan en un sistema eléctrico y las que más influyen en su solución. Estas variables son la magnitud de voltaje y ángulo, potencia activa y reactiva. Con cada conjunto de datos se analizó el algoritmo **ELM** tenga convergencia, y estos resultados fueron expuestos a tres métricas, (1) **Error medio absoluto (MAE)**, (2) **Raíz del Error Cuadrático Medio (RMSE)** y (3) el **R-cuadrado (R²)**, para evaluar el rendimiento de **ELM** y verificar los rangos de operación de cada conjunto de datos.

El capítulo está organizado de la siguiente manera. La sección 5.1, representa la caracterización de las variables e indicadores de entrada y salida del sistema para verificar su convergencia en base a los casos de la **IEEE**. En la sección 5.2, representa la implementación del algoritmo de **ELM** para el uso de la **ACOPF**. En la sección 5.3 trata acerca del rendimiento y el uso del costo computacional que conlleva el algoritmo en tener una solución y el costo de producción de la función objetivo del flujo de potencia.

5.1. Rendimiento de **ELM**

La eficiencia del enfoque propuesto de aprendizaje automático **ELM** se basa en gran medida en la predicción de extremo a extremo de los parámetros de entrada. Las predicciones se realizan en conjuntos de prueba convergentes para los casos de la **IEEE**: case300, case500 y case3375wp. Antes de realizar los escenarios de entrenamiento y prueba todos los conjuntos de datos fueron normalizados para tener un control sobre el entrenamiento de la red neuronal. Los resultados como es el caso de la predicción del voltaje y ángulo se realiza por unidad y radianes respectivamente para una mejor precisión de la predicción, para la visualización el ángulo se traza en grados para una mejor apreciación. Para la predicción de potencia activa y reactiva se realiza por unidad [p.u], para tener un mejor manejo del entrenamiento y una mejor interpretación de los resultados. Para

los casos de prueba, los límites inferior y superior de los ángulos son -180° y $+180^\circ$ ($-\pi$ y $+\pi$ en radianes), respectivamente.

Dado que los resultados de la predicción de la magnitud del voltaje son 0,9 y 1,1 por unidad [p.u.], y del ángulo son -100 , 100 grados y están dentro de los límites en la mayoría de los casos, se podría evitar la violación de restricciones severas. En las Figuras (5.1), (8.1), (8.2) muestran los errores absolutos medios de cada entrenamiento y validación de MAE de los tres casos de prueba de la IEEE. Como se puede observar el error es mínimo lo cual indica que la red ELM cuenta con un entrenamiento eficiente. Los valores en los que tanto el entrenamiento como los errores de validación son consistentes, se seleccionan para el entrenamiento del modelo final, los conjunto de datos para el entrenamiento y validación se divide en una proporción del 80 % al 20 % cada uno.

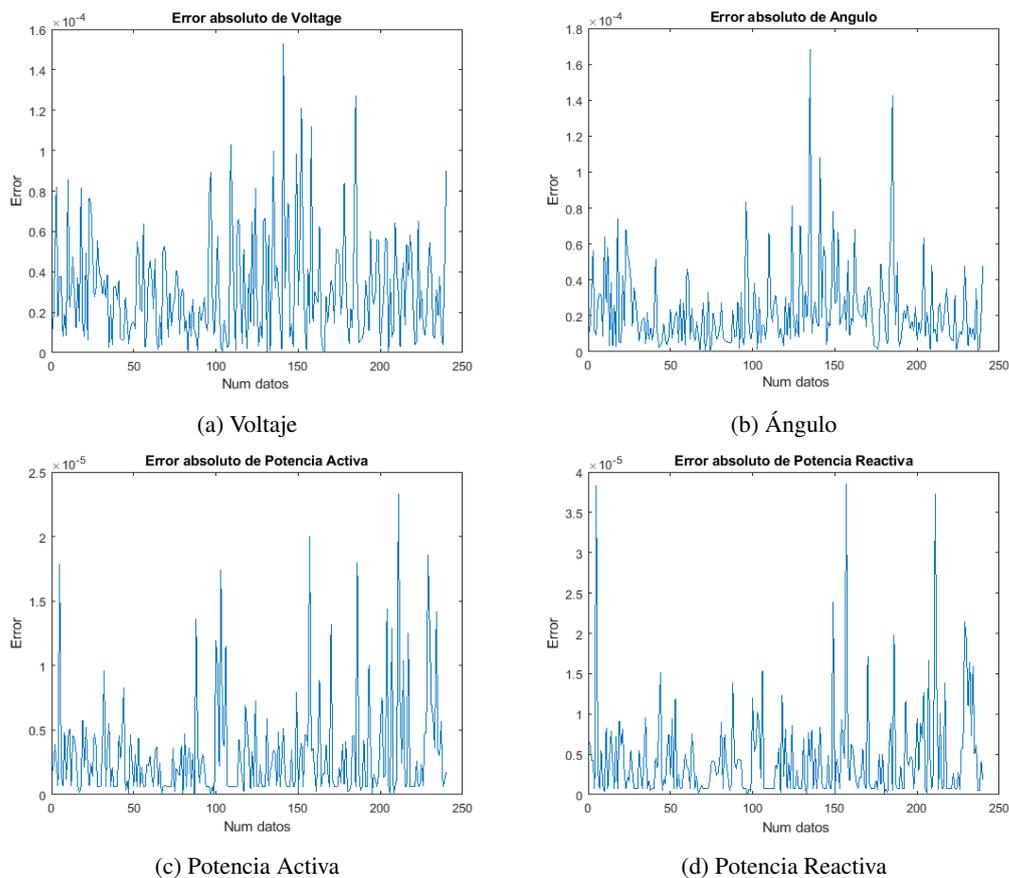


Figura 5.1: Error absoluto case300

Después de evaluar el error absoluto de cada caso de la IEEE para cada parámetro (Potencia activa y reactiva; voltaje y ángulo), se evalúa la métrica del error, para ello se calcula RMSE este cálculo se normaliza con el fin de obtener un valor entre 0 y 1. Además, se calcula el error promedio de entrenamiento y prueba, esto se formula en la Ecuación (5.1).

$$Error = \frac{1}{m} \sum_{i=1}^m \frac{\|Y - P\|_2^2}{\|P\|_2^2} \quad (5.1)$$

Se implementa la Ecuación (5.1) y se obtiene la precisión máxima de la prueba en base a la métrica de

error descrita anteriormente. Esto se muestra en la Tabla (5.1), (8.1) y (8.2) donde se aprecia una precisión de entrenamiento tan alta debido a la falta de valores atípicos en las soluciones de cada escenario.

Parámetro	Precisión de entrenamiento	Exactitud de la prueba
Potencia Activa	98.3 %	98.8 %
Potencia Reactiva	98.1 %	98.7 %
Voltaje	99.1 %	99.5 %
Ángulo	99.1 %	99.4 %

Tabla 5.1: Errores de prueba y entrenamiento del case300

5.2. Evaluación de ELM sobre ACOF

La mayoría de las magnitudes de voltaje predichas de ELM para el caso de prueba 500 están dentro de los límites técnicos en el Escenario 1, Figura (5.2) aunque se observan algunas violaciones. Las Figuras (5.2a), (5.2c), (5.2b) y (5.2d) muestran los resultados de la predicción del ángulo y voltaje. se observa que los ángulos de voltaje predichos están dentro de sus límites operativos y están dentro de los límites en la mayoría de los casos, y por lo tanto, no existen violaciones de restricciones severas.

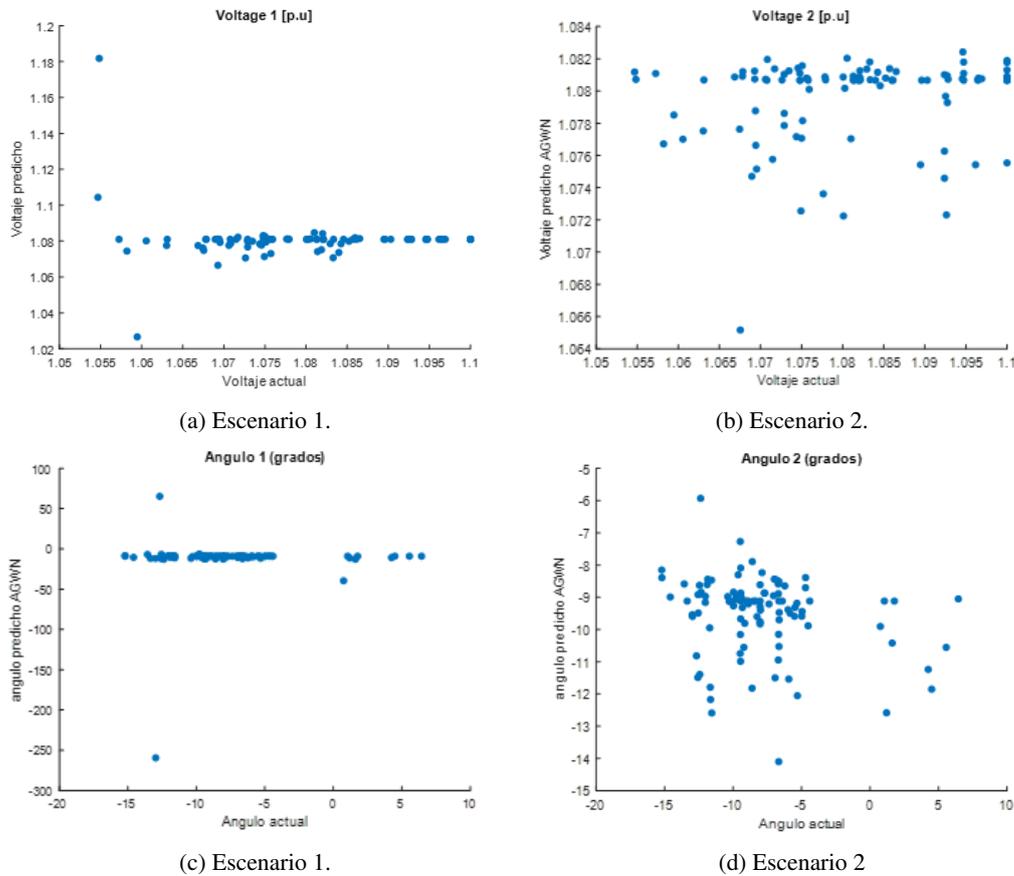


Figura 5.2: Representación gráfica de valores de voltaje y ángulo con ruido correspondiente al Escenario 1 y 2.

En las Tablas 5.2 y 5.3 se resume una descripción detallada del rendimiento estadístico de los modelos ML en

base a tres métricas, (1) **MAE**, (2) **RMSE** y (3) el R-cuadrado (R^2). Los resultados del escenario 2 son inferiores a los del escenario 1, que se espera debido a la variación adicional del 5 % en el conjunto de datos. Pero en general, el rendimiento del método de aprendizaje **ELM** sigue siendo razonable y aceptable. También se observa el tiempo de entrenamiento que tomo a cada case de prueba al llegar a una solución lo que es relativamente aceptable lo cual representa un costo de uso computacional menor. Este tiempo es comparado con el tiempo que lleva el método **MIPS** en tener una solución y se resume en la Tabla 5.5.

Entrenamiento y validación con voltaje y ángulo					
Case	Escenario	MAE (deg)	RMSE (deg)	R^2	TIEMPO (seg)
CASE300	1	0.0163	0.1135	0.2161	0.2188
CASE500	1	0.0102	0.0918	0.3152	0.2500
CASE3375WP	1	0.0015	0.0347	0.4597	1.2031
Entrenamiento y validación con potencia activa y reactiva					
CASE300	1	0.0163	0.1135	0.2161	0.2188
CASE500	1	0.0102	0.0918	0.3152	0.2500
CASE3375WP	1	0.0015	0.0347	0.4597	1.2031

Tabla 5.2: Detalle del rendimiento estadístico para la predicción de potencia activa y reactiva, y magnitud del voltaje y ángulo.

Entrenamiento y validación con voltaje y ángulo					
Case	Escenario	MAE (deg)	RMSE (deg)	R^2	TIEMPO (seg)
CASE300	2	0.0138	0.0980	0.0891	0.2813
CASE500	2	0.0114	0.0853	0.2443	0.1719
CASE3375WP	2	0.0012	0.0341	0.0149	1.1875
Entrenamiento y validación con potencia activa y reactiva					
CASE300	2	0.0166	0.1113	0.1196	0.1875
CASE500	2	0.0095	0.0879	0.0664	0.1094
CASE3375WP	2	0.0017	0.0342	0.1387	1.2344

Tabla 5.3: Detalle del rendimiento estadístico para la predicción de potencia activa y reactiva, y magnitud del voltaje y ángulo con ruido del 5 %.

Aunque el algoritmo **ELM** predijo la potencia y el voltaje generados con alta precisión, algunos de los resultados violaron las restricciones de red del problema de optimización. Las restricciones son que el voltaje debe estar dentro del 6 % de la clasificación nominal y la potencia generada debe ser igual a la potencia demandada en todos los nodos. Menos del 60 % de las predicciones violaron en algún lugar de la red cuando se intentó predecir el voltaje conjuntamente con la potencia, es decir, al menos una restricción en algún lugar de la red no cumplió con las leyes físicas de Kirchhoff o a su vez con el rango de operación del voltaje o de potencia, pero esto no influyó en que la red no converja. Para solventar este incumplimiento en la red se intentó forzar para que el algoritmo cumpla de cualquier manera las restricciones en la salida; sin embargo, dado que la potencia y el voltaje no se consideran juntos, fue imposible satisfacer ambos simultáneamente. Por ejemplo, fijar el algoritmo para que solo genere voltajes válidos satisfaría las restricciones de voltaje en todas partes, pero cuando se calculó la potencia, se encontró que violaba en ciertos lugares de la red y de igual manera cuando se evaluó de forma inversa. Al final, el beneficio fue insignificante. En lugar de utilizar los valores predichos directamente, se pueden utilizar como punto de partida del problema de optimización. Tener un punto de partida muy cercano hace que el algoritmo **ELM** converja y a su vez tenga valores más exactos. Esto es muy preciso

porque la energía demandada y generada no cambia mucho dentro del marco de tiempo. Sin embargo, con una mayor penetración de energías renovables, la energía generada y demandada puede volverse muy variable dentro de este marco de tiempo.

5.3. Comparación de tiempo computacional

La eficiencia computacional es uno de los objetivos principales de este trabajo y factor impulsor para emplear **ML** para resolver **ACOPF**. Para evaluar la ventaja computacional del método propuesto de **ELM**, el tiempo de resolución entre el método propuesto **ELM** y **MIPS** se compara en la Tabla 5.5. Ambos escenarios se simularon en una computadora con un procesador Intel Core i5-1135G7 de 11th Generación, disco SSD y memoria RAM de 12 GB. Dado que el número de muestras consideradas en ambas redes de prueba es diferente, se considera el tiempo medio de solución por instancia de carga en lugar del tiempo total de solución. El tiempo medio de solución también representa la eficiencia computacional de resolver una instancia **ACOPF**. Para los enfoques de aprendizaje **ELM**, el tiempo de solución consiste tanto en la predicción **ML** como en el tiempo de posprocesamiento de la ecuación de red. El tiempo computacional de posprocesamiento es similar entre los escenarios 1 y 2. El factor de aceleración del modelo **ML** se calcula en base al tiempo computacional de referencia de **MIPS**. Se observa que el modelo **OPF** con aprendizaje **ML** es aproximadamente de 15 a 20 veces y de 70 a 100 veces más rápidos que **MIPS** para las redes de 500 buses y 3375 buses, respectivamente. Para medir la calidad de la solución de **ELM**, se adopta una métrica de verificación de la optimización O , formulada en la Ecuación (5.2) [16]. Donde P es el número total de muestras originalmente convergentes en el conjunto de datos de prueba. La optimización del enfoque propuesto se resume en la Tabla 5.6, y se considera en un rango aceptable de optimización. El costo de producción de la función objetivo del flujo óptimo de potencia se resumen en la tabla 5.4, donde se detalla el costo de cada case de prueba de la **IEEE**. Este valor es obtenido luego de ejecutar el solucionador **MIPS**, como el costo del generador se almacena en una matriz en cada case y este no es manipulado, este valor varía muy poco por la varianza del 5 % que se agrega en el escenario 2.

Case	Escenario	\$/hr
CASE300	1	719725.11
CASE500	1	72578.30
CASE3375WP	1	7412030.68
CASE3375WP	2	7412030.68

Tabla 5.4: Costo de producción de la función objetivo del flujo óptimo

$$O = \frac{1}{P} \sum_{n=1}^P \left(\frac{C_g(P_{Propuesta}) - C_g(P_{Actual})}{C_g(P_{Actual})} \right) \quad (5.2)$$

Case	Escenario	Tiempo promedio de solución (s)	<i>Speedup</i> factor
CASE300	1	0.028	15.68
CASE500	1	0.044	14.00
CASE3375WP	1	0.150	70.20
CASE3375WP	2	0.213	68.07

Tabla 5.5: Comparación de tiempo computacional entre el solucionador **OPF** con aprendizaje aumentado y **MIPS**.

Case	Escenario	Optimización
CASE300	1	0.0005
CASE500	1	0.011
CASE3375WP	1	0.004
CASE3375WP	2	0.0005

Tabla 5.6: Resultados de optimización de los enfoques **OPF** de **ELM**

Conclusiones

Este trabajo desarrolló un método de aprendizaje automático para resolver el flujo de energía óptimo de **CA**, que integra las ecuaciones de la red de energía. Luego de estudiar y analizar el estado del arte se implementó un algoritmo **ELM** y se aplicó a diferentes casos de prueba de la **IEEE** (case300, case500 y Case3375wp) teniendo una red entrenada exitosa. Después de cada entrenamiento el algoritmo mostró una precisión y una exactitud de entrenamiento del 98 %, y un rango de error promedio de entrenamiento y prueba de 0.004 % hasta 0.016 % siendo un algoritmo eficiente y preciso para resolver problemas en diferentes escenarios. Así, las principales ventajas que se han podido obtener de este algoritmo son las siguientes:

- La velocidad de aprendizaje del algoritmo **ELM** es rápida, demostrando ser capaz de reducir significativamente el tiempo de entrenamiento y predicción, en comparación con los algoritmos clásicos de **ML**.
- **ELM** destaca por su simplicidad de configuración de los parámetros que debe ajustar para que afecten al rendimiento del modelo.
- La integración de ecuaciones de red asegura la satisfacción de las leyes físicas inherentes al problema de **OPF**.

En este trabajo se estudiaron y compararon tres métodos de aprendizaje automático: **RF**, **DT** y **ELM**. La aplicación del algoritmo **ELM** integrando las ecuaciones de la red de energía para resolver el problema de flujo de óptimo (**OPF**) de **CA** produjo un resultado casi óptimo; el resultado no fue óptimo debido a que ciertos lugares y eventos en la red no se cumplieron con su totalidad. La integración de ecuaciones de red ayudó a validar el rendimiento del algoritmo **ELM** para satisfacer las leyes físicas inherentes al problema de **OPF**. Los resultados en las redes de prueba de la **IEEE** (case300, case500 y case3375wp) han mostrado una violación mínima de las restricciones y por ende una pérdida de optimización mínima, también se encontró que el algoritmo **ELM** mejora la eficiencia computacional **OPF** entre 15 y 100 veces, dependiendo del tamaño de la red.

Los resultados que tenemos son prometedores y el algoritmo **ELM** se puede utilizar para reducir los tiempos de cálculo durante las simulaciones. Sin embargo, nuestros datos de entrenamiento son relativamente pequeños y poco realistas. El caso de prueba de la **IEEE** case300 convergió a la solución óptima en milisegundos en una computadora estándar, por lo que no proporciona un buen punto de referencia para el tiempo computacional de los problemas típicos de flujo de energía óptimo en la vida real. También por la independencia entre conjuntos

de datos como la potencia y voltaje, no se logró obtener una solución que satisfaga en ambas situaciones ya que entre estos dos conjuntos de datos se predice de forma independiente, pero existe una relación cuadrática entre el voltaje y la potencia que debe explorarse. Esto podría ayudar a reducir la cantidad de violaciones de restricciones que tienen nuestras predicciones.

Como trabajo futuro se utilizará el algoritmo [ELM](#) para resolver redes más grandes; la idea es analizar el comportamiento y que tan bien generaliza los resultados. Además, se puede realizar una implementación a mayor escala en una red que cuente con cargas de automóviles eléctricos y una gran penetración de energías renovables. Un punto a evaluar en sistemas grandes es el tiempo computacional de la solución. Los resultados de este trabajo también resultan alentadores para utilizar el aprendizaje como una herramienta para encontrar rápidamente soluciones a problemas de optimización no convexos. En aplicaciones potenciales donde aprender y encontrar la solución óptima mientras se asegura la viabilidad es difícil, se podría desarrollar o usar técnicas de optimización asistidas por aprendizaje o mejoradas por el aprendizaje. Finalmente, se puede aplicar los resultados a una red de distribución con almacenamiento y generación renovable distribuida. Esta complejidad adicional hace que el flujo de energía tradicional sea lento, pero predecir una solución cercana sería inmensamente beneficioso para las decisiones de control de almacenamiento.

A1

7.1. Código de Matlab

El código del desarrollo del algoritmo Extreme Learning Machine se encuentra disponible en el siguiente link.

<https://github.com/WalterAstudillo/TTE.git>

A2

8.1. Gráficas y Resultados

En la Tabla (5.1), (8.1) y (8.2), muestra que el algoritmo cuenta con un alta precisión, aunque en algunos de los resultados violaron las restricciones de red ya sea con voltaje y ángulo o potencias. Al final, el beneficio fue insignificante. En lugar de utilizar los valores predichos directamente, se pueden utilizar como un punto de partida del problema de optimización y así tener un inicio de entrenamiento muy cercano a la solución y hacer que el algoritmo converja en menos tiempo de cálculo.

Parámetro	Precisión de entrenamiento	Exactitud de la prueba
Potencia Activa	98.31 %	98.8 %
Potencia Reactiva	98.2 %	98.65 %
Voltaje	99.11 %	99.7 %
Ángulo	99.16 %	99.43 %

Tabla 8.1: Errores de prueba y entrenamiento del case500

Parámetro	Precisión de entrenamiento	Exactitud de la prueba
Potencia Activa	98 %	99 %
Potencia Reactiva	98.12 %	98 %
Voltaje	99 %	99.8 %
Ángulo	99.3 %	99 %

Tabla 8.2: Errores de prueba y entrenamiento del case3375wp

En las figuras (8.1), (8.2) muestra los errores absolutos medios de entrenamiento y validación MAE de los tres casos de la IEEE. Los valores en los que tanto el entrenamiento como los errores de validación son consistentes, se seleccionan para el entrenamiento del modelo final, los conjunto de datos para el entrenamiento y validación se divide en una proporción del 80 % al 20 % respectivamente.

En la figura (8.3), se observa el comportamiento del voltaje predicho vs el voltaje actual de cada nodo del

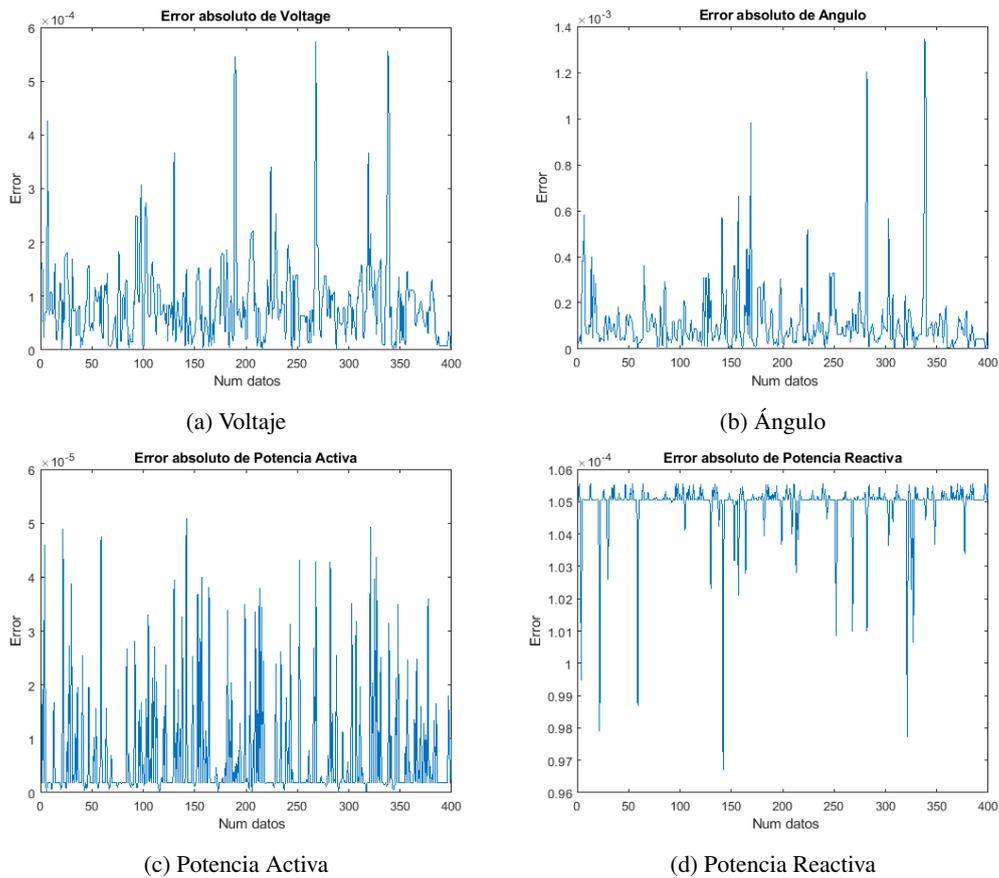


Figura 8.1: Error absoluto case500

case 300, después de agregar ruido en los parámetros de entrada, este proceso corresponde al escenario 2 que se detalla en el capítulo 3. Como se observa el comportamiento del voltaje la red ELM responde muy bien y se estabiliza en un rango aceptable para la red.

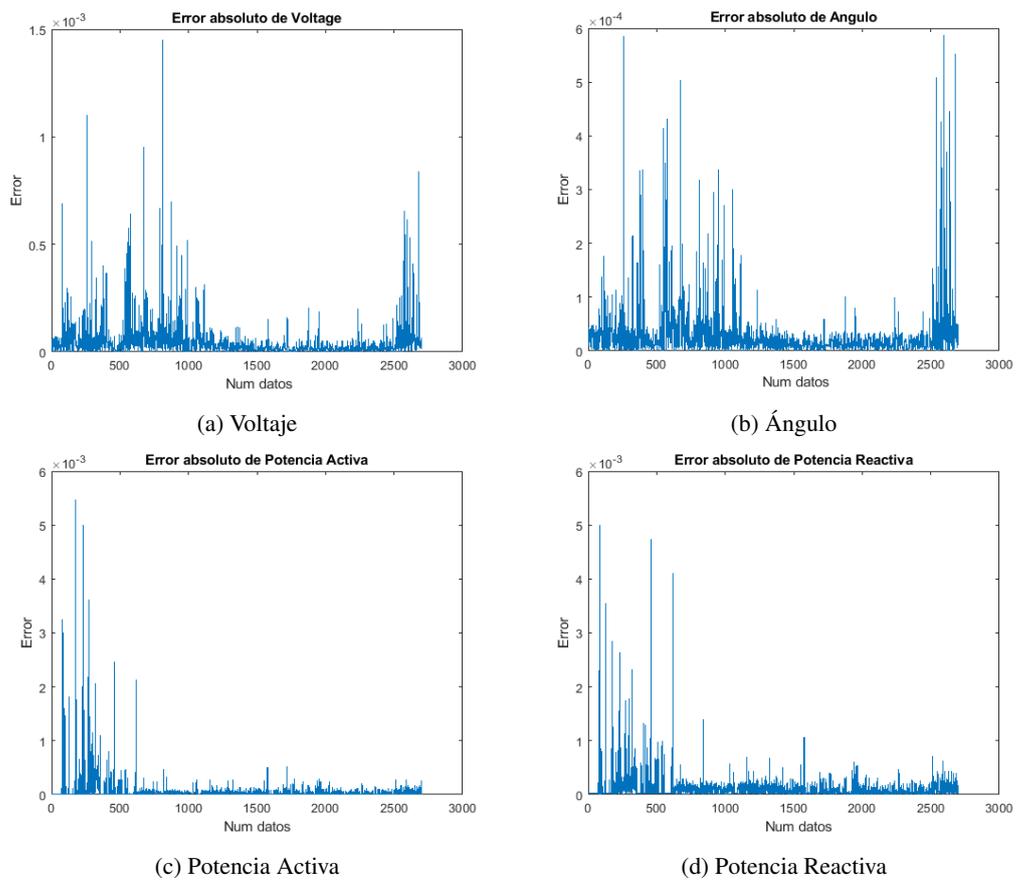


Figura 8.2: Error absoluto case3375wp

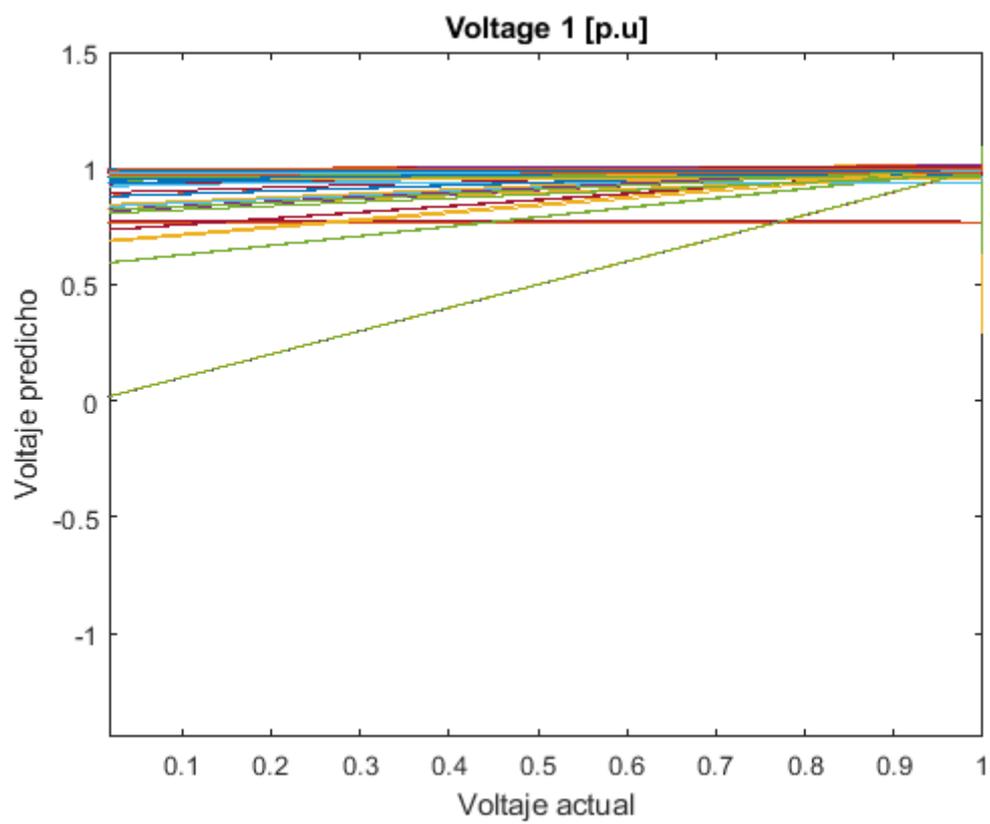


Figura 8.3: Iteraciones de Voltaje predico Vs Voltaje real

Bibliografía

- [1] A. V. Fiacco y G. P. McCormick, “Nonlinear Programming: Sequential Unconstrained Minimization Techniques,” 1968, reprinted by *SIAM Publications in 1990.*, num. December, 2017.
- [2] A. Wächter y L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” 1968, reprinted by *SIAM Publications in 1990.*, num. April, pp. 25–57, 2005.
- [3] C. Coffrin y P. Van Hentenryck, “Coffrin C, Van Hentenryck P. A linear-programming approximation of ac power flows,” *Commission (FERC)*, num. 19 May, p. 718–34., 2014.
- [4] N. Guha, Z. Wang, M. Wytock, y A. Majumdar, “Machine Learning for AC Optimal Power Flow,” *Submitted*, num. 19 Oct, pp. 1085–1092, 2019.
- [5] C. M. B. y O. R. P. C. Anya, “History of Optimal Power Flow and Formulations Optimal Power Flow Paper 1,” *Commission (FERC)*, num. oct, pp. 12–2, 2012.
- [6] L. SH, “Convex relaxation of optimal power flow—Part I: Formulations and equivalence,” *IEEE Trans Control Network Syst 2014*, num. 05 March, pp. 15 – 27, 2014.
- [7] C. Shilaja y K. Ravib, “Optimal Power Flow Using Hybrid DA-APSO Algorithm in Renewable Energy Resources,” *IEEE Trans Control Network Syst 2014*, num. June, pp. 1085–1092, 2017.
- [8] R. Canyasse, G. Dalal, y S. Mannor, “Supervised learning for optimal power flow as a real-time proxy,” *2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, num. 30 October, p. 1–5, 2017.
- [9] F. Fioretto, T. W. Mak, y P. Van Hentenryck, “Predicting AC Optimal Power Flows: Combining Deep Learning and Lagrangian Dual Methods Authors Ferdinando Fioretto,” *2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, num. April, pp. 630–637, 2020.
- [10] A. S. Zamzam y K. Baker, “Learning optimal solutions for extremely fast AC optimal power flow.” *In 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids*, num. November, pp. 1–6, 2020.
- [11] J. Rahman, C. Feng, y J. Zhang, “A learning-augmented approach for AC optimal power flow, International Journal of Electrical Power & Energy Systems,” *International Journal of Electrical Power & Energy Systems, Volume 130, 2021, 106908, ISSN 0142-0615.*, num. September, pp. 1–6, 2021.
- [12] P. J. García Laencina, R. Verdú Monedero, J. Larrey Ruiz, J. Morales Sánchez, y J. L. Sancho Gómez, “Nuevas tendencias en redes neuronales artificiales: extreme learning machine.” *Grupo Teoría y Tratamiento de la Señal (GTTS)*, num. Jun, pp. 1–6, 2009.

- [13] T. Navidi, S. Bhooshan, y A. Garg, “Predicting solutions to the optimal power flow problem. Project Report,” *Project Report, cs229.stanford.edu*, pp. 1–6, 2016.
- [14] X. Pan, M. Chen, T. Zhao, y S. H. Low, “DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems,” *preprint arXiv:2007.01002*, num. Sep, pp. 12–2, 2021.
- [15] Z. Yang, H. Zhong, Q. Xia, y C. Kang, “Fundamental review of the opf problem: Challenges, solutions, and state-of-the-art algorithms,” *J Energy Eng 2018*, num. Jun, p. 144(1): 04017075, 2018.
- [16] F. M. Castro y P. E. Jojoa, “Entrenamiento Comprimido Basado en Máquinas de Aprendizaje Extremo,” *Información tecnológica, 30(4)*, num. ago, pp. 227–236, 2019.
- [17] J. Rahman, C. Feng, y J. Zhang, “Machine learning-aided security constrained optimal power flow,” *2020 IEEE Power and Energy Society General Meeting (PESGM)*, num. 16 December, pp. 1944–9925, 2020.
- [18] F. Izaurieta y C. Saavedra, ““Redes Neuronales Artificiales”, Departamento de Física, Universidad de Concepción,,” *Concepción, Chile*, 2000.
- [19] M. Gestal Pose, “Introducción a las Redes de Neuronas Artificiales,” *Dpto. Tecnologías de la Información y las Comunicaciones. Universidade da Coruña*, 2009.
- [20] D. C. Ruiz y G. Q. Gavino, “Aplicación del algoritmo Backpropagation de redes neuronales para determinar los niveles de morosidad en los alumnos de la Universidad Peruana Unión,” *Revista de Investigación Business Intelligence, 1(2)*, pp. 171–175, 2011.
- [21] C. Cortes y V. Vapnik, ““Support vector networks”, Machine Learning ,” *vol. 20, no. 3*, pp. 273–297, 1995.
- [22] D. Lowe, “Adaptative radial basis function nonlinearities and the problema of generalisation,” *in Proceedings of First IEE International Conference on Artificial Neural Networks*, pp. 171–175, 1989.
- [23] G. S. K. B y X. D, “Empirical asset pricing via machine learning ,” *National Bureau of Economic Research*, pp. 489–501, 2018.
- [24] L. Breiman, “Random forests,” *Machine learning, 2001 - Springer*, num. 45(1), pp. 5–32, 2001.
- [25] B. Jiménez, “Predicción del caudal promedio horario de la estación hidrológica Palmar, utilizando modelos de Machine Learning basados en Árboles de decisión,” *JVD Ospina - Cuaderno activa, 2021 - fucsalu-ojs3.metabiblioteca.com.com*, 2020.
- [26] J. G. W. D. H. T y Tibshirani, “An Introduction to Statistical Learning,,” *Springer Texts in Statistics. Springer New York*, num. volume 103 of Springer, 2020.
- [27] G.-B. Huang, Q.-Y. Zhu, y C.-K. Siew, “Extreme learning machine: theory and applications.” *Neurocomputing 2006;70(1–3)*:, num. December, p. 489–501, 2006.
- [28] C. R. Rao y S. K. Mitra, “Generalized Inverse of Matrices and It’s Applications. Wiley ,” *Neurocomputing, volume 70, Journal of the Royal Statistical Society: Series A (General)*, pp. 489–501, 1972.
- [29] K. Akyol, “Comparing of deep neural networks and extreme learning machines based on growing and pruning approach,” *Expert Systems with Applications*, vol. 140, p. 112875, 2020. [En línea]. Disponible: <https://www.sciencedirect.com/science/article/pii/S0957417419305858>

- [30] G. B. Huang, Q. Y. Zhu, y C. K. Siew, “Extreme learning machine: Theory and applications ,” *Neurocomputing*, volume 70,, pp. 489–501, 2006.
- [31] W. Huang, X. Pan, M. Chen, y S. H. Low, “DeepOPF-V: Solving AC-OPF Problems Efficiently,” *IEEE Transactions on Power Systems (Volume: 37, Issue: 1, Jan. 2022)*, num. 21 September, pp. 800 – 803, 2021.
- [32] J. Rahman, C. Feng, y J. Zhang, “A learning-augmented approach for AC optimal power flow,” *2020 IEEE Power and Energy Society General Meeting (PESGM)*, num. September, 2021.
- [33] E. Spyromitros-Xioufis, G. Tsoumakas, y W. Groves, “Multi-target regression via input space expansion: treating targets as inputs,” *Mach Learn 2016*, num. 19 February, p. 55–98, 2016.
- [34] R. F. M. Merino y C. I. Ñ. Chacón, “Bosques aleatorios como extensión de los árboles de clasificación con los programas R y Python,” *Interfases, ISSN-e 1993-4912, N°. 10, 2017.*, pp. 165–189, 2017.
- [35] D. Kocev, A. Naumoski, K. Mitreski, S. Krstić, y S. Džeroski, “Learning habitat models for the diatom community in Lake Prespa,” *Ecol Model 2010;221(2):330–7.*, num. 24 January, pp. 330–337, 2010.