APLICACIÓN DEL ALGORITMO ADABOOST.RT PARA LA PREDICCIÓN DEL INDICE COLCAP Y EL DISEÑO DE UN CONTROLADOR NO LINEAL

LAURA MARCELA REYES FAJARDO

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Proyecto Curricular Ingeniería Electrónica
Bogotá, Colombia
2017

APLICACIÓN DEL ALGORITMO ADABOOST.RT PARA LA PREDICCIÓN DEL INDICE COLCAP Y EL DISEÑO DE UN CONTROLADOR NO LINEAL

LAURA MARCELA REYES FAJARDO

Trabajo de grado para optar al título de:

Ingeniero Electrónico

Director:

ANDRES EDUARDO GAONA BARRERA

Profesor Asistente – Facultad de Ingeniería

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Proyecto Curricular Ingeniería Electrónica
Bogotá, Colombia
2017

Nota de aceptación

_____
_____
_____
_____
_____

_____

Firma del Jurado

Resumen

En este proyecto se lleva a cabo el desarrollo de un procedimiento basado en el algoritmo AdaBoost.RT con φ auto-adaptativo para la predicción del índice económico COLCAP y el control del sistema del péndulo invertido, además para poder comparar los resultados obtenidos se generó una solución a los mismos problemas por medio de Redes Neuronales.

El algoritmo AdaBoost ha sido muy usado en problemas de clasificación obteniendo muy buenos resultados con modelos de baja complejidad, motivo por el cual se ha generado un creciente interés en aplicar este algoritmo en problemas de regresión, por esto se desarrolló el algoritmo AdaBoost.RT con φ auto-adaptativo para resolver este tipo de problemas. Además no se presenta aplicación del algoritmo AdaBoost en problemas de control, lo que motivó este proyecto.

Para llevar a cabo la predicción de la serie económica, primero se lleva a cabo un análisis de la serie de tiempo del índice COLCAP comprobando la estacionalidad de esta por medio de test de raíz unitaria. Luego se seleccionan los parámetros de entrada del sistema predictor y se analizan por medio de análisis de relación gris y de componentes principales. A partir de la base de datos generada con los parámetros de entrada del sistema, se lleva a cabo el entrenamiento y validación del modelo predictor por medio de AdaBoost y de redes neuronales. Para el caso de AdaBoost, se realiza una variación de los parámetros propios para encontrar el modelo óptimo predictor y luego se lleva a cabo una comparación con el modelo predictor neuronal. Los resultados muestran que es posible realizar una predicción del índice COLCAP por medio de AdaBoost con un desempeño mayor al 98% pero se hace necesario realizar una sintonización de los parámetros del algoritmo para obtener un modelo óptimo, dado que el desempeño del modelo tiene una relación directa con los parámetros del algoritmo.

En cuanto al control del péndulo invertido, se generó el modelo a partir de las ecuaciones de variables de estado del sistema. Puesto que no se ha utilizado anteriormente el algoritmo de AdaBoost en problemas de control, se desea conocer si los modelos generados a partir de AdaBoost tienen la capacidad de controlar un sistema no lineal, como lo es el péndulo invertido. Por este motivo, se hace uso de un controlador de referencia para generar la base de datos de entrenamiento que luego es usada para llevar a cabo el entrenamiento de los controladores, tanto el neuronal como el basado en AdaBoost. El controlador AdaBoost logra estabilizar el péndulo dentro de un rango de estabilización definido aún ante la presencia de ruido. Al igual que en el caso de predicción, el desempeño del modelo tiene una relación directa con los parámetros propios del algoritmo, lo que hace que sea necesario llevar a cabo una sintonización de estos para obtener un modelo con buen desempeño.

Agradecimientos

A mi director Andrés Gaona por su constante guía, apoyo y dedicación durante el desarrollo del proyecto.

A mis padres por su apoyo incondicional, sus consejos y por siempre ayudarme a ser mejor persona e impulsarme y motivarme cada día a perseguir mis sueños.

A mis hermanos por su compañía y paciencia.

TABLA DE CONTENIDO

Lista de Figuras

Lista de Tablas

Índice de Abreviaturas

AB	AdaBoost
ADF	Augmented Dickey-Fuller test
ANFIS	Adaptive Neuro Fuzzy Inference System
ANN:	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
BVC	Bolsa de Valores de Colombia
CCI	Commodity Channel Index
DWT	Direct Wavelet Transform
ELM	Extreme Learning Machine
FLDT	Feedback Linerarization and Decoupling Transform
GA	Genetic Algorithms
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
GRG	Grey Relational Grade
IA	Inteligencia Artificial
K	Stochastic Oscillator
KPSS	Kwiatkowski – Phillips - Schmidt-Shin test
LQR	Linear Quadratic Regulator
MACD	Moving Average Convergence Divergence
MAD	Mean Absolute Deviation
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MRE	Mean Relative Estimation
MSE	Mean Squared Error
NARX	Nonlinear Autoregressive Exogenous Model
NMSE	Normalized Mean Square Error
OS	OverShoot
PD	Control Proporcional Derivativo
PID	Control Proporcional Integral Derivativo
RBF	Radial Basis Function

RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROC	Rate Of Change
RSI	Relative Strength Index
SA	Simulated Annealing
SVM	Support Vector machine

1 GENERALIDADES

Se presentan los aspectos del proyecto realizado como el planteamiento del problema, los objetivos que se esperan cumplir, la justificación y los alcances y limitaciones del mismo.

1.1 PLANTEAMIENTO DEL PROBLEMA

En estudios, investigaciones o inclusive en la vida cotidiana, se presentan situaciones en las cuales se obtiene información relevante de distintas fuentes y es necesario determinar la relación que existe entre ellas con el fin de tomar decisiones en futuros eventos. Este proceso se conoce en la estadística como Análisis de regresión. Tiene aplicación en múltiples campos del conocimiento en donde es empleado para predecir un amplio rango de problemas, desde comportamientos del ser humano [1], [2], pasando por economía [3], [4], análisis de información médica [5], diagnóstico de cáncer [6], reconstrucción de señales cerebrales [7], sistemas de comunicación [8],[9], epidemiología [10], entre otros. El caso más sencillo del análisis de regresión es el proceso mediante el cual se utilizan una o más variables para predecir otra [11], sin embargo puede resolver problemas tan complejos como la predicción anticipada de un ataque cardiaco, el cual es un caso no lineal, con ruido inherente y altos niveles de incertidumbre [12].

De las dificultades propias del problema a resolver, depende la complejidad del modelo de regresión, por ejemplo, para el caso de una variable independiente y una variable dependiente (a predecir), se puede utilizar regresión simple, mientras para hallar una solución satisfactoria en el caso de predicción anticipada de un ataque cardiaco [12] es necesario tener en cuenta modelos que consideren la incertidumbre, que sean robustos ante el ruido y que puedan manejar no linealidades. Para esto, existen gran cantidad de métodos para resolver problemas de regresión, entre los cuales destacan los métodos estadísticos [13], [14], estocásticos [15], no lineales [16], entre otros. Estos métodos destacan gracias a sus ventajas como buen desempeño ante problemas con no linealidades, robustez ante el ruido y estimación óptima [17]. De la misma manera, poseen desventajas como lo son la necesidad de tener amplios conocimientos estadísticos para poder ser implementados, dificultad para escoger el método de regresión óptimo para el problema y alta complejidad al tratar problemas de alta no linealidad [17]

Debido a las desventajas que presentan los métodos clásicos de regresión y gracias al desarrollo tecnológico de los últimos años, se han desarrollado sistemas de Inteligencia Artificial para la resolución de problemas de regresión, en donde destacan las Redes Neuronales [17], las cuales han tenido gran aceptación debido a que no requieren una formación estadística avanzada, tienen habilidad para detectar relaciones complejas no lineales entre variables, los datos pueden estar contaminados de ruido e incertidumbre y tienen la capacidad de adaptar la red y los métodos de aprendizaje de esta para cada tipo de problema [18], [19]. Las Redes Neuronales han ayudado a resolver gran diversidad de problemas en el área de regresión obteniendo resultados óptimos en múltiples áreas del conocimiento [20]–[25], entre estas implementaciones destacan como métodos de entrenamiento, métodos clásicos en la Inteligencia Artificial como lo son Back Propagation [23]–[25], métodos de descenso de gradiente [26] o regulación Bayesiana. A pesar de las ventajas que

presentan las redes neuronales, también poseen desventajas, entre las principales se encuentran su naturaleza de "caja negra", gran esfuerzo computacional y tendencia a sobre aprendizaje.

Para luchar con las desventajas propias de las redes neuronales, dentro de la Inteligencia Artificial se ha desarrollado una técnica llamada Boosting, cuyo objetivo es: con base en aprendices débiles[1] crear un aprendiz fuerte[2] [27]. Originalmente los métodos de Boosting fueron planteados como algoritmos de clasificación binaria [28] y destaca principalmente el algoritmo AdaBoost [29] debido a que ha demostrado tener mejor desempeño que otros métodos de clasificación clásicos [30]–[32]. Gracias a los resultados obtenidos en clasificación, se extendió el algoritmo para problemas de regresión, AdaBoost.R, en donde se reduce el problema de regresión a un problema de clasificación, proyectando el conjunto de datos de regresión en un conjunto de datos de clasificación, y luego se aplica AdaBoost.

Sin embargo, aunque AdaBoost.R puede ser efectivo en la proyección del conjunto de datos, sufre dos inconvenientes. Primero, expande cada ejemplo en la muestra de regresión en múltiples ejemplos de clasificación y luego aumenta linealmente el número de iteraciones de Boosting y segundo, cambia la función de error de iteración a iteración e inclusive difiere en muestras de la misma [33]. Para cambiar la manera de reducir el problema de regresión a clasificación binaria, en AdaBoost.RT se propone una constante φ como un valor límite de error relativo usado para demarcar la predicción como correcta o incorrecta, dado que la tasa de error es calculada contando el número de predicciones correctas o incorrectas [34]. Dado que el algoritmo es relativamente nuevo, se han presentado pocas aplicaciones de este [35]–[37], las cuales han tenido buen desempeño, aunque presentan inconvenientes para determinar el valor óptimo de φ, puesto que se ha mostrado que el algoritmo es sensible a φ [38]. Debido a que no es fácil calibrar el límite φ, se propuso un algoritmo para resolver el problema de AdaBoost.RT. En este método se usa un valor de φ variable, el cual es auto-adaptativo de acuerdo a la tasa de error durante todo el proceso [38].

En la revisión bibliográfica, además de [38] no se han encontrado más aplicaciones de AdaBoost.RT con φ variable, de la misma manera, tampoco se ha encontrado aplicación de AdaBoost en problemas de control y es muy limitada las aplicaciones encontradas de predicción de series económicas [39], [40]. De acá surge la pregunta que se busca resolver con este proyecto, ¿Es posible extender el concepto de AdaBoost.RT con φ auto-adaptativo a un problema de regresión de una serie económica y un problema clásico de control?

1.2 JUSTIFICACIÓN

El análisis de regresión ha sido ampliamente desarrollado y ha sido un tema de investigación con una amplia gama de áreas de aplicación, principalmente aquellos problemas que presentan no linealidades, ya que los métodos clásicos de regresión, como lo son los estadísticos o estocásticos, no tienen en cuenta la contaminación de los datos por ruido o incertidumbre y desarrollar el modelo matemático o validarlo presenta un nivel de complejidad alto. Esto se puede evitar utilizando Redes Neuronales las cuales representan el problema a partir de los datos del mismo, además han sido ampliamente utilizadas, bien sea independientemente o como una herramienta auxiliar [37].

[1] Algoritmo de aprendizaje débil, con un desempeño ligeramente mejor que el azar [27].
[2] Algoritmo de aprendizaje fuerte, con una alta probabilidad de tener una hipótesis que es correcta en todos menos en una pequeña fracción arbitraria de los casos [27].

Sin embargo, en algunas aplicaciones de Redes Neuronales se presentan inconvenientes inherentes como convergencia local y velocidad de entrenamiento del algoritmo lenta, las cuales restringen su desarrollo en la práctica [41], para evitar estas dificultades se escogió el uso de AdaBoost, un método de Boosting ampliamente usado en problemas de clasificación con muy buen desempeño [30]–[32] y en los últimos años se ha extendido a problemas de regresión destacando por sus buenos resultados [35]–[37]. De igual manera, el uso de este algoritmo en problemas de control no se encontró en la revisión bibliográfica realizada, Por estos motivos es de gran interés hacer uso del algoritmo de AdaBoost para predecir una serie de tiempo y además resolver un problema de control.

Para el caso de la serie de tiempo, se escogió una serie de tiempo económica dado que en el algoritmo AdaBoost este tipo de series ha sido poco investigado, pero los casos encontrados han sido comparados con redes neuronales y el algoritmo AdaBoost ha tenido mejor desempeño [39], [40].

En [39], se hace la predicción de quiebra corporativa de una empresa, donde se comparan tres variaciones del algoritmo AdaBoost. Cabe resaltar que en este caso, aunque es una predicción dado que se realiza a futuro, se puede considerar como un problema de clasificación dado que la variable de salida es binaria. En [40] se hace uso de un algoritmo de AdaBoost de clasificación múltiple para predecir las dificultades financieras que se pueden presentar en una empresa y se compara con una red neuronal. Para su validación se hace uso de una base de datos de 30 empresas chinas, en la cual el algoritmo basado en AdaBoost en la mayoría tiene un error menor que la red neuronal.

Otro motivo para escoger una serie de tiempo económica es que es una serie de tiempo difícil de predecir debido a la influencia de incertidumbre intrínseca, caos y ruido, además de que la distribución de las secuencia de tiempo varía con el tiempo [35]. Se buscó una serie económica que tuviera relevancia a nivel local, por tanto se escogió un índice bursátil de la Bolsa de Valores Colombiana que refleja las variaciones de las 20 acciones más líquidas de Colombia [42]. Además de esto, esta serie de tiempo ha sido estudiada anteriormente [43], [44], en donde destaca la predicción del índice por medio de una red neuronal y se compara con un método económico de predicción [44]. En este se concluye que aunque los dos modelos se ajustan de manera apropiada al índice, el sistema basado en redes neuronales es levemente mejor. Debido al estudio previo de la aplicación de redes neuronales en esta serie, surge el interés en aplicar el algoritmo de AdaBoost.RT en este índice económico.

Para el caso del problema de control se escoge el péndulo invertido debido a que es altamente no lineal y en lazo abierto es un sistema inestable [45], las no linealidades del sistema tienen repercusiones en la escogencia del tipo de controlador, por ejemplo, los PID lineales estándar no pueden ser usados para este sistema dado que no pueden manejar las complejidades del péndulo, como se puede observar en [46] donde se hace uso de una red neuronal y un controlador PID para resolver el problema, en este se puede observar que para controlar el ángulo del péndulo basta con el controlador PID, pero el control de posición del carro falla debido a las características no lineales del sistema, lo que se superó por medio de la inclusión de una red neuronal. Además en [47] se hace uso de diferentes tipos de controladores basados en redes neuronales para resolver el sistema de control, en este se concluye que aunque las técnicas de control basados en redes supervisadas y no supervisadas son efectivas, aquellas supervisadas son más eficientes. Con base en esto se puede precisar que los controladores básicos como el PID no cumplen los requerimientos mínimos del péndulo invertido, mientras aquellos basados en sistemas inteligentes supervisados satisfacen todos los requerimientos del sistema. Dado que AdaBoost es una técnica de Inteligencia Artificial y gracias a su simplicidad, puesto que es un aprendiz fuerte conformado por un conjunto de aprendices débiles, surge el interés en hacer uso del algoritmo AdaBoost para diseñar un controlador del péndulo invertido.

1.3 OBJETIVOS

1.3.1 Objetivo General

Desarrollar un procedimiento basado en el algoritmo AdaBoost.RT con φ auto-adaptativo para la predicción de una serie económica y el diseño de un controlador no lineal.

1.3.2 Objetivos Específicos

- Desarrollar y programar el algoritmo AdaBoost.RT con φ auto-adaptativo en software.
- Predecir y comparar el valor futuro de la serie COLCAP usando una arquitectura neuronal y AdaBoost.RT con φ auto-adaptativo.
- Modelar e implementar un controlador usando AdaBoost.RT con φ auto-adaptativo para el problema de péndulo invertido.
- Evaluar el desempeño del algoritmo AdaBoost.RT con φ auto-adaptativo en un problema de regresión de series económicas y un problema de control

1.4 ALCANCES Y LIMITACIONES

A lo largo del desarrollo del proyecto, sólo se codificará, programará y evaluará el desempeño del algoritmo AdaBoost.RT con φ auto-adaptativo. Las demás variantes de este algoritmo no se tendrán en cuenta, dado que o son métodos de clasificación binaria o multi-clase o aquellos que tienen uso en regresión no son de interés en el proyecto.

La base de datos a usar para la predicción de la serie COLCAP será tomada de la página de la Bolsa de Valores de Colombia [48], donde se cuenta con datos de cada día bursátil desde el 24 de Diciembre del 2007 hasta el 24 de Octubre del 2016. En la revisión bibliográfica, se observó que los indicadores de análisis técnico ayudan a obtener resultados satisfactorios en la predicción de índices económicos y principalmente en el caso del índice COLCAP [43], por este motivo para la predicción del índice sólo se hará uso de la serie de tiempo y índices técnicos calculados a partir de la misma.

Para el péndulo invertido, se modelará la planta de acuerdo a la dinámica no lineal del sistema que se muestra en (19) y (20). Con base en estas ecuaciones, se simulará el modelo del péndulo invertido y a partir del modelo simulado se diseñarán los controladores y se realizarán las pruebas.

Dado que para poder medir el desempeño del algoritmo es necesario compararlo con resultados existentes de los mismos problemas, para el caso de la serie económica se diseñará un sistema predictor con base en redes neuronales, de la misma manera para el problema de control, se desarrollará un controlador con base en redes neuronales. Se escogió redes neuronales como sistema base para las comparaciones debido a las semejanzas en los métodos y a que ambos hacen parte de la Inteligencia Artificial.

Además es necesario mencionar que el proyecto será llevado a cabo únicamente en simulación en el software MATLAB®, donde se realizarán simulaciones de tipo numérico y funcional. Además de esto el algoritmo será evaluado únicamente en las dos aplicaciones escogidas, índice COLCAP en el caso de la regresión y péndulo invertido en el problema de control. Los resultados obtenidos con base en las aplicaciones realizadas no serán generalizados para cualquier clase de problema de regresión o control dado que cada caso tiene características específicas.

2 MARCO TEÓRICO

En este capítulo se presentan los conceptos y fundamentos teóricos usados en este proyecto. Se encuentra dividido en cinco secciones. Se presentan los conceptos básicos y el funcionamiento de las Redes Neuronales y el algoritmo AdaBoost, además se explican las características principales de las series de tiempo económicas y del péndulo invertido. Adicionalmente, se presentan antecedentes de problemas de predicción de series económicas y de control del péndulo invertido.

2.1 REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales son un método de la Inteligencia Artificial que está usualmente definida como una red compuesta de un gran número de procesadores simples (neuronas) que están masivamente interconectadas, que operan en paralelo y aprenden de la experiencia (ejemplos), estas son las características primarias conocidas de un sistema neuronal biológico que son fácilmente aprovechables en una red neuronal artificial [49]. El modelo más simple de una ANN es una red de una neurona, por este motivo se presenta en la Figura 2.1 sus componentes principales.

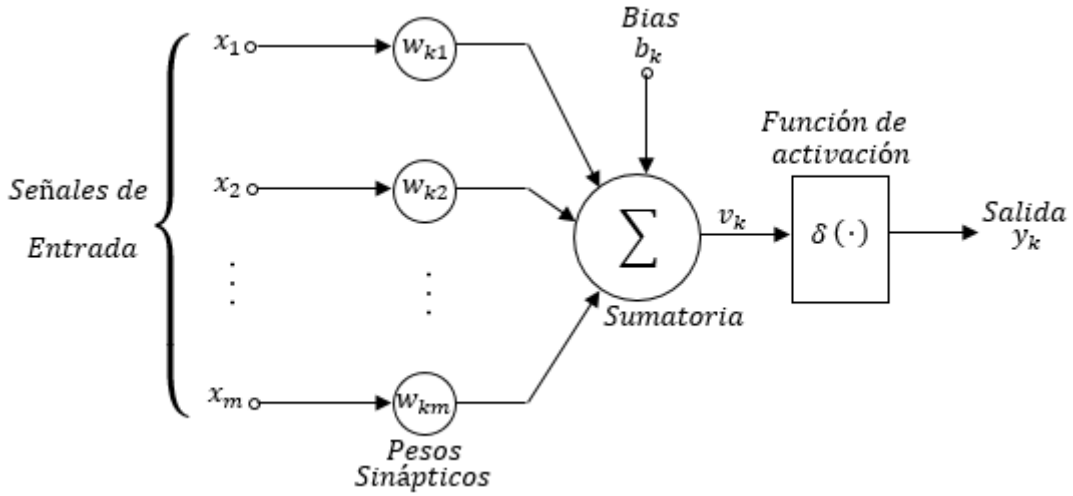


Figura 2.1. Diagrama de una neurona

Donde m es la cantidad de muestras, k el indicador de la neurona, $x_1, x_2, \ldots, x_m$ las señales entradas de la neurona, $w_{k1}, w_{k2}, \ldots, w_{km}$ el peso sináptico de cada entrada de la neurona k que puede ser una entrada del sistema o estar interconectada a otra neurona, $b_k$es un bias aplicado externamente cuya función es incrementar o disminuir la entrada a la red de la función de activación, una sumatoria encargada de sumar las entradas de la neurona ponderadas por su respectivo peso, v_k denominado como el estado interno de la neurona que es el resultado de la sumatoria, y_k que es la salida de la neurona, descrita por (1) y $\delta(\bullet)$ es la función de activación que se encarga de limitar la amplitud del rango de salida a un valor finito, normalmente normalizado en el rango de $[\,0\,,1\,]$ o $[\,-1\,,1\,]$ [50]. Existen diversos tipos de funciones de activación, todas ellas no lineales, de las cuales la más usada para modelos de una capa es el limitador duro y para modelos multi-capa se suelen usar funciones continuas como Gaussiana o Sigmoidea. En términos matemáticos se puede describir la neurona k por medio de (1) [50].

$$y_k = \delta\left(\sum_{i=1}^{m}(x_i * w_i) + b_k\right) \quad (1)$$

2.1.1 Arquitectura de la red

Una red neuronal está conformada por múltiples neuronas como las mostradas en la Figura 2.1, de tal manera que se estructuran en capas y entre más neuronas interconectadas, el modelo se vuelve más complejo. La manera en la cual se encuentran estructuradas las neuronas en una red neuronal está íntimamente relacionada con el algoritmo de aprendizaje usado para entrenar la red. De manera general se puede dividir en 3 diferentes clases de arquitecturas de red.

2.1.1.1 Redes Feed-forward de capa simple

En una red neuronal "estratificada" las neuronas están organizadas en forma de capa. En su forma más simple, se tiene una capa de entrada de nodos de origen que se proyecta sobre una capa de neuronas de salida, más no en viceversa, es decir, es una red estrictamente conectada hacia delante. En esta arquitectura de red neuronal no se cuenta la capa de entrada de nodos de origen, dado que no se realizan operaciones en esta capa [26].

2.1.1.2 Redes Feedforward Multicapa

La segunda clase de una red neuronal conectada estrictamente hacia delante se distingue por la presencia de una o más capas ocultas, cuyos nodos de cálculos son llamados neuronas ocultas o unidades ocultas, cuya función es intervenir entre las entradas externas y las salidas de la red de manera útil. El añadir una o más capas ocultas permite a la red extraer estadísticas de mayor orden. Los nodos de origen en la capa de entrada de la red suplen los elementos respectivos a las neuronas de la segunda capa (la primera capa oculta). Las señales de salida de la segunda capa son usadas como entrada de la tercera capa y así sucesivamente. Típicamente las neuronas de cada capa de la red neuronal tienen únicamente como entrada las señales de salida de la capa predecesora [51].

Se considera que una red neuronal está *completamente conectada* cuando cada nodo de cada capa de la red está conectada a cada nodo de la capa siguiente. Sin embargo, si alguna de las conexiones sinápticas de la red falta, se dice que la red está *parcialmente conectada* [50].

2.1.1.3 Redes Recurrentes

Las redes neuronales recurrentes difieren de las redes feedforward porque tienen por lo menos un lazo de realimentación. Por ejemplo, una red recurrente puede consistir de una capa simple de neuronas con una neurona, realimentando hacia atrás su señal de salida a una de las entradas de una o varias neuronas. La presencia de lazos de realimentación tiene un gran impacto en la capacidad de aprendizaje de la red y en su desempeño, puesto que implican un comportamiento dinámico no lineal, asumiendo que la red neuronal contiene unidades no lineales [52].

2.1.2 Métodos de entrenamiento

En el proceso de entrenamiento o aprendizaje, el ambiente le suple información al elemento de aprendizaje que usa esta información para hacer mejoras en su *base de conocimiento*. La información provista usualmente es imperfecta, de tal manera que el elemento de aprendizaje no sabe a priori

como ignorar los detalles que no son importantes. La red neuronal por tanto opera adivinando, y luego recibe una retroalimentación de la red, el mecanismo de retroalimentación permite a la red evaluar su hipótesis y revisarla de ser necesario [26].

2.1.2.1 Aprendizaje Supervisado

En este tipo de aprendizaje se presenta a la red neuronal una base de datos de patrones de entrada junto con la salida esperada (conjunto de entrenamiento) y se calcula la respuesta de la red neuronal, la diferencia entre la salida esperada y la de la red neuronal es considerada error. Los parámetros de la red son ajustados bajo la influencia tanto del conjunto de entrenamiento como de la señal de error, este ajuste es llevado a cabo iterativamente con el objetivo de que la red neuronal se asemeje al conjunto de entrenamiento, cuando este objetivo se alcanza se considera que la red neuronal está entrenada [53].

2.1.2.2 Aprendizaje no supervisado

En este aprendizaje se presenta a la red neuronal una base de datos de patrones de entrada (conjunto de entrenamiento) pero no hay información disponible sobre la salida esperada. Los parámetros de la red neuronal se ajustan con base a la correlación existente entre los datos de entrenamiento. Una vez que la red se ha sincronizado a las regularidades estadísticas del conjunto de entrenamiento, desarrolla la habilidad de formar representaciones internas y por tanto crear nuevas clases automáticamente [54].

2.1.2.3 Aprendizaje por refuerzo

Este tipo de aprendizaje se ubica en medio de los dos anteriores, se presenta a la red neuronal una base de datos de patrones de entrada (conjunto de entrenamiento) y se le indica a la red si la respuesta obtenida es correcta o incorrecta, más no se le proporciona el valor de la salida esperada. Este tipo de aprendizaje es muy útil en casos en que se desconoce cuál es la salida exacta que debe proporcionar la red [55].

Las redes neuronales cuentan con la capacidad de partir de una base de datos de entrada-salida y crear un modelo capaz de resolver una gran diversidad de problemas a partir de la experiencia. Esta versatilidad de las ANN, las hace una gran opción para problemas de regresión, dado que definir el tipo de regresión a utilizar puede ser una tarea difícil, pero en la mayoría de los casos se cuenta con una base de datos de la cual se puede aprender y desarrollar un modelo óptimo [56].

2.2 ADABOOST

AdaBoost es un algoritmo de Inteligencia Artificial que es una variación de un método denominado Boosting, cuyo objetivo es hacer un conjunto de aprendices débiles para generar un proceso de aprendizaje fuerte para resolver un problema de clasificación binaria[3] [57]. Boosting funciona corriendo repetidamente un algoritmo dado de aprendizaje débil en diferentes distribuciones de la base de datos de entrenamiento y finalmente combinar sus salidas. En cada iteración la

[3] Clasificación binaria se refiere a la tarea de clasificar los elementos de un conjunto de datos dado en dos grupos en base a una regla de clasificación.

distribución de la base de datos de entrenamiento depende del desempeño del algoritmo en la iteración previa.

2.2.1 Aprendiz débil

Se considera a un aprendiz débil como un método de aprendizaje que está ligeramente correlacionado con la clasificación o predicción verdadera, es decir, tiene una tasa de error ligeramente mayor que una suposición aleatoria [28]. Normalmente pero no necesariamente, además del desempeño ligeramente mayor al azar se considera la sencillez computacional del método de aprendizaje [29]. En general, se considera un modelo en el cuál a partir de un conjunto de datos de entrada, se aplica una regla de decisión la cual determina la salida del modelo. La regla de decisión se determina a través del proceso de aprendizaje.

Aprendiz débil

Dada una base de datos de entrenamiento $(x_1, y_1), \dots, (x_m, y_m)$; donde m es la cantidad de muestras, $x_i \in \mathbf{X}$ es la muestra a clasificar, $y_i \in \{-1, +1\}$ es la clasificación de la muestra i.

Definir el algoritmo de entrenamiento del aprendiz débil.

Definir it_{max}, que corresponde al parámetro de parada del algoritmo

Para $i_t < it_{max}$

1 Calcular la salida actual

$$out = c_{it}\{x_m, y_m\},$$

donde $c_{it}(x_i, y_i) = \begin{cases} 0 \\ 1 \end{cases}$ y depende del tipo de algoritmo aprendiz débil escogido.

2 Actualizar la regla de decisión según el algoritmo de entrenamiento del aprendiz débil.

3 Calcular el error del modelo

Figura 2.2 Aprendiz débil

2.2.1.1 Perceptrón

El Perceptrón es un algoritmo de aprendizaje supervisado. Se basa en el aprendizaje de una neurona no lineal, como la presentada en la Figura 2.1. El algoritmo se encarga de ajustar el bias y los pesos de conexión de la neurona y debido a sus características suele ser usado para resolver satisfactoriamente problemas linealmente separables [37]. Para el algoritmo de aprendizaje se necesita una base de datos de entrenamiento, los cuales son la entrada del algoritmo. Durante el entrenamiento se modifican los pesos de entrada y el bias de la neurona, dependiendo de la tasa de aprendizaje, que debe ser un número positivo pequeño. A continuación se presenta el algoritmo de aprendizaje del perceptrón de manera general.

Algoritmo de aprendizaje del perceptrón

Dada una base de datos de entrenamiento $(x_1, y_1), \dots, (x_m, y_m)$; donde m es la cantidad de muestras, $x_i \in \mathbf{X}$ es la muestra a clasificar, $y_i \in \{-1, +1\}$ es la clasificación de la muestra i.

Inicializar los pesos w

Definir $\mathrm{it}_{\max}$, que corresponde al parámetro de parada del algoritmo

Para $\mathrm{it} < \mathrm{it}_{\max}$

Calcular la salida actual

$$\mathrm{out} = \sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{x_i} * \mathrm{w_i}(\mathrm{it})$$

Calcular el error del modelo

$$\mathrm{error} = \frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} |\mathrm{y_i} - \mathrm{out_i}(\mathrm{it})|$$

Si error = 0, se da fin al algoritmo, si no, se continúa con el algoritmo

Actualizar los pesos

$$\mathrm{w}_i(\mathrm{it}+1) = \mathrm{w_i}(\mathrm{it}) + \big(\mathrm{y}_i - \mathrm{out_i}(\mathrm{it})\big)\mathrm{x_i}$$

Figura 2.3 Algoritmo de aprendizaje del perceptrón

- Ejemplo

La manera más fácil de entender el algoritmo de aprendizaje del perceptrón es por medio de un ejemplo A continuación se presenta el caso más sencillo de una clasificación binaria que es la compuerta AND.

En la Figura 2.4a se presenta la base de datos, donde se observan cuatro puntos pertenecientes a dos clases, tres de los puntos pertenecientes a la clase "roja" y el otro perteneciente a la clase "azul" El objetivo es encontrar una recta que separe correctamente los datos según la clase a la que corresponden.

Como parte de la inicialización antes de comenzar el algoritmo es necesario definir el número de iteraciones máximas it_{max} e inicializar los pesos w de manera aleatoria, luego se comienza con el proceso iterativo.

- $it = 1$
  Según el paso 1, se calcula la salida actual del modelo con la inicialización aleatoria de los pesos, la cual se muestra en la Figura 2.4b, en donde la zona azul se consideran los datos clasificados en la clase azul y los datos pertenecientes a la zona rosa se consideran de la clase roja. De acuerdo al paso 2, se calcula el error del modelo, a partir de la figura se observa que dos datos están mal clasificados, por tanto es necesario continuar al paso 3 y actualizar los pesos del modelo, dando paso a la iteración 2.
- $it = 2$
  Se calcula la actual salida del modelo con la actualización de pesos la cual se presenta en la Figura 2.4c, en donde la zona azul se considera los datos clasificados en la clase azul y los pertenecientes a la zona rosa se consideran de la clase roja. En seguida se lleva a cabo el paso 2, que corresponde al cálculo del error, a partir de la figura se puede observar que hay un dato rojo dentro de la zona azul motivo por el cual se continúa con la actualización de los pesos del modelo (paso 3) y se sigue con la iteración 3.
- $it = 3$

A partir de la actualización de pesos llevada en el paso 3 de $it = 2$, se calcula la salida actual presentada en la Figura 2.4d. A continuación se lleva a cabo el paso 2 del algoritmo de aprendizaje, en donde se calcula el error del modelo, a partir de la figura se observa que los datos se encuentran correctamente clasificados, por lo cual el error es igual a cero y se finaliza el aprendizaje del perceptrón.

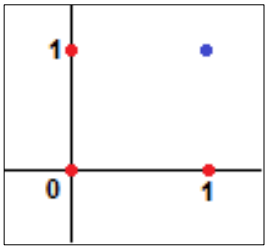


Figura 2.4a.. Problema de clasificación

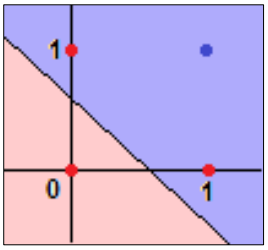


Figura 2.3b Perceptrón iteración 1

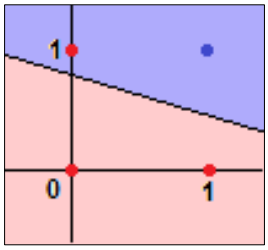


Figura 2.3c Perceptrón iteración 2

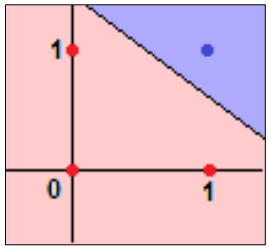


Figura 2.3d Perceptrón iteración 3

2.2.1.2 Árbol de decisión

Es un método de aprendizaje supervisado cuyo propósito es crear un modelo que prediga el valor de una variable objetivo a través de aprender reglas simples de decisión inferidas de las características de los datos. Se divide una base de datos en pequeños subconjuntos mientras al mismo tiempo se va creando un árbol de decisión asociado. El resultado final es un árbol con nodos de decisión y nodos de hoja. Un nodo de decisión tiene 2 o más ramas, mientras un nodo hoja representa una clasificación o una decisión, al ser usado como algoritmo débil, el árbol débil debe tener un solo nodo de decisión [58].

2.2.1.3 Support Vector Machine

Una máquina de vectores de soporte (SVM) es un clasificador discriminador formalmente definido por un hiperplano de separación. Dada una base de datos de entrenamiento, el algoritmo genera un hiperplano óptimo el cual categoriza los nuevos ejemplos. La operación del algoritmo SVM está basado en encontrar el hiperplano que da la mayor mínima distancia (margen) entre las muestras de entrenamiento, es decir el hiperplano de separación óptimo maximiza el margen de las muestras de entrenamiento [31]

2.2.2 Algoritmo AdaBoost

Originalmente el algoritmo de AdaBoost fue desarrollado para resolver problemas de clasificación binaria, luego el algoritmo fue extendido a clasificación multi-clase [28]. AdaBoost ha sido ampliamente usado para resolver problemas de clasificación y ha demostrado tener buen desempeño en este tipo de problemas [39],[40],[59]. Su mayor avance se basa en incluir una distribución de la base de datos de entrenamiento y existen diversos métodos para llevar esto a cabo.

La manera en la que se actualiza la distribución de la base de datos de entrenamiento depende del método de Boosting que se esté usando, el método original se basa en Boosting por filtrado, pero necesita una gran cantidad de datos de entrenamiento, lo cual en muchos casos no es posible. El algoritmo AdaBoost fue diseñado para superar este inconveniente y existen múltiples versiones de este, aunque se pueden diferenciar de acuerdo a la manera en que se redistribuyen los datos de

entrenamiento, AdaBoost por submuestreo[4] y AdaBoost por ponderación[5][29]. En la Figura 2.5 se presenta el algoritmo de AdaBoost por ponderación, en el cual se basa el que será usado en este proyecto.

Algoritmo de AdaBoost por ponderación

Dada una base de datos de entrenamiento $(x_1, y_1), \dots, (x_m, y_m)$; donde m es la cantidad de muestras, $x_i \in \boldsymbol{X}$ es el objeto o muestra a clasificar, $y_i \in \{-1, +1\}$ es la clasificación.

Definir T que corresponde al número de clasificadores débiles a usar.

Inicializar la función de distribución de probabilidad $D_1(i) = \frac{1}{m}$ para todo $i = 1, \dots, m$.

Para $t = 1, \dots, T$

1. Entrenar un clasificador débil teniendo en cuenta la distribución D_t
2. Seleccionar una hipótesis débil $h_t : \boldsymbol{X} \rightarrow \{-1, +1\}$ con el menor error ε_t

$$\varepsilon_t = \sum_{i=1}^{m} D_t(i) * P_t(i),$$

$$donde\ P_t(i) = \begin{cases} 1, & y_i \neq h_t(i) \\ 0, & y_i = h_t(i) \end{cases}$$

3. Calcular el coeficiente α_t, que corresponde al peso del aprendiz débil de la iteración t

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

4. Actualizar D_t, para $i = 1, \dots, m$:

$$D_{t+1}(i) = \frac{D_t(i) * \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$donde\ Z_t$ es un factor de normalización escogido para que D_{t+1} sea una función de distribución de probabilidad.[6]

Calcular la salida del clasificador fuerte $H_{final}(x)$

$$H_{final}(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Figura 2.5 Algoritmo AdaBoost por ponderación

[4] Se submuestrea el tamaño de la base de datos de entrenamiento y los ejemplos de entrenamiento se utilizan y se remuestrean según una distribución de probabilidad dada durante el entrenamiento [29].

[5] Se ponderan todos los datos de entrenamiento, los cuales son usados para entrenar el método de aprendizaje débil con pesos asignados para cada muestra. Sólo es útil cuando el aprendiz débil puede manejar muestras ponderadas[29].

[6] De tal manera que $D_t(i)$ cumpla con las propiedades de la función de distribución de probabilidad, tal que $\sum D_{t+1} = 1$

- **Ejemplo AdaBoost por ponderación**

En la Figura 2.6, se presenta un ejemplo del procedimiento de AdaBoost por ponderación. Se tiene el conjunto de datos presentado en la Figura 2.6a donde los puntos representan la base de datos de entrada x_m siendo $m = 10$ y el color de cada punto representa la clase de cada uno de ellos y_i, para $i = 1, .., m$.

Antes de iniciar el procedimiento es necesario definir la cantidad de clasificadores débiles (T) que se van a usar e inicializar la función de distribución de probabilidad de los pesos de las muestras $D_1(i)$, en la cual todas las muestras tienen el mismo peso, es decir, la misma importancia. Luego de tener definidos estos parámetros se comienza el proceso iterativo del algoritmo, que consta de 4 pasos.

- Paso 1
  Se "llama" al aprendiz débil y se le proporcionan los datos de entrenamiento y la distribución de pesos $D_i(t)$. El aprendiz débil retorna un conjunto de clasificadores dependiendo del número de iteraciones del algoritmo de aprendizaje del aprendiz débil (it_{max}).
- Paso 2
  Seleccionar el "mejor clasificador" de la iteración t (h_t), que es aquel que presenta el menor error ε_t
- Paso 3
  Calcular el peso del "mejor clasificador" de la iteración t (h_t).
- Paso 4
  Actualizar la función de distribución de probabilidad de los pesos de las muestras que será usada en la iteración $t + 1$.
- $t = 1$

El "mejor clasificador" débil h_1 realiza una división tratando de separar los datos, la cual se presenta en la Figura 2.6.b, donde se observa que comete tres errores al dejar tres datos azules dentro de la zona roja, los cuales tendrán un mayor peso en el la distribución de pesos de las muestras $D_2(i)$ utilizada en el entrenamiento del clasificador h_2. El peso de los datos en donde se presentó error en $t = 1$ se incrementa, como se mencionó anteriormente, de tal manera que el nuevo clasificador h_2 le dé más importancia a resolver satisfactoriamente los puntos erróneos de la iteración anterior.

- $t = 2$

El "mejor clasificador" débil de la segunda iteración h_2 logra discriminar correctamente los datos negativos que tenían mayor peso pero clasifica erróneamente tres puntos rojos que tenían un bajo peso. Por este motivo, en la distribución de pesos de las muestras de la tercera iteración $D_3(i)$ se incrementa el peso correspondiente a las muestras rojas clasificadas erróneamente y aquellos datos que ya fueron correctamente clasificados reciben un menor peso.

- $t = 3$

Se realiza un tercer clasificador h_3 que de igual manera tratará de encontrar el mejor clasificador, de acuerdo a la distribución de pesos de las muestras $D_3(i)$.

Al evaluar el conjunto de clasificadores débiles de la iteración $tres$, correspondiente a H_{final}, este soluciona de manera satisfactoria el problema propuesto, por este motivo no es necesario incluir más clasificadores débiles, puesto que aumenta la complejidad del sistema más no mejora su desempeño. El modelo conjunto final se presenta en la Figura 2.7.

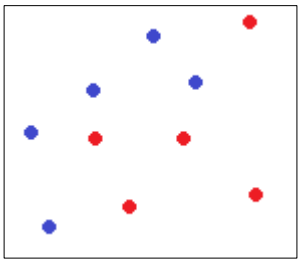


Figura 2.6a. Problema de clasificación

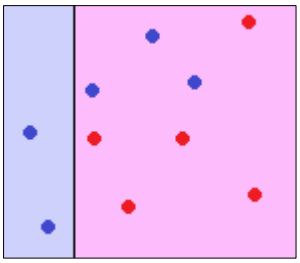


Figura 2.6b. Clasificador h_1

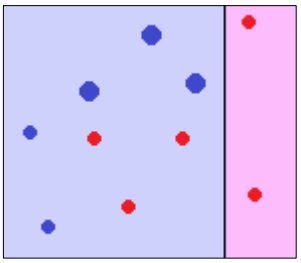


Figura 2.6c. Clasificador h_2

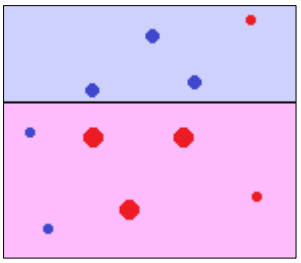


Figura 2.6d. Clasificador h_3

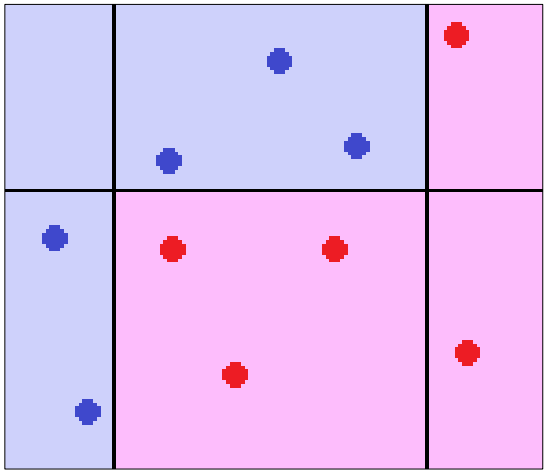


Figura 2.7. Clasificador Final AdaBoost

Debido a la gran acogida de AdaBoost en problemas de clasificación, el algoritmo fue extendido para problemas de regresión con el algoritmo AdaBoost.R, en donde se reduce el problema de regresión a un problema de clasificación binaria, sin embargo esta extensión presentaba dos inconvenientes, En primer lugar, expande cada ejemplo en la muestra de regresión en múltiples ejemplos de clasificación y luego aumenta linealmente el número de iteraciones de Boosting y segundo, cambia la función de error de iteración a iteración e inclusive esta función difiere en muestras de la misma iteración[28]. Luego de esto se han propuesto diversos métodos para reducir el problema de regresión a clasificación, en donde resalta el algoritmo AdaBoost.RT dado que obtiene mejor desempeño y es más fácil de implementar que AdaBoost.R [57].

2.2.3 Algoritmo AdaBoost.RT

Algoritmo de AdaBoost.RT

1. Definir las entradas:
   Secuencia de m ejemplos $(x_1, y_1), (x_2, y_2), \ldots , (x_m, y_m)$, donde $y \in \mathbf{R}$
   Algoritmo de aprendizaje débil
   Entero T, el cual especifica el número de iteraciones del algoritmo
   Límite $\varphi (0 < \varphi < 1)$ para demarcar las muestras correctas de las muestras incorrectas
2. Inicializar:
   $t = 1$ Número de iteración del algoritmo
   Función de Distribución de Probabilidad $D_t(i) = 1/m$ para todo i
   Tasa de error $\varepsilon_t = 0$
3. Iterar mientras $t \leq T$;

Llamar al aprendiz débil, proveyéndole la distribución D_t

Construir el modelo de regresión: $f_t(x) \rightarrow y$.

Calcular el error relativo absoluto para cada muestra de entrenamiento

$$ARE(i) = \frac{f_t(x_i) - y_i}{y_i} \tag{2}$$

Calcular la tasa de error de $f_t(x)$:

$$\varepsilon_t = \sum_{i:ARE_t(i)>\varphi} D_t(i) \tag{3}$$

Establecer $\beta_t = \varepsilon_t^n$, donde $n = 1,2$ o 3 (lineal, cuadrado o cúbico)

Actualizar la distribución D_t

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \text{ x } \begin{cases} \beta_t, & \text{si } ARE_t(i) \leq \varphi \\ 1, & \text{de otra manera} \end{cases} \tag{4}$$

Donde Z_t es un factor de normalización escogido de tal manera que D_{t+1} sea una distribución.

Establecer $t = t + 1$

4. Calcular la hipótesis final

$$f_{fin}(x) = \frac{\sum_{t=1}^{T} \left\{ \left(\log \frac{1}{\beta_t} \right) . f_t(x) \right\}}{\sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right)} \tag{5}$$

Figura 2.8 Algoritmo AdaBoost.RT

- **Ejemplo AdaBoost.RT**

En la Figura 2.9 se presenta un ejemplo del funcionamiento del algoritmo AdaBoost.RT en el cual se busca encontrar una función que aproxime los datos presentados la Figura 2.9a, la base de datos, en donde los datos de entrada se presentan en azul y los datos de validación se presentan en rojo. La función de aproximación es la dada por la combinación lineal resultado del modelo conjunto de AdaBoost $f_{fin}(x)$.

Antes de comenzar con el algoritmo, es necesario definir el algoritmo de aprendizaje, en este caso se hace uso del perceptrón, la cantidad de iteraciones del algoritmo (T) y la función de distribución de probabilidad $D_1(i)$ como se presenta en el segundo paso del algoritmo. Además es necesario definir el valor del discriminador de pruebas correctas e incorrectas φ.

- $t = 1$

Se "llama" al perceptrón, el cual lleva a cabo su algoritmo de aprendizaje como se presentó en la Figura 2.3 y da como resultado el modelo de regresión f_1 que se presenta en la Figura 2.9b como los puntos verdes. Se calcula el error relativo absoluto como se presenta en (2) y a continuación se calcula la tasa de error de f_1 según (3). Finalmente se actualiza la función de distribución de probabilidad de

pesos de las muestras ($D_2(i)$) que será usada para el entrenamiento del aprendiz débil de la iteración 2.

- $it = 2$

En la segunda iteración se aumenta el peso de las muestras "erróneas" de la iteración anterior, lo cual se refleja en la distribución de pesos $D_2(i)$ y se puede observar en la Figura 2.9c las muestras erróneas se presentan como los círculos rojos rellenos y tendrán más importancia en el entrenamiento del aprendiz débil f_2. Se "llama" al perceptrón, el cual lleva a cabo su algoritmo de aprendizaje como se presentó en la Figura 2.3 y da como resultado el modelo de regresión f_2 que se presenta en la Figura 2.9c como los puntos verdes. Se calcula el error relativo absoluto como se presenta en (2) y a continuación se calcula la tasa de error de f_1 según (3), para este caso se consideran 2 muestras correctas. Finalmente se actualiza la función de distribución de probabilidad de pesos de las muestras ($D_3(i)$) que será usada para el entrenamiento del aprendiz débil de la iteración 3.

- $it = 3$

En la tercera iteración se aumenta el peso de las muestras "erróneas" de la iteración anterior, lo cual se refleja en la distribución de pesos $D_3(i)$ y se puede observar en la Figura 2.9d, las muestras erróneas se presentan como los círculos rojos rellenos y tendrán más importancia en el entrenamiento del aprendiz débil f_3. Se "llama" al perceptrón, el cual lleva a cabo su algoritmo de aprendizaje como se presentó en la Figura 2.3 y da como resultado el modelo de regresión f_3 que se presenta en la Figura 2.9d como los puntos verdes. Se calcula el error relativo absoluto como se presenta en (2) y a continuación se calcula la tasa de error de f_1 según (3), para este caso se consideran 3 muestras correctas. Finalmente se actualiza la función de distribución de probabilidad de pesos de las muestras ($D_4(i)$) que será usada para el entrenamiento del aprendiz débil de la iteración 4.

Al evaluar el conjunto de aprendices débiles de la iteración tres, correspondiente a f_{fin}, calculado según (5) este soluciona de manera satisfactoria el problema propuesto, el modelo conjunto final se presenta en la .

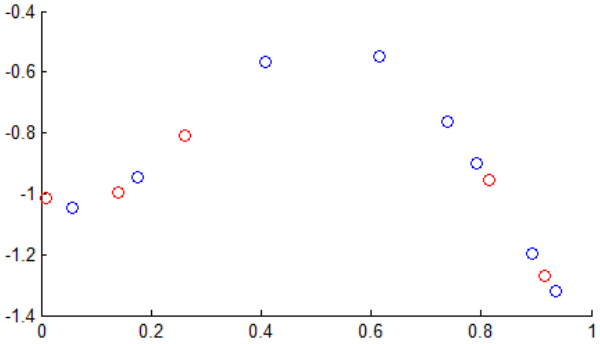


Figura 2.9a Datos a aproximar

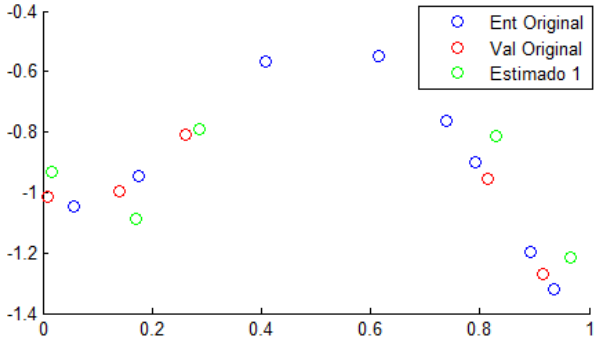


Figura 2.9b Aprendiz débil 1

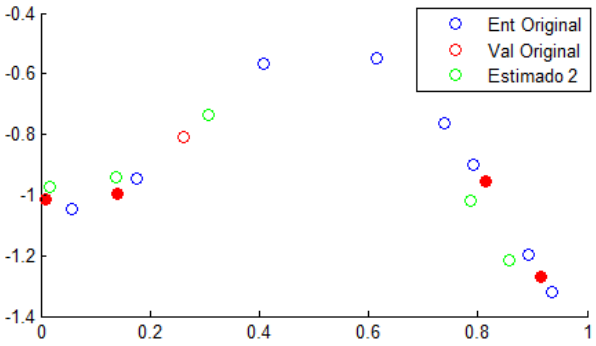


Figura 2.9c Aprendiz débil 2

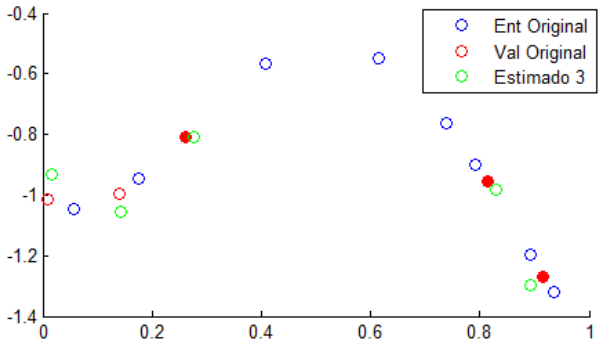


Figura 2.9d Aprendiz débil 3

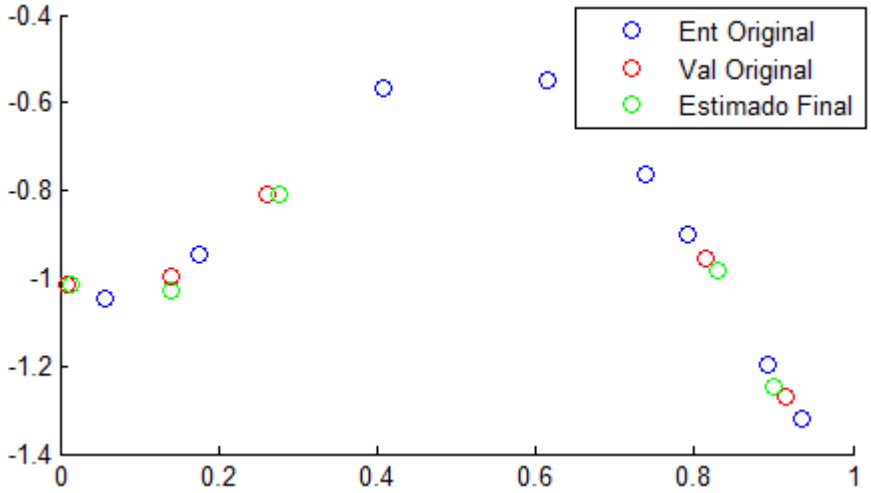


Figura 2.10 Modelo conjunto final

La precisión del modelo está relacionado con el valor escogido del discriminador de muestras correctas e incorrectas φ, entre menor sea el valor de este parámetro mayor será la precisión del modelo conjunto, pero si se escoge un valor muy pequeño, el modelo se vuelve inestable y no converge en ningún momento puesto que todas las muestras las consideraría incorrectas, lo mismo sucede en el caso de un valor muy grande todas las muestras se consideran correctas y no converge [38] [60].

2.2.4 AdaBoost.RT con φ auto-adaptativo

El algoritmo AdaBoost.RT con φ auto-adaptativo hace uso del algoritmo AdaBoost.RT (descrito en la Figura 2.8) con modificaciones para hacer que el valor de φ sea auto-adaptativo de acuerdo a la evolución del error del modelo a través de cada iteración. El valor de φ aumentará mientras la tasa de error de la iteración actual (ε_t) sea mayor que la de la iteración anterior ($\varepsilon_t - 1$), de la misma manera, si el error ε_t es menor que el error $\varepsilon_t - 1$, el valor de φ disminuirá. La adaptación del valor de φ es ajustado durante todas las iteraciones del algoritmo AdaBoost.RT. De esta manera se puede superar la limitación del algoritmo original de AdaBoost.RT atribuida a la estimación del valor de φ. En la investigación de Solomatine, en la cual se presentó el algoritmo original de AdaBoost.RT [57], las conclusiones muestran que el algoritmo es estable, mientras $0 < \varphi < 0.4$, si el valor es mayor de 0.4

AdaBoost.RT se vuelve inestable debido al overfitting y al ruido propio de Boosting. Por este motivo, el valor inicial de φ se escoge entre 0 y 0.4 al principio del algoritmo.

El modelo de auto-adaptación de φ utilizado, fue tomado de [38] y se encuentra descrito a continuación. Primero es necesario calcular el error raíz cuadrático promedio (RMSE) de la salida de cada iteración, como se muestra en (6)

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{y_i} - y_i)^2} \tag{6}$$

Donde,
m es el número de muestras
$\widehat{y_i}$ es el valor esperado
y_i es el valor predicho
Para el algoritmo el error RMSE se denomina e

.El valor de φ disminuirá, mientras $e_t < e_t - 1$, de la misma manera el valor de φ aumentará, mientras $e_t > e_t - 1$. El detalle del cambio es mostrado en (7).

$$\begin{cases} \varphi_{t+1} = \varphi_t * (1 - \lambda), & mientras\ e_t < e_{t-1} \\ \varphi_{t+1} = \varphi_t * (1 + \lambda), & mientras\ e_t > e_{t-1} \end{cases} \tag{7}$$

Donde λ es relativo a la tasa de cambio del RMSE

$$\lambda = r * \left|\frac{e_t - e_{t-1}}{e_t}\right| \tag{8}$$

El valor por defecto de r es 0.5. Usando este método de adaptabilidad de φ, no es necesario seleccionar el valor del límite experimentalmente.

2.3 SERIES DE TIEMPO

Los datos obtenidos a partir de la observación y colocados de manera secuencial a través del tiempo son muy comunes en todas las áreas del conocimiento y se les denomina series de tiempo, pero el verdadero propósito de las series de tiempo es su análisis y generalmente tiene dos objetivos: primero, entender o modelar el mecanismo estocástico que da lugar a la serie previamente observada y/o segundo, predecir valores futuros de la serie basado en la historia de la misma y posiblemente, otras series o factores relacionados [61].Una característica algo única de las series de tiempo y sus modelos es que normalmente no se puede suponer que las observaciones surgen independientemente. La clave del análisis de series de tiempo es la dependencia presente entre señales y por tanto son necesarios modelos que tengan en cuenta la dependencia. En la Figura 2.11, se presenta un ejemplo de serie de tiempo, la cual describe la velocidad del viento a través del tiempo.

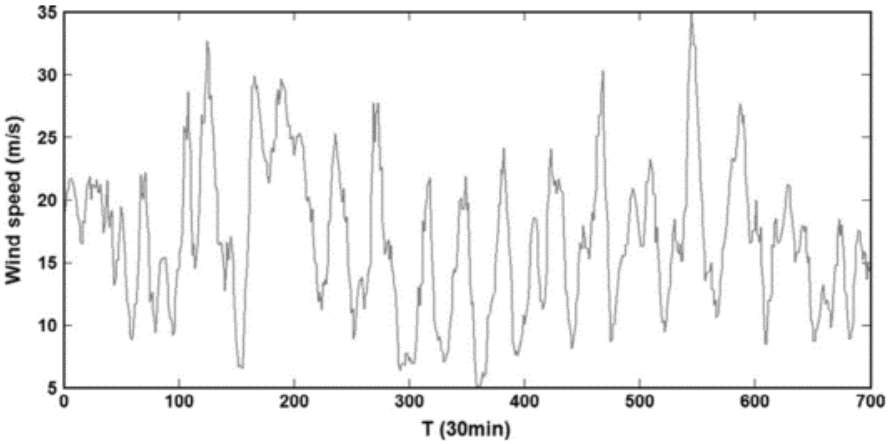


Figura 2.11. Ejemplo de serie de tiempo. Tomada de [37]

2.3.1 Componentes de las series de tiempo

Las series de tiempo pueden ser descompuestas en componentes básicas. Habitualmente se encuentran tres componentes, las cuales sobrepuestas o actuando en conjunto, contribuyen a los cambios observados en un período de tiempo y dan a la serie su aspecto.

2.3.1.1 Tendencia

La tendencia secular o tendencia a largo plazo de una serie es por lo común el resultado de factores a largo plazo. Se puede definir como un cambio constante en la misma dirección a largo plazo que se produce en relación al nivel medio o el cambio a largo plazo de la media. En términos intuitivos, la tendencia de una serie de tiempo caracteriza el patrón gradual y consistente de las variaciones de la serie, que se consideran consecuencia de acciones constantes que afectan a la serie en un crecimiento o decrecimiento de la misma [62].

2.3.1.2 Variación Estacional

El componente de la serie de tiempo que representa la variabilidad en los datos debida a influencias de las estaciones, se llama componente estacional. Esta variación corresponde a los movimientos de la serie que transcurren año tras año en los mismos meses (o trimestres) del año, bien sea con la misma intensidad a parecida. Esto quiere decir que la serie presenta una periodicidad, estos tipos de efectos son fáciles de medir y de ser necesario, es posible eliminarlos del conjunto de datos [62].

2.3.1.3 Variación Irregular o Aleatoria

Se debe a factores a corto plazo, imprevisibles y no recurrentes que afectan a la serie de tiempo. Este componente explica la variabilidad aleatoria de la serie, es impredecible, es decir, no se puede esperar predecir el impacto de esta sobre la serie de tiempo [62].

Debido a las componentes presentes en las series de tiempo, principalmente a la componente aleatoria, encontrar un modelo apropiado para las series de tiempo no es una tarea fácil, pero se puede dividir el proceso en tres pasos, de los cuales cada uno puede ser usado varias veces. Especificación o identificación, ajuste y diagnóstico del modelo. En la especificación o identificación del modelo se seleccionan las clases de modelos de series de tiempo que pueden ser apropiadas para la serie dada.

En la escogencia del modelo, se debe tener en cuenta la simplicidad del mismo, es decir, que requiera el menor número de parámetros que representan adecuadamente la serie de tiempo. El modelo tendrá inevitablemente uno o más parámetros los cuáles deben ser estimados a partir de la serie de tiempo observada [61]. El ajuste del modelo consiste en encontrar el mejor estimado de los valores desconocidos y el diagnóstico del modelo consiste en medir la calidad del modelo con parámetros que se han determinado previamente.

2.3.2 Series económicas

Las series económicas son series de tiempo tomadas de procesos económicos, típicos ejemplos de estas incluyen agregados macroeconómicos, precios o tasa de interés. Ese tipo de series se caracteriza porque poseen dependencia en serie, por lo que muestreos aleatorios son inapropiados. Normalmente suelen estar disponibles en una frecuencia baja (anual, trimestral o mensual) a excepción de los datos financieros los cuales están disponibles a frecuencias altas (semanal, diaria, por hora o por transacción), por lo que el tamaño de las bases de datos puede ser bastante grande [62]. Dentro de los datos financieros hacen parte los índices bursátiles.

Un índice bursátil es un indicador estadístico que refleja la evolución de los activos más representativos dentro de los mercados de renta variable (acciones), renta fija (deuda, bonos) o el mercado monetario en el caso colombiano, en un determinado periodo de tiempo. Se considera que los índices son el termómetro de la economía debido a que reflejan las rentabilidades promedio de los valores que lo componen, así como también permite analizar el movimiento del mercado en períodos anteriores [63].

2.3.2.1 Índice Bursátil COLCAP

El COLCAP es un índice de capitalización que refleja la variación de los precios de las 20 acciones más líquidas de la Bolsa de Valores de Colombia (BVC), donde la participación de cada una se establece por el valor de la capitalización bursátil ajustada[7] de cada compañía. El valor del índice se calcula haciendo la sumatoria de cada acción que conforma el índice por el peso que tiene dentro de la misma ajustado por un factor de enlace [42]. En (9) se presenta la fórmula para el cálculo del índice.

$$I(t) = E \sum_{i=1}^{n} W_i \, P_i(t) \tag{9}$$

donde $I(t)$ es el valor del índice en el instante t, E es el factor de enlace mediante el cual se da continuidad al índice, n es el número de acciones en el índice, W_i es el ponderador de la i-ésima acción y P_i es el precio de la misma. El valor base $I(t)$ con el que inició el COLCAP en la apertura de la bolsa del 4 de Enero de 2008 fue de 1.000 puntos.

La recomposición del índice consiste en la selección de las acciones que conformarán la canasta de acciones del índice durante el siguiente año. En el proceso de recomposición se determina, igualmente, la participación en el índice de cada acción seleccionada para el siguiente trimestre. La recomposición del COLCAP se realizará, después del cierre del mercado, el último día hábil del mes

[7] La capitalización bursátil ajustada hace referencia a la porción de la compañía que no se encuentra en manos de inversionistas con interés de control [63].

de octubre y estará vigente entre el primer día hábil de noviembre del mismo año y el último día hábil de octubre del año siguiente [42].

El rebalanceo del índice consiste en determinar la participación de cada acción en la canasta de acciones del índice. El rebalanceo del COLCAP se realizará el último día hábil de los meses de enero, abril y julio de cada año. Como resultado del rebalanceo se ajustan los ponderadores W_i de las acciones que conforman el índice para reflejar los cambios en la capitalización bursátil ajustada de cada acción [42]. La participación máxima que puede tener una acción en el índice en la fecha de cálculo de la canasta es del 20%. Luego del rebalanceo toma gran importancia el factor de enlace dado que es el número que permite dar continuidad al valor del índice después de aplicar procesos bien sea de recomposición o rebalanceo [42].

El índice COLCAP ha sido investigado anteriormente. En el primer caso se modela la serie de retornos del índice por medio de dos modelos de Heterocedasticidad Condicional Generalizados (GARCH), el primero desde una perspectiva clásica y el segundo a partir de un enfoque Bayesiano [44]. En el segundo caso, el índice fue predicho por medio ANN del tipo Back-Propagation, ANN del tipo NARX y redes mixtas que incluyen un pre-procesamiento de los datos, con el método DWT. Como vectores de entrada se probaron varias combinaciones generando diferentes arquitecturas de ANN, incluyendo Medias Móviles, el indicador técnico MACD, el RSI y el vector de la serie de tiempo a pronosticar, los cuales fueron normalizados a $\{-1, +1\}$. Para evaluar las distintas arquitecturas se hizo uso de coeficiente de correlación múltiple, coeficiente de determinación R^2, R^2 ajustado, error típico, MAE, MAPE RMSE NMSE y MAD.

2.4 PÉNDULO INVERTIDO

El péndulo invertido es un sistema altamente no lineal e inestable en lazo abierto, esto quiere decir que las técnicas de control lineal estándar no pueden modelar las dinámicas no lineales del sistema, lo que lo hace un proceso idóneo para probar prototipos de controladores. El sistema consiste de un péndulo articulado sobre un carro que es libre de moverse únicamente en la dirección x. Las ecuaciones del sistema dinámico pueden ser obtenidas usando Ecuaciones de LaGrange [64]. Haciendo uso de este método se pueden determinar las ecuaciones dinámicas de un sistema como el péndulo invertido. En la Figura 2.12 se presenta el diagrama del cuerpo libre del péndulo invertido.

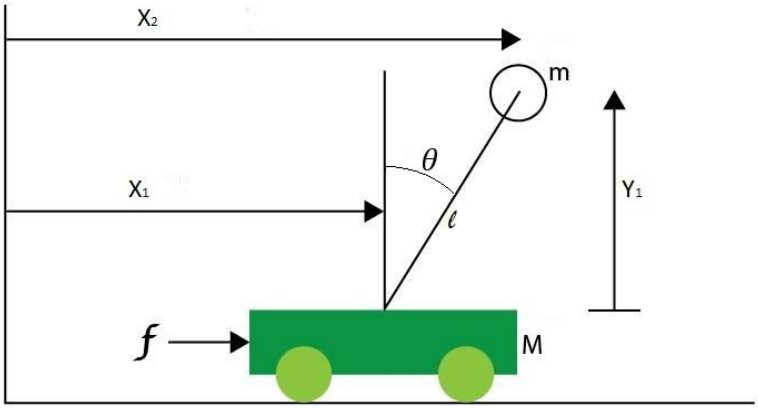


Figura 2.12. Diagrama del cuerpo invertido del péndulo invertido

Donde M es la masa de carta, m es la masa de la polea, l es la longitud de la polea y f es la fuerza de control.

Las ecuaciones de LaGrange usan la energía cinética y potencial del sistema para determinar las ecuaciones dinámicas del sistema del carro-polea. El péndulo invertido puede ser modelado matemáticamente a través de un sistema de ecuaciones de LaGrange, en [47] presentan de (10) a (20) la explicación del modelo matemático, a continuación se presenta el desarrollo de las ecuaciones de LaGrange.

La energía cinética del carro C_1 se presenta en (10). La polea se puede mover tanto en dirección horizontal como vertical, por tanto la energía cinética de la polea C_2 queda como se muestra en (11).

$$C_1 = \frac{1}{2} M\dot{x}_1{}^2 \tag{10}$$

$$C_2 = \frac{1}{2} m\left(\dot{x}_2{}^2 + \dot{y}_1{}^2\right) \tag{11}$$

Del diagrama de cuerpo libre presentado en la Figura 2.12, se deducen las ecuaciones de x_2 y y_1 y sus respectivas derivadas, las cuales se presentan en (12) y (13).

$$x_2 = x_1 + l\sin\theta \qquad \dot{x}_2 = \dot{x}_1 + l\dot{\theta}\cos\theta \tag{12}$$

$$y_1 = l\cos\theta \qquad \dot{y}_1 = -l\dot{\theta}\sin\theta \tag{13}$$

La energía cinética total C del sistema se presenta en (11)

$$C = C_1 + C_2 = \frac{1}{2}\left[M\dot{x}_1{}^2 + m\left(\dot{x}_2{}^2 + \dot{y}_1{}^2\right)\right] \tag{14}$$

Reemplazando (12) y (13) en (14), da paso a (15).

$$T = \frac{1}{2}M\dot{x}_1{}^2 + \frac{1}{2}m\left[\dot{x}_1{}^2 + 2\dot{x}_1\dot{\theta}l\cos\theta + l^2\dot{\theta}^2\right] \tag{15}$$

La energía potencial P del sistema, se encuentra almacenada en el péndulo, por tanto,

$$P = mgy_1 = mgl\cos\theta \tag{16}$$

En la mecánica clásica, la función de LaGrange está definida como

$$L = C - P = \frac{1}{2}(M+m)\dot{x}_1{}^2 + ml\dot{x}_1\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 - mgl\cos\theta \tag{17}$$

Las variables de estado del sistema con y y θ, por tanto las ecuaciones de LaGrange son

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_1}\right) - \frac{\partial L}{\partial x_1} = f \qquad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \tag{18}$$

Por tanto

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_1}\right) - \frac{\partial L}{\partial x_1} = (\boldsymbol{M+m})\ddot{x}_1 + \boldsymbol{ml}\left(\ddot{\theta}\boldsymbol{cos\theta} - \dot{\boldsymbol{\theta}}^{\boldsymbol{2}}\boldsymbol{\sin\theta}\right) = \boldsymbol{f} \tag{19}$$

Considerando θ, se obtiene

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = \boldsymbol{ml}(\ddot{x}_1\boldsymbol{\cos\theta} - \boldsymbol{g}\sin\boldsymbol{\theta} + \boldsymbol{l}\ddot{\boldsymbol{\theta}}) = \boldsymbol{0} \tag{20}$$

Como se puede observar a partir de las ecuaciones que describen la dinámica del sistema, (19) y (20), el sistema del péndulo invertido es uno de los problemas más difíciles en el campo de ingeniería de control, dado que es inestable, no lineal y de alto orden, por esto, los métodos de control del péndulo invertido pueden ser aplicados a resolver otros problemas de control similares, por ejemplo, mantener el balance de un autómata, el control de un cohete, entre otras [65]. Por este motivo se ha dado mucha atención a explorar una solución de control óptima al problema del péndulo invertido.

Para presentar mejor el comportamiento del sistema del péndulo invertido se lleva a cabo la simulación del modelo a partir de las variables de estado presentadas anteriormente en (19) y (20) y se le aplica al modelo un impulso unitario. En la Figura 2.13 se presenta el comportamiento del ángulo del péndulo (θ) en el modelo, a partir de la cual se puede observar que el péndulo decae rápidamente de la posición deseada (0°) a la posición de equilibrio (-180°) y se queda en esta posición.

En la Figura 2.14 se presenta el comportamiento de la posición del carro (x) , a partir de la cual se observa que debido al impulso unitario aplicado como fuerza, se inicia el movimiento del carro y dado que en el modelo no se considera fricción, el carro sigue en movimiento indefinidamente. A partir de las figuras presentadas, se observa que el péndulo invertido es un sistema no lineal e inestable.

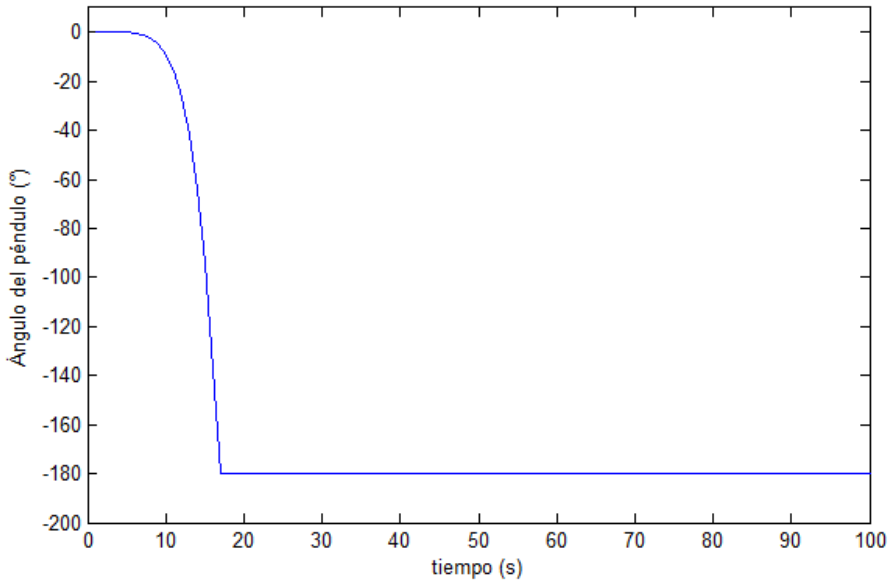


Figura 2.13. Comportamiento de θ en el sistema del péndulo invertido

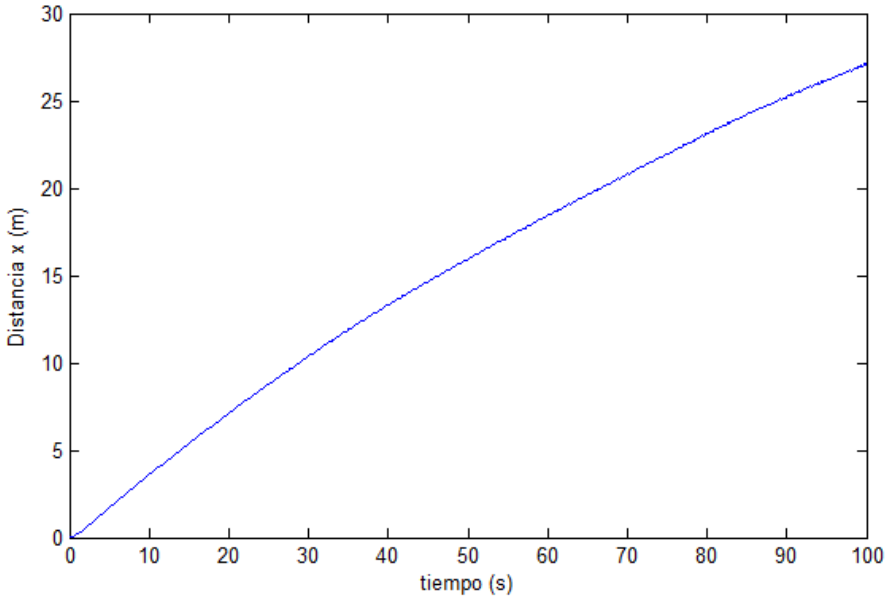


Figura 2.14. Comportamiento de x en el sistema del péndulo invertido

2.5 ANTECEDENTES

2.5.1 Regresión

El objetivo principal de la regresión es hallar las relaciones de una variable dependiente o de salida con un conjunto de variables independientes. Existen diversos métodos de resolver este tipo de problemas, entre los que destacan los métodos estadísticos [13], [14], estocásticos [15], no lineales [16], redes neuronales[66] debido a la capacidad que tienen este tipo de modelos ante el ruido, incertidumbre y no linealidades. El interés de este proyecto radica en regresión por medio de métodos

de Inteligencia Artificial, como redes neuronales y AdaBoost, por esto se hace énfasis en antecedentes de estas técnicas. El problema de regresión basado en Inteligencia Artificial puede ser descrito por medio de diagrama de bloques como se muestra en la Figura 2.15.

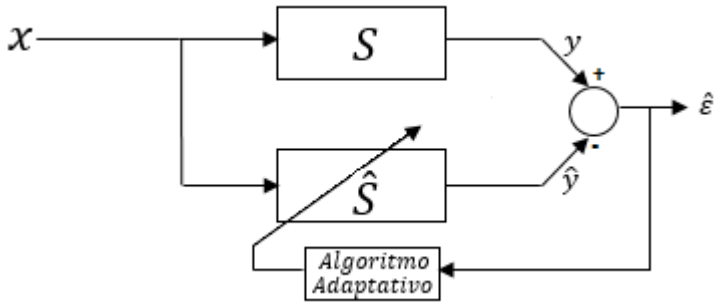


Figura 2.15. Modelo de bloques del problema de regresión basado en redes neuronales

Donde x son los parámetros de entrada del sistema, S el modelo que describe el sistema, y la salida del sistema, $\hat{S}$ el modelo aproximado del sistema el cual se adapta al problema por medio de un algoritmo adaptativo, $\hat{y}$ la salida aproximada y $\hat{\varepsilon}$ el error que es la diferencia entre la salida real y la aproximada del sistema. Esta señal de error es retroalimentada para ajustar los parámetros del algoritmo adaptativo, que a su vez cambiará parámetros del modelo aproximado del sistema, de tal manera que se minimice la diferencia entre la salida real del sistema y la salida aproximada.

2.5.1.1 Aplicación de redes neuronales en problemas de regresión

Las redes neuronales han tenido una amplia aplicación en problemas de regresión de múltiples áreas de conocimiento. En [66] se lleva a cabo la predicción de carga a corto plazo haciendo uso de ANNs con Back Propagation en donde se varía la función de activación, el número de capas ocultas, la cantidad de neuronas por capa oculta y la función de entrenamiento, luego de entrenar las diferentes redes neuronales se llevó a cabo una comparación de la mejor arquitectura con un modelo ANFIS. Para evaluar el desempeño de los modelos fue usado el error medio de porcentaje absoluto (MAPE). En [23] se realizó la predicción del flujo de entrada de aire a un motor de gasolina de aire haciendo uso de redes neuronales debido a la alta no linealidad del sistema,, una con Back Propagation y la otra con Funciones de Base Radial (RBF), para escoger la red con mejor desempeño se hizo uso de error MSE, MAE y MRE, donde la Red Neuronal entrenada con Back Propagation obtuvo mejores resultados. En [24] se llevó a cabo la predicción de cáncer de próstata por medio de regresión logística y ANN con Back Propagation de capa simple, en donde se varió la cantidad de neuronas de la capa oculta, el desempeño de los modelos se midió por medio del índice Youden, la sensibilidad y la especificidad.

2.5.1.2 Aplicación de AdaBoost.RT en problemas de regresión

El algoritmo original de AdaBoost.RT es relativamente nuevo, dado que fue desarrollado en el 2004, por lo cual ha sido ampliamente utilizado en diversos tipos de problemas de regresión, donde se puede observar que el mayor inconveniente de este algoritmo es la inicialización del límite φ [35]-[37]. En cuanto al algoritmo AdaBoost.RT con φ auto-adaptativo, el cual se usa en el desarrollo del proyecto, sólo ha sido usado para la predicción de temperatura requerida en un horno para la fundición de acero. La predicción de la temperatura se llevó a cabo por medio del algoritmo AdaBoost.RT con

φ auto adaptativo utilizando como aprendiz débil un algoritmo de aprendizaje de máxima extremo (ELM por sus siglas en inglés) para el entrenamiento de una red neuronal de una capa oculta, se usó una base de datos de 200 datos, de los cuales el 80% fue usado para entrenamiento y los datos restantes para validación del modelo. Se compararon 3 modelos de Inteligencia Artificial, un modelo ELM simple donde las entradas del modelo fueron los 7 factores más importantes que influyen en la temperatura, el segundo fue el modelo conjunto de ELM, es decir, aquel creado usando AdaBoost.RT en el cual los parámetros de entrenamiento de cada clasificador débil, fueron los mismos usados para entrenar el modelo ELM y el algoritmo AdaBoost.RT constó de 10 iteraciones, es decir, el modelo "fuerte" se compone de 10 regresores "débiles" [38]

2.5.1.3 Aplicación de predicción de índices económicos

La predicción de índices económicos es uno de los problemas más importantes en los mercados financieros. La alta complejidad y la naturaleza incierta del mercado financiero han dejado a los inversionistas alarmados. Los corredores de inversiones y los inversionistas individuales están continuamente en la búsqueda de una manera confiable para predecir tendencias en el mercado de índices. Sin embargo, el mercado de índices es muy complejo debido a su naturaleza, y ha sido objeto de estudio para modelar sus características en múltiples ocasiones.

Tradicionalmente la metodología básica para la construcción del modelo, para series de tiempo financieras y económicas ha sido estadístico, lo cual no es un proceso sencillo, es necesario tener conocimientos avanzados tanto del problema como de estadística. Además de esto, la serie necesita cumplir con ciertos requisitos para poder construir el modelo estadístico [67]. De acuerdo a múltiples autores, los métodos estadísticos tradicionales fallan en la predicción de series de tiempo financieras debido al comportamiento dinámico del mercado de índices [68]. A continuación se presenta una serie de diversos modelos usados para la predicción de índices económicos.

En [69], se lleva a cabo la predicción del precio de cierre del índice DELL cotizado en la bolsa de valores de New York por medio de un método estadístico y un método de Inteligencia Artificial: primero, un promedio móvil integrado auto regresivo (ARIMA, por sus siglas en inglés, Autoregressive Integrated Moving Average) comúnmente usado en análisis y predicción de series de tiempo, debido a que en problemas con incertidumbres, como las series de tiempo económicas, dado que no asume conocimiento de ningún modelo o relaciones presentes en el problema como lo hacen otros métodos. En segundo lugar, se hace uso de Redes Neuronales para llevar a cabo la predicción debido a que resuelven de manera satisfactoria las altas no linealidades y problemas variantes en el tiempo como los son las series de económicas. Se escoge el mejor modelo de predicción para cada caso por medio del error promedio absoluto porcentual (MAPE). Para construir los diversos modelos de predicción del índice DELL con ARIMA, se determinan los parámetros propios a prueba y error y se escoge aquel con $MAPE = 15.0406\%$ que es el de menor error. Para el caso de la red neuronal se hace uso de una red con una capa oculta y se varía el número de neuronas de la capa oculta para obtener el mejor modelo, el cual consta de 17 neuronas en la capa oculta y tiene un error $MAPE = 11.1347\%$. A partir de los resultados, se concluyó que ambos modelos, tanto ARIMA como Red Neuronal puede alcanzar una buena predicción en problemas de la vida real y por tanto puede ser usada para realizar predicción confiable del precio de cierre índices económicos.

En [70], se lleva a cabo la predicción del índice de Taiwán por medio de redes neuronales recurrentes. Además se aplica un modelo de predicción gris y análisis de relaciones gris. El modelo

de predicción gris, es aplicado para predecir el valor del índice del día siguiente. El análisis de relaciones gris es usado para filtrar los índices técnicos cuantitativos más importantes. Luego, se lleva a cabo el entrenamiento y la predicción del precio de cierre del índice al día siguiente por medio de una red neuronal recurrente. Se entrenan dos tipos de redes: el primero haciendo uso solamente de indicadores técnicos calculados a través de la serie y el segundo toma las mismas entradas del modelo anterior pero le añade el valor del índice predicho por medio de predicción gris. Para ambos casos se lleva a cabo el entrenamiento de las redes con una capa oculta y variando el número de neuronas de esta. Como medida de desempeño para escoger los mejores modelos de predicción se hace uso del error cuadrático medio (MSE). El mejor modelo de la red neuronal recurrente tiene un error del 0.0932, mientras el modelo de red incluyendo la predicción gris tiene un error del 0.0618. A partir de los resultados, se concluyó que el modelo de predicción gris puede ser adoptado como un nuevo tipo de índice técnico en la predicción de series de tiempo económicas por medio de redes neuronales artificiales.

En [67], se lleva a cabo la predicción del índice de la bolsa de valores de Corea por medio de redes neuronales combinadas con métodos computacionales evolutivos como Algoritmos Genéticos (GA, debido a sus siglas en inglés, Genetic Algorithms) y "Annealing" simulado (SA, por sus siglas en inglés, Simulated Annealing). Se lleva a cabo el entrenamiento de las redes neuronales haciendo uso de los algoritmos evolutivos, GA, SA y un modelo híbrido de los dos para buscar los pesos óptimos y el número de neuronas en la capa oculta de la red neuronal. La capa de entrada tiene el mismo número de neuronas que la cantidad de entradas del problema. La capa oculta contiene un número de neuronas seleccionado por el entrenamiento del algoritmo y se tiene una sola neurona en la capa de salida. Como medida de desempeño se hace uso del error promedio absoluto porcentual (MAPE), donde el mejor modelo de red neuronal con GA posee 6 neuronas en la capa oculta y tiene un error $MAPE = 30.173\%$, en el caso de la red con SA la red tiene 4 neuronas en la capa oculta y un error $MAPE = 22.603\%$ y el modelo de red neuronal con híbrido de SA y GA tiene 13 neuronas en la capa oculta y un error $MAPE = 40.5445\%$. Los resultados sugieren que SA combinado con redes neuronales es un método prometedor para la predicción de índices económicos.

En [71], se realiza la predicción de la acción de General Motors por medio de 4 modelos. El primero de ellos se basa en un modelo ARIMA, el segundo hace uso de Support Vector Machine (SVM), en el tercer caso se suman los dos modelos anteriores (SVM+ARIMA) y finalmente es un modelo híbrido de SVM y ARIMA propuesto en [71]. En el caso de los modelos ARIMA se dividen de tres fases: identificación del modelo, estimación de parámetros y diagnóstico. Para el caso de los modelos basados en SVM se realiza el ajuste de parámetros y aquel que presente el menor error MAPE es seleccionado como el mejor modelo. Para los modelos híbridos, ARIMA fue utilizado para filtrar los patrones lineales de la base de datos, luego el error de ARIMA es la entrada de SVM, cuyo objetivo es minimizar el error de ARIMA. En cuanto al modelo ARIMA+SVM, se hace uso de los mejores modelos ARIMA y SVM y se suman. Para ARIMA el mejor modelo tiene un error $MAPE = 14.214\%$, para SVM el error $MAPE$ es 12.654, para ARIMA+SVM el error $MAPE$ es 13.391 y finalmente el modelo híbrido tiene un error $MAPE = 7.55\%$. A partir de los resultados, se observó que dos modelos de predicción diferentes pueden mejorar el desempeño del otro al generar un modelo híbrido cuyo objetivo sea minimizar el error del otro.

En la Tabla 2.1, se presenta un resumen de los antecedentes usados para la predicción de índices económicos.

Tabla 2.1. Resumen Antecedentes de predicción de índices Económicos

Referencia	Índice	Método de Predicción	Tipo de Error	Desempeño
Assessment and prediction of tropospheric ozone concentration levels using artificial neural networks [69]	DELL	ARIMA	MAPE (%)	11.1347
		ANN	MAPE (%)	15.0406
A Forecasting Approach for Stock Index Future Using Grey Theory and Neural Networks [70]	Taiwán	RNN	MSE	0.0932
		RNN con Predicción Gris	MSE	0.0618
Predicting Stock Index Using Neural Network Combined with Evolutionary Computation Methods [67]	Corea Price	NN con GA	MAPE (%)	30.1737
		NN con SA	MAPE (%)	22.6034
		NN con GA+SA	MAPE (%)	40.5445
A hybrid ARIMA and support vector machines model in stock price forecasting [71]	General Motors Corporation	ARIMA	MAPE (%)	14.214
		SVM	MAPE (%)	12.654
		ARIMA+SVM	MAPE (%)	13.391
		Híbrido (ARIMA&SVM)	MAPE (%)	7.55%

2.5.2 Control Adaptativo

El control adaptativo es una técnica no lineal de control, cuya particularidad radica en que puede proveer un procedimiento de sintonización automático de los parámetros del controlador en lazo cerrado, lo cual es muy útil cuando los parámetros del modelo dinámico de la planta cambian de manera impredecible en el tiempo, así que para mantener un nivel de desempeño del sistema de control aceptable, se necesita un controlador adaptativo. En la Figura 2.16 se muestra en diagrama de bloques el problema de control adaptativo.

Donde $r(t)$ es la señal de referencia de la planta, $y_m(t)$ la salida del modelo de referencia, $\alpha(t)$ son los parámetros de ajuste del controlador, $u(t)$ es la señal de control, $y_p(t)$ es la salida de la planta y $e(t)$ es la señal de error, calculada como la diferencia entre la salida del modelo de referencia y la de la planta.

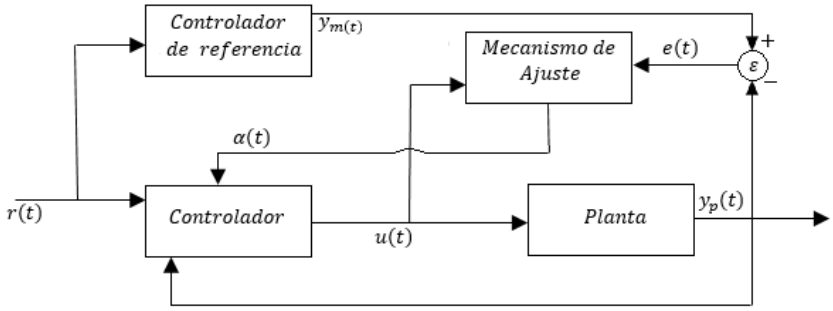


Figura 2.16. Diagrama de bloques Control Adaptativo

2.5.2.1 Aplicaciones de control en el sistema del Péndulo Invertido

El péndulo invertido es un sistema con dinámica altamente no lineal, el cual pertenece a un tipo de sistemas denominados "under-actuated", lo que quiere decir que el número de entradas de control es menor a la cantidad de grados de libertad del sistema [72], esto hace al péndulo invertido un punto de referencia clásico para diseñar, probar, evaluar y comparar diferentes tipos de métodos de control. En general, el problema de control consiste en obtener el modelo dinámico del sistema y usar este modelo para determinar leyes o estrategias de control para lograr la respuesta y el desempeño del sistema deseado.

En la literatura se ha implementado una gran cantidad de esquemas de control aplicadas al sistema dinámico del péndulo invertido [72]–[78], desde control clásico como PID[72], [75], hasta técnicas de Inteligencia Artificial como sistemas difusos [74] o redes neuronales [79], [80].

En [75] se hace uso de dos esquemas de control, PID y Regulador Cuadrático Lineal (LQR), para el diseño de estos se toman como parámetros de diseño overshoot (%OS) de la posición del carro(x) máximo de 22.5%, el tiempo de subida (T_r) menor a 1 segundo, el error de establecimiento de la posición del carro y del ángulo del péndulo menor a 5 segundos y un error de estado estacionario entre el 2%. Luego de diseñados los esquemas de control, el controlador LQR tiene un tiempo de establecimiento mayor, sin embargo, el porcentaje de rango del overshoot del controlador PID es menor, además de esto el controlador LQR supera el límite del máximo rango de Overshoot definido en 20° aunque logra estabilizar satisfactoriamente el péndulo invertido. En conclusión, los dos controladores tienen un error de estado estacionario 0, lo que los hace óptimos para el problema, pero se considera más adecuado el controlador LQR debido a que posee un menor tiempo de establecimiento. Los criterios de evaluación de los controladores se presentan en la *Tabla 2.2*.En este caso no se analizó el comportamiento de los controladores ante perturbaciones.

En [72], se hace uso de controladores PID y LQR para alcanzar una posición del carro deseada y estabilizar el péndulo en la posición invertida. El controlador LQR es diseñado usando el modelo lineal de espacio de estado del sistema y el PID se entrena a prueba y error observando la respuesta obtenida hasta hacerla óptima. Luego de diseñar los controladores PID y LQR, se hace uso de tres esquemas de control, 1. Control PID compuesto por dos PID uno para el control del ángulo y el otro para la posición del carro. 2. Dos PID (ángulo y posición) con LQR y 3. Un PID (ángulo o posición) con LQR. Se aplica un parámetro de perturbaciones de ruido blanco limitado en banda con potencia de 0.001. Como criterio de evaluación de los controladores se usó el valor absoluto máximo de cada variable de estado, los cuales se presentan en la Tabla 2.2, a partir de estos valores se puede concluir que la respuesta de las alternativas LQR+PID son mejores que el control PID, además de que la

respuesta de PID+LQR y 2PID+LQR son similares, aunque 2PID+LQR es ligeramente mejor, pero PID+LQR es estructuralmente más sencillo.

En [76], se realiza un estudio comparativo de cinco esquemas de control diferentes. Se realiza la comparación de controladores: PD, Regulador Cuadrático Lineal (LQR), no lineal, neuronal y difuso. Se muestra cómo cambia el tamaño de la "porción controlable" con los diferentes esquemas de control, para esto definen el *coeficiente de efectividad* como se presenta en la (21)

$$coef.efectividad = \frac{360° - (rango\ no\ controlable)}{360°} \tag{21}$$

Donde el rango no controlable, es aquel en el que el péndulo oscila. En caso de que el péndulo no oscile el rango no controlable es igual a cero y el coeficiente de efectividad sería 1. En la Tabla 2.2 se presentan los coeficientes de efectividad para cada caso, donde se puede observar que el controlador neuronal es el que tiene mayor coeficiente de efectividad y por tanto se considera el mejor controlador.

En la Tabla 2.2 se presenta un resumen de los antecedentes de controladores usados para el problema del péndulo invertido con el respectivo criterio de evaluación utilizado en cada referencia.

Tabla 2.2. Resumen antecedentes controladores

Referencia	**Criterio de evaluación**	**Tipo de Controlador**	**Desempeño**	
Performance comparison between LQR and PID controllers for an inverted pendulum system [75]**.**	Tiempo de establecimiento	PID	$4.48\ s$	
		LQR	$3.34\ s$	
	Rango del Overshoot	PID	$0.50° \rightarrow -4.39°$	
		Regulador Cuadrático Lineal (LQR)	$19.6° \rightarrow -46.5°$	
Optimal control of nonlinear inverted pendulum system using PID controller and LQR: Performance analysis without and with disturbance input [72]	Máximo valor absoluto de los estados del sistema	PID	θ	0.0046
			$\dot{\theta}$	-
			x	0.0976
			$\dot{x}$	-
			u	0.1402
		1PID+LQR	θ	0.0029
			$\dot{\theta}$	0.0067
			x	0.1
			$\dot{x}$	0.0331

			u	0.15
		2PID+LQR	θ	0.0029
			$\dot{\theta}$	0.0070
			x	0.1
			$\dot{x}$	0.0330
			u	0.15
Comparative Study of Motion Control Methods for a Nonlinear System **Comparative Study of Motion Control Methods for a Nonlinear System** [76]	Coeficiente de efectividad	PD	0.324	
		LQR	0.524	
		Red Neuronal	0.785	
		No lineal	0.248	
		Lógica difusa	0.349	

3 SERIE COLCAP

En este capítulo se presenta el análisis de la serie COLCAP en busca de estacionalidad, se lleva a cabo la selección de los parámetros de entrada del sistema predictor y el análisis de los mismos. Además, se define el método utilizado para obtener los modelos de predicción del índice COLCAP, así como las características usadas para el modelo de redes neuronales y de AdaBoost.RT con φ auto-adaptativo. Finalmente se presentan los resultados obtenidos al llevar a cabo las pruebas para cada uno de los modelos y se define el modelo óptimo para cada modelo de predicción.

COLCAP es el índice de mayor interés en la Bolsa de Valores de Colombia (BVC), dado que representa la variación de las 20 acciones más líquidas de esta. La serie de tiempo se obtiene a través del histórico del índice [63]. En la Figura 3.1 se observa la serie de tiempo de COLCAP utilizada en el desarrollo del proyecto, la cual consta de 2184 muestras que corresponde a los días hábiles de la BVC equivalentes al período de tiempo comprendido entre el 4 de Enero del 2008 hasta el 16 de Diciembre del 2016

Figura 3.1. Serie COLCAP 4 Enero 2008 - 16 Diciembre 2016 [48]

3.1 ANÁLIS DE LA SERIE DE TIEMPO COLCAP

Antes de llevar a cabo el análisis de los parámetros de entrada del sistema de predicción, es necesario analizar la serie de tiempo en busca de tendencia y componentes estacionales para comprobar la estacionalidad de la serie, dado que de estar presentes en las serie este tipo de características es necesario llevar a cabo procedimientos para eliminarlas, puesto que la presencia de estas en la serie requieren métodos específicos de predicción dado que le añaden complejidad al

modelo de predicción [68]. Para el análisis tanto de tendencia como de componentes estacionales se considera realizar un test de raíz unitaria para comprobar la estacionalidad de la serie [81].

3.1.1 Raíz Unitaria

Un proceso estocástico tiene una raíz unitaria si una de las raíces de la ecuación característica del proceso es 1, de ser así, se considera que el proceso es no estacionario. La raíz unitaria puede ser tanto creciente como decreciente, es decir, la media puede aumentar o disminuir con respecto al tiempo, dependiendo del signo de la raíz unitaria. Los procesos con raíces unitarias poseen características específicas, por ejemplo, tienen una varianza que depende del tiempo y diverge al infinito, o un valor aislado tiene efectos permanentes en el proceso. Existen test con el objetivo de determinar si una serie es no estacionaria y posee una raíz unitaria, en los cuales se considera una hipótesis nula que es generalmente definida como la presencia de una raíz unitaria, por tanto si se rechaza la hipótesis nula, la serie de tiempo a la que le fue aplicado el test es estacionaria, y en caso de fallar el rechazo de la hipótesis nula la serie de tiempo se considera no estacionaria [81]. Dos de los test más usados son el test Dickey-Fuller aumentado (ADF, por sus siglas en inglés) y el test KPSS.

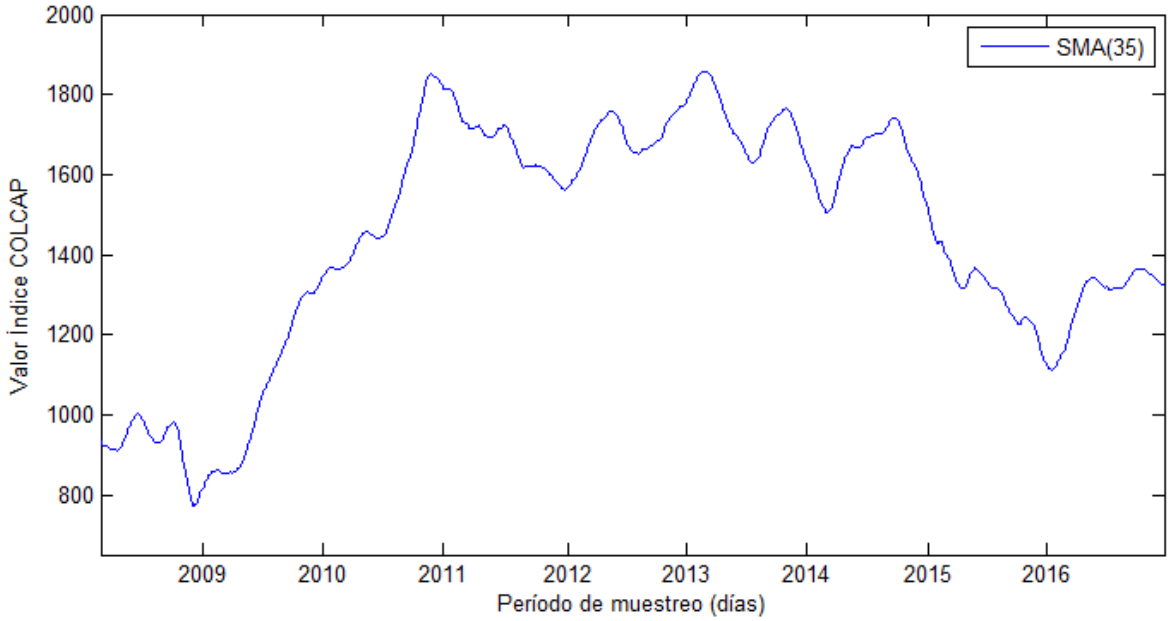


Figura 3.2. Promedio Móvil de la serie COLCAP

3.1.2 Tendencia

- Primero se busca eliminar la tendencia que puede ser definida como un cambio a largo plazo que se produce en relación al nivel medio o el cambio a largo plazo de la media, la cual puede ser identificada con un movimiento suave de la serie a largo plazo. Para esto se suele calcular un promedio móvil simple (SMA, por sus siglas en inglés) a largo plazo, normalmente mayor a 30, para suavizar la serie y poder ver la media de la serie a través del tiempo. En la Figura 3.2 se presenta el Promedio Móvil de la serie COLCAP con 35 muestras, en la cual se puede observar que la serie en el período de tiempo seleccionado no posee tendencia, dado que a través del tiempo el valor del índice posee un comportamiento variante en el cual hay sub-períodos en

los cuales el valor aumenta y otros en los que disminuye, lo que podría indicar tendencia a corto plazo, más no a largo plazo lo que indica que no se posee tendencia a largo plazo la serie con el período de tiempo seleccionado. Además de esto, como se mencionó anteriormente existen test de raíz unitaria que permiten comprobar la estacionalidad de la serie. Para el caso de determinar si una serie posee tendencia a largo plazo se suele usar el test KPSS.

- **Test KPSS**
  El test Kwiatkowski-Phillips-Schmidt-Shin es usado para probar la hipótesis nula de estacionalidad alrededor de una tendencia determinística contra la alternativa de una raíz unitaria. Se caracteriza porque a diferencia de los demás test de raíz unitaria, la presencia de una raíz unitaria no es la hipótesis nula sino una alternativa, es decir, no descarta raíces unitarias en la serie de tiempo, pero elimina la presencia de tendencia estacionaria en la serie. Por tanto, para descartar raíces unitarias, es necesario llevar a acabo otro test. El test se llevó a cabo en MatLab® y este tipo de test da como resultado un valor lógico donde 1 representa el rechazo de la tendencia estacionaria dando paso a la posibilidad de presencia de raíces unitarias y 0 representa fallo al rechazar la tendencia estacionaria, es decir, la serie posee una tendencia estacionaria más no excluye la presencia de raíces unitarias:
  Luego de aplicar el test KPSS a la serie, se obtuvo un rechazo a la hipótesis de tendencia estacionaria, lo que indica que la serie de tiempo analizada no posee tendencia a largo plazo y conlleva a la necesidad de aplicar otro test de raíz unitaria para descartar la presencia de raíces unitarias en la serie de tiempo.

3.1.3 Componentes estacionales

En segundo lugar se encuentra la componente estacional o también denominado periódico puesto que se observa como una repetición en la serie en un período de tiempo definido y normalmente se presenta por las estaciones. Estos tipos de efectos son fáciles de medir a partir de la serie suavizada, al igual que en el caso de la tendencia, por este motivo se hizo uso del promedio móvil tomado anteriormente para analizar la presencia de componentes estacionales en la serie a utilizar [81]. En la Figura 3.2 se puede observar que no hay patrones que se repitan periódicamente, teniendo en cuenta que un año corresponde a 242 muestras aproximadamente, por tanto se puede deducir que la serie no presenta componentes estacionales. Además de esto, para comprobar la estacionalidad de la serie de tiempo, se lleva a cabo un test de raíz unitaria denominado Test Dickey Fuller Aumentado (ADF; por sus siglas en inglés)

- **Test Dickey Fuller Aumentado**
  Prueba la hipótesis nula de presencia de raíz unitaria en una serie de tiempo. Es comúnmente usado cuando se tiene un amplio rango de muestras. La prueba se llevó a cabo en MatLab® y este test da como resultado un valor lógico con la decisión del rechazo a la raíz unitaria, en donde 1 equivale a rechazo de la hipótesis nula, es decir, la serie se considera estacionaria o 0 que equivale al fallo del rechazo de la hipótesis lo que quiere decir que la serie posee una raíz unitaria y por tanto se considera no estacionaria.
  Luego de llevar a cabo el test ADF, se obtuvo un rechazo a la hipótesis y dado que el test KPSS descartó la presencia de tendencia y el test ADF descarta la presencia de raíces unitarias se puede concluir que la serie COLCAP es estacionaria.

3.2 SELECCIÓN DE LOS PARÁMETROS DE ENTRADA

Luego de analizar la serie COLCAP es necesario determinar los parámetros de entrada para la predicción del índice, para esto se hizo un histórico de las acciones que han hecho parte de él desde que se empezó a cotizar. En este período de tiempo, el índice sufrió 36 rebalanceos y 8 recomposiciones conformada por 46 acciones y cada una ha tenido un porcentaje diferente a lo largo de cada rebalanceo. En la Tabla 3.1, se presentan las 46 acciones que han conformado el índice a lo largo del tiempo, la cantidad de veces que lo conformaron y el porcentaje absoluto que cada una tiene en el índice.

Tabla 3.1. Acciones que han conformado el índice COLCAP

	Nombre	Apariciones	Porcentaje Total
1	BCOLOMBIA	10	2.27925
2	BIOMAX	4	0.0105
3	BNA	6	0.0288
4	BOGOTA	15	1.18037
5	BVC	34	0.48809
6	CELSIA	13	0.71302
7	CEMARGOS	31	5.01062
8	CHOCOLATES	10	2.04359
9	CLH	16	0.9392
10	CNEC	24	0.30554
11	COLINVERS	14	1.42684
12	COLTEJER	8	0.14215
13	CONCONCRET	3	0.0277
14	CORFICOLCF	34	3.31525
15	ECOPETROL	35	16.6227
16	EEB	19	1.40574
17	ENKA	10	0.04762
18	ETB	23	0.42501
19	EXITO	36	4.03531
20	FABRICATO	21	0.3248
21	GRUPOAVAL	14	1.14614
22	GRUPOARGOS	14	3.18194
23	GRUPOSURA	23	7.04462
24	INTERBOLSA	13	0.26404
25	INVERARGOS	22	5.44895
26	ISA	36	6.52879
27	ISAGEN	36	3.57415
28	NUTRESA	22	4.16396
29	PAZRIO	3	0.03705
30	PFAVAL	20	2.2
31	PFAVH	14	0.32294
32	PFAVTA	7	0.13128
33	PFCARPAK	1	0.00477
34	PFBCOLOM	34	12.4427
35	PFBCREDITO	6	0.12446
36	PFCEMARGOS	12	1.32541
37	PFDAVVNDA	23	1.63879

38	PFGRUPOARG	7	0.67291
39	PFGRUPSURA	14	3.34736
40	PFHELMBANK	7	0.08819
41	PMGC	8	0.14128
42	PREC	24	3.62642
43	SIE	2	0.05719
44	SURAMINV	6	2.04047
45	TABLEMAC	20	0.20144
46	TERPEL	1	0.0222

Para visualizar mejor la frecuencia de aparición de cada una de las acciones dentro del índice, en la Figura 3.3 se presenta el histograma de aparición de las acciones que han compuesto el índice COLCAP en el período de tiempo escogido, en donde se puede observar que solo 7 acciones han formado parte del cálculo del índice entre 33 y 36 ocasiones.

A partir de los datos presentes en la Tabla 3.1 y el histograma de la Figura 3.3 se busca escoger las acciones que más información aportan sobre el índice, por este motivo se pre-seleccionan aquellas que hayan estado incluidas en el cálculo del índice por más de 32 rebalanceos. Se escoge el límite de 32 apariciones, dado que cada aparición dura 3 meses del índice, por tanto, si una acción no tiene aparición en 4 rebalanceos, esa acción no tiene efecto en 1 año del índice. Las acciones que afectan el cálculo del índice más de 32 veces se presentan en la Tabla 3.2.

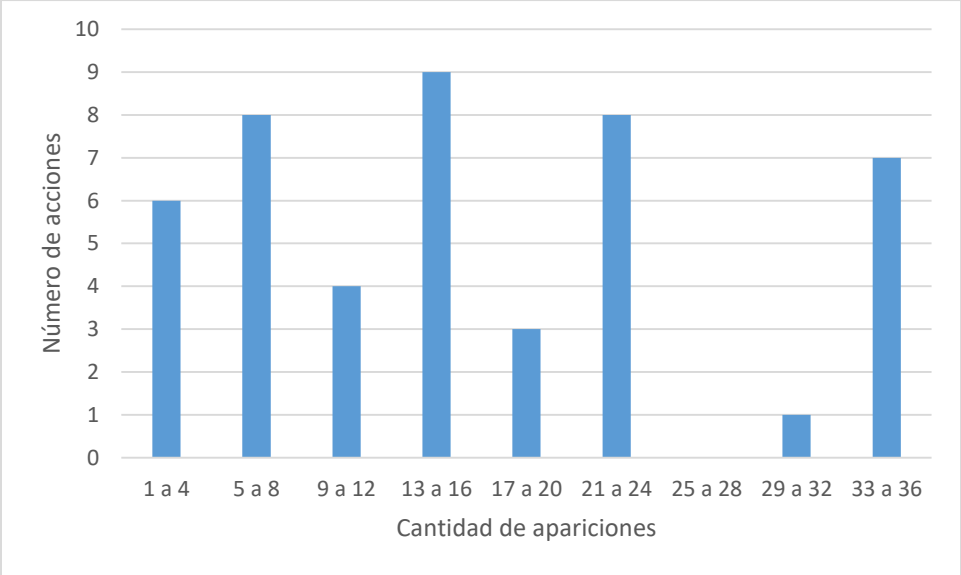


Figura 3.3. Histograma de las acciones que componen el índice COLCAP en su historia

Tabla 3.2. Acciones con 32 o más apariciones en el cálculo del índice COLCAP

ACCIÓN	PORCENTAJE
ECOPETROL	16.622682
PFBCOLOM	12.4427288
ISA	6.52878919

ÉXITO	4.03530537
ISAGEN	3.57415447
CORFICOLCF	3.31525326
BVC	0.48809373

A partir de la Tabla 3.2, se observa que las acciones ECOPETROL y PFBCOLOM aportaron en conjunto un 29.0654% del índice COLCAP en el período de tiempo seleccionado, de la misma manera se observa que ISA, ÉXITO, ISAGEN y CORFICOLCF tuvieron un aporte constante pero no tan notorio en el índice dado que cada una aporta entre el 6% y el 4% y aunque BVC fue una acción que estuvo presente en más del 90% de la cotización del índice pero su aporte fue menor al 0.5%, motivo por el cual no se tendrá en cuenta para la predicción del índice. De igual manera se descarta CORFICOLCF a pesar del aporte que tuvo para el índice debido a que no estuvo presente en 2 periodos de cotización del índice y estos períodos no fueron uno seguido del otro, sino se presentaron en diferentes años, lo que afecta la continuidad del índice en caso de usarse para la predicción. Por tanto, las acciones escogidas para la predicción del índice, debido a su permanencia e importancia en el cálculo del mismo son ECOPETROL, PFBCOLOM, ISA, ÉXITO, e ISAGEN. Además de las acciones que componen el índice COLCAP y teniendo en cuenta que es el índice económico más significante de la economía colombiana, se consideró importante incluir la tasa de cambio del peso colombiano (COP) con respecto al dólar (USD). Cada una de las series que son utilizados como parámetros de entrada cuenta con el mismo tamaño la serie COLCAP que es de 2184 muestras.

3.3 ANÁLISIS DE PARÁMETROS DE ENTRADA

Luego de haber escogido los parámetros de entrada para el sistema de predicción, es necesario realizar análisis sobre estos buscando posibles relaciones entre ellos o cuál aporta más información, para esto se llevan a cabo dos análisis: Análisis de Relaciones Gris y Análisis de Componentes Principales, cuya descripción matemática se presenta en el Anexo1.

3.3.1 Análisis de Relaciones Gris

El análisis relacional gris consta de tres pasos principales, el primero de ellos es pre-procesamiento de datos, en donde la serie de datos original (COLCAP) se representa como la referencia (x_0) y las series de las acciones usadas para la predicción como series comparativas (x_i). Existen dos métodos de normalización, "mayor mejor" y "menor mejor", para el caso de este proyecto se escogió "mayor menor", según (25), lo que indica que entre mayor sea el valor del grado de relación gris, mayor será la relación entre la serie de referencia y la comparativa. También es necesario normalizar la serie de referencia (x_0). El rango de los datos está ajustado de tal manera que esté dentro del rango [0,1].

El segundo paso es hallar el coeficiente relacional gris usando (26), dentro de la cual se hace uso del coeficiente $\zeta \in [0,1]$, el cual puede ser ajustado para tener una mejor distinción entre la serie de referencia normalizada y las series comparativas normalizadas. Normalmente $\zeta = 0.5$ dado que ofrece un efecto distinguidor moderado y estabilidad.

Luego de calcular el coeficiente relacional gris (γ_i), se calcula el grado relacional gris (GRG, por sus siglas en inglés), el cual está distribuido entre 0 y 1 y se calcula usando (27)

Generalmente $\gamma_i > 0.8$ indica una influencia marcada, $\gamma_i > 0.7$ una influencia relativamente marcada, $\gamma_i > 0.6$ una influencia notable, $\gamma_i > 0.5$ una influencia relevante y $\gamma_i < 0.5$ una influencia despreciable [82],

El análisis de relaciones gris se llevó a cabo sobre la base de datos utilizada usando 3 coeficientes de identificación diferentes $\zeta = 0.3\,, 0.5$ y 0.7, esto para observar el efecto del coeficiente en el análisis. Los resultados de este análisis se presentan en la Tabla 3.3

Tabla 3.3a. Análisis Relacional Gris con $\zeta = 0.3$

$\zeta = 0.3$		
PFCOLOMB	0.798899	Relativamente Marcada
ÉXITO	0.742272	Relativamente Marcada
ISA	0.646232	Notable
ECOPETROL	0.640715	Notable
ISAGEN	0.547123	Relevante
DOLAR	0.443130	Despreciable

Tabla 5b Análisis Relacional Gris con $\zeta = 0.5$

$\zeta = 0.5$		
PFCOLOMB	0.866182	Marcada
ÉXITO	0.822895	Marcada
ISA	0.742808	Relativamente Marcada
ECOPETROL	0.742254	Relativamente Marcada
ISAGEN	0.653328	Notable
DOLAR	0.555987	Relevante

Tabla 5c Análisis Relacional Gris con $\zeta = 0.7$

$\zeta = 0.7$		
PFCOLOMB	0.901979	Marcada
ÉXITO	0.867084	Marcada
ISA	0.800489	Marcada
ECOPETROL	0.798978	Relativamente Marcada
ISAGEN	0.719488	Relativamente Marcada
DÓLAR	0.630288	Notable

A partir de la Tabla 3.3, se puede observar que las acciones PFCOLOMB y ÉXITO tienen una influencia relativamente marcada en el caso de $\zeta = 0.3$ y en los otros dos casos tiene influencia marcada en la serie COLCAP, ISA tiene una influencia Notable en el caso de $\zeta = 0.3$, Relativamente marcada con $\zeta = 0.5$ y marcada con $\zeta = 0.7$, ECOPETROL tiene una influencia notable para $\zeta = 0.3$, y relativamente marcada para los otros dos casos, ISAGEN tiene una influencia relevante para $\zeta = 0.3$, notable para $\zeta = 0.5$ y relativamente marcada para $\zeta = 0.7$. Finalmente el DÓLAR presenta una influencia despreciable para $\zeta = 0.3$, relevante para $\zeta = 0.5$ y notable para $\zeta = 0.7$. También se puede observar que el valor de ζ afecta la magnitud del coeficiente relacional y la distinción de la

influencia de cada series, más no afecta el orden del grado relacional gris, puesto que en los tres casos se la acción con mayor influencia es PFCOLOM, seguida de ÉXITO, ISA, ECOPETRO, ISAGEN y finalmente el DÓLAR. Aunque para el caso de $\zeta = 0.3$, la serie DÓLAR tiene una influencia despreciable, se continúa con esta serie por la importancia que tiene para la bolsa de valores colombiana [43].

3.3.2 Análisis de Componentes Principales (PCA)

El análisis de componentes principales se trata de encontrar combinaciones lineales de las variables para conseguir un nuevo conjunto de variables que no posea correlación entre ellas y que posea la máxima varianza, llamadas Componentes Principales, ubicadas en orden decreciente de importancia. Se espera que las primeras variables del nuevo conjunto posean la mayor parte de la varianza para obtener una reducción de la cantidad de datos y de esta manera simplificar el problema a resolver [83]. En lo que concierne a este proyecto, no se busca crear una nueva base de datos para la predicción sino conocer el porcentaje de varianza que aporta cada una de las variables que serán usadas para predecir el índice y en caso de que una de las varianzas sea despreciable comparada con las demás, no se tenga en cuenta esta y así poder reducir la complejidad del problema.

El análisis de componentes principales se lleva a cabo en MatLab®, se hace a partir de los parámetros que serán usados para la predicción del índice y tiene como resultado los coeficientes de cada variable para cada uno de los componentes principales, además de un vector con el porcentaje de la varianza de cada componente principal. Se obtienen tantos componentes principales como variables de entrada. En este caso, se tienen 6 parámetros de entrada, por tanto se obtienen 6 componentes principales.

Los valores mostrados en la Tabla 3.4 representan los coeficientes que conforman cada uno de los componentes principales, siendo cada columna un componente principal y cada fila los coeficientes de la respectiva serie en cada uno de los componentes principales. Se puede observar que la participación de cada serie en un componente puede ser tanto positiva como negativa. Para el análisis que atañe a este proyecto, se desea saber el porcentaje de varianza que aporta cada serie, para esto se deben transformar los coeficientes de los componentes principales en el porcentaje que cada serie aporta a ese componente, por tanto se realiza la suma de cada participación de las series en un componente, sin importar el signo de la participación y se divide en la participación de todas las series en ese componente.

Tabla 3.4. Coeficientes Análisis de Componentes Principales

	PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**
PFCOLOMB	0.346767	0.567279	0.211356	-0.001472	-0.499642	-0.513449
ECOPETROL	0.55485428	-0.111323	-0.183152	0.733756	0.304711	-0.122278
ISAGEN	-0.108347	0.60622468	0.000548	-0.163993	0.757779	-0.140100
ISA	0.300412	-0.065918	0.874530	-0.02537	0.152705	0.341527
ÉXITO	0.515743	0.29968	-0.389144	-0.290208	-0.112195	0.629248
DÓLAR	-0.451564	0.451813	0.074475	0.591473	-0.21764	0.43496

En la Tabla 3.5 se muestra el porcentaje de participación de casa serie en cada uno de los componentes y en la Tabla 3.6 se presenta el porcentaje de varianza de cada uno de los componentes

principales. Para hallar el porcentaje de varianza de cada serie, se multiplica la participación de la serie en cada componente por el respectivo porcentaje de varianza de este componente.

Tabla 3.5. Porcentaje de participación de las series en PCA

	PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**
PFCOLOMB	15.22453	26.98442	12.19454	0.081544	24.4361	23.53575
ECOPETROL	24.36039	5.29543	10.56723	40.62247	14.90260	5.605069
ISAGEN	4.75688	28.83699	0.031641	9.079052	37.06097	6.422002
ISA	13.18934	3.135598	50.45733	1.404946	7.468392	15.65511
ÉXITO	22.64329	14.25560	22.45224	16.06662	5.487191	28.84382
DÓLAR	19.82554	21.49192	4.297000	32.74536	10.64464	19.93822

Tabla 3.6. Porcentaje de Varianza de cada Componente Principal

	PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**
Varianza (%)	55.079438	27.485671	11.719388	2.446567	2.350654	0.91828

Los porcentajes de varianza correspondientes a cada serie se presentan en la Tabla 3.7, donde se puede observar que el aporte de varianza entre las series es similar y el mínimo aporte lo tiene ISAGEN con un 11.7%, el cual no es despreciable, por este motivo se decide realizar la predicción del índice COLCAP a partir de las series de tiempo de estos 6 parámetros.

Tabla 3.7. Porcentaje de Varianza de cada Serie

ACCIÓN	VARIANZA (%)
ÉXITO	19.80824
DÓLAR	18.56502
PFCOLOMB	18.02409
ECOPETROL	17.5071
ISA	14.39343
ISAGEN	11.70209

3.4 MODELO DE PREDICCIÓN DEL ÍNDICE COLCAP

Uno de los objetivos principales de este proyecto es diseñar un sistema de predicción del índice COLCAP por medio de Redes Neuronales y otro sistema basado en el algoritmo AdaBoost.RT con φ auto-adaptativo. Para ambos sistemas se hace uso de una base de datos que consta de 6 parámetros, cada uno de ellos con 2182 muestras ubicados en una matriz de 2182x6, donde cada columna representa cada uno de los parámetros de entrada y está ubicado en el orden que se muestra en la Tabla 3.8.

Tabla 3.8. Distribución de la base de datos

Columna #	**1**	**2**	**3**	**4**	**5**	**6**
Acción	PFCOLOM	ECOPETROL	ISAGEN	ISA	EXITO	DÓLAR

Dado que tanto las redes neuronales como AdaBoost son algoritmos de Inteligencia Artificial supervisados, es necesario contar con la señal que se desea predecir, en este caso el índice COLCAP, es necesario tenerlo como parámetro de entrada para poder calcular el error, este, al igual que los demás parámetros de entrada consta de 2182 muestras. De acuerdo a la validación cruzada, se dividen los datos de muestra y se realiza el análisis de un subconjunto (denominado datos de entrenamiento) y se valida el análisis en el otro subconjunto (denominado datos de prueba), de forma que la función de aproximación se ajusta sólo con el conjunto de datos de entrenamiento y se calculan los valores de salida para el conjunto de datos de prueba (valores que no han sido analizados anteriormente). Normalmente la base de datos se separa en entrenamiento (70%) y validación (30%), debido al tipo de serie de tiempo que se busca predecir, no tiene sentido llegar a predecir la serie en un período demasiado extenso, por este motivo se decide predecir el índice en un período de un año, es decir, del 16 de Diciembre del 2015 al 16 de Diciembre del 2016. Finalmente es necesario resaltar que las bases de datos de entrenamiento y validación para los dos modelos de predicción, redes neuronales y AdaBoost, son las mismas.

De acuerdo al período escogido para realizar la predicción se realiza la división de la base de datos quedando el subconjunto de entrenamiento con 1935 muestras y el de validación con 247 muestras. La base de entrenamiento como su nombre lo indica es usada para llevar a cabo el entrenamiento de los modelos de predicción y la base de validación es usada para validar los modelos previamente entrenados.

3.4.1 Modelo de predicción con Redes Neuronales

Para realizar el entrenamiento de las redes neuronales que serán usadas para la predicción del índice COLCAP se hace uso del toolbox de MatLab® de Redes Neuronales, se hace uso de funciones de la aplicación de Aproximación de funciones y regresión no lineal. En cuanto a la arquitectura de la red, se decidió contar con una sola capa oculta dado que una red neuronal con una capa oculta es considerada un aproximador universal [50]. Además, dado que no es posible de manera a priori conocer el número de neuronas necesarias para predecir con buen desempeño el índice, se diseñó un algoritmo que variaba el número de neuronas en la capa oculta entre 1 y 50 y se repitió este procedimiento 200 veces de tal manera que se tuviera una base de datos lo suficientemente grande y variada dado que las redes neuronales dependen mucho de la inicialización de los pesos que se realiza de manera aleatoria.

Los parámetros utilizados para el entrenamiento de las redes neuronales se presentan en la Tabla 3.9. En la Figura 3.4 se presenta la estructura general de los modelos creados a partir de redes neuronales. Son 6 entradas dado que el índice COLCAP será predicho por medio de las 6 acciones presentadas en la Tabla 3.8 normalizadas entre 0 y 1, NNCO es el número de neuronas de la capa oculta que será variado entre 1 y 50 para encontrar la arquitectura con mejor desempeño, la función de activación de las neuronas de la capa oculta es Sigmoidea Tangencial Hiperbólica, la función de entrenamiento de la red para actualizar el valor de los pesos y el bias de cada neurona se lleva a cabo de acuerdo al algoritmo de optimización Levenberg-Marquardt, se cuenta con una neurona en la capa

de salida con función de activación limitador duro que representa el valor del índice, el cual se encuentra normalizado entre 0 y 1 y luego se desnormaliza para tenerlo en la escala original. Finalmente, es necesario resaltar que la inicialización de las sinapsis de la red neuronal se hace de manera aleatoria.

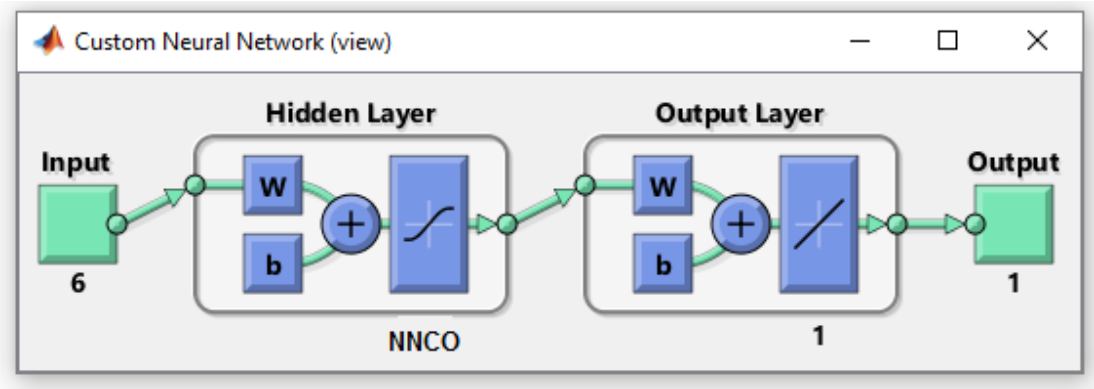


Figura 3.4. Estructura general Red Neuronal para la predicción del índice COLCAP

Tabla 3.9. Parámetros de entrenamiento de las redes neuronales

Parámetro	Valor
Número de Capas Ocultas	1
Número de neuronas de la capa oculta	1-50
Función de activación neuronas capa oculta	Sigmoidea Tangencial Hiperbólica
Función de activación capa de salida	Limitador duro
Función de entrenamiento de la red	Levenberg - Marquardt
Número máximo de Épocas	500
Tiempo máximo de entrenamiento	Infinito
Desempeño deseado	$1 * 10^{-7}$
Gradiente mínimo	$1 * 10^{-7}$
Número de pruebas de validación	6

3.4.2 Modelo de predicción con AdaBoost.RT con φ auto-adaptativo

Para realizar tanto el entrenamiento como la validación del modelo conjunto del algoritmo AdaBoost, se hace uso del algoritmo AdaBoost.RT descrito en la Figura 2.8 con el modelo auto-adaptativo de φ presentado en la sección 2.2.4. MatLab® posee el algoritmo básico de AdaBoost en el Toolbox de Estadística y Aprendizaje de Máquina, pero no posee algoritmos de regresión con AdaBoost, por este motivo, es necesario codificar el algoritmo. La codificación del algoritmo de AdaBoost.RT se llevó a cabo en MatLab®, donde se creó una función llamada *adaboost_reg*, el código de esta se presenta en el Anexo 2. Los parámetros de entrada de esta función son, primero la base de datos de entrenamiento, luego la serie deseada y finalmente el número de Aprendices Débiles deseado en el modelo final, que es el número de iteraciones del algoritmo de AdaBoost que se llevaran a cabo. La salida de esta función son dos, la serie predicha por el modelo final y el modelo final que es un objeto, que cuenta con las características de cada Aprendiz Débil, el peso de cada uno y los

respectivos errores MAPE , definido por (22) y RMSE definido por (6) de cada iteración que son los criterios de desempeño utilizados para escoger el mejor modelo.

$$\text{MAPE} = \sqrt{\frac{100}{\text{m}}\sum_{\text{I}=1}^{\text{m}}\left|\frac{\widehat{\text{y}_\text{I}} - \text{y}_\text{i}}{\widehat{\text{y}_\text{I}}}\right|} \tag{22}$$

Donde,
m es el número de muestras
$\widehat{y_i}$ es el valor esperado
y_i es el valor predicho

En cuanto al aprendiz débil, MatLab® cuenta con múltiples algoritmos que podrían ser considerados aprendices débiles, pero son funciones predeterminadas que no pueden ser modificadas y no cuentan con las características necesarias para tener en cuenta la distribución de pesos de las muestras, lo que hizo necesario, al igual que en el caso de AdaBoost.RT codificar el algoritmo propio para el aprendiz débil. Se codificó el perceptrón, presentado en la Figura 2.3. La función se llama *weak_learner* y su código se presenta en el Anexo 3. Los parámetros de entrada de esta función son cuatro, primero la base de datos de entrenamiento, luego la serie deseada, en seguida la distribución de pesos D_i y finalmente φ, los parámetros de salida son tres, el modelo, error y la serie estimada del mejor aprendiz débil. En la Tabla 3.10, se presentan los parámetros del algoritmo AdaBoost.RT con φ auto-adaptativo utilizados para el entrenamiento del modelo de predicción del índice COLCAP. Es necesario resaltar que el número de aprendices débiles (t) se refiere a la cantidad de aprendices débiles que conforman el modelo conjunto AdaBoost y es diferente al número de iteraciones del aprendiz débil (it) que como su nombre lo indica se refiere al número de iteraciones que realiza el perceptrón para encontrar el mejor regresor que será el aprendiz débil de la iteración t, además, la inicialización de las sinapsis del algoritmo del perceptrón se realiza de manera aleatoria.

Tabla 3.10. Parámetros del algoritmo AdaBoost.RT con φ auto-adaptativo

Parámetro	**Valor**
Algoritmo de Aprendiz Débil	Perceptrón
Número de aprendices débiles (t)	1-50
Número iteraciones aprendiz débil (it)	100
Criterios de evaluación	$MAPE, RMSE$
$\varphi_{inicial}$	0.1-0.5
r	0.3, 0.5, 0.7
$\beta = error^n$	$n = 1, 2, 3$

3.5 RESULTADOS PREDICCIÓN DEL ÍNDICE COLCAP

3.5.1 Resultados obtenidos con Redes Neuronales

Se realizaron 200 pruebas variando la cantidad de neuronas desde 1 hasta 50, dado que no es posible conocer de manera a priori la cantidad de neuronas que darán el mejor resultado. Primero se

entrenaron las redes neuronales con la base de datos de entrenamiento y luego se utilizaron estas redes para predecir el índice con la base de datos de validación, hallando el error medio absoluto (MAPE) para usarlo como criterio de evaluación.

En la Figura 3.5 se presenta el error medio absoluto de las redes neuronales entrenadas y validadas. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada número de neuronas. Se puede observar que el error máximo no presenta relación con el número de neuronas de capa oculta que conforman la red, además del error promedio de las 200 pruebas de la red se puede observar que a partir de las 12 neuronas de capa oculta presenta una tendencia en aumento. En cuanto al error mínimo, no se puede diferenciar correctamente los valores debido a la escala, y dado que son los más importantes, puesto que indican cual fue la mejor red neuronal en cada caso, se presentan de manera separada en la Figura 3.6.

En la Figura 3.6 se presenta el error medio absoluto MAPE de las mejores redes neuronales para cada número de neuronas, las cuales son aquellas que presentan el mínimo error de las 200 pruebas realizadas. Se puede observar como el error en el caso de una neurona en la capa oculta es el mayor de todas las pruebas con un valor de 10.01%, luego el error va disminuyendo a medida que aumenta el número de neuronas en la capa oculta hasta alcanzar el menor error obtenido que es de 1.206% con 9 neuronas; a partir de ahí, empieza a aumentar el error a medida que aumenta el número de neuronas hasta llegar a las 50 neuronas, cuya red presenta un error del 3.21% que es menor al de la red neuronal con una sola neurona en la capa oculta. Se toma la red con 9 neuronas en la capa oculta como la mejor red neuronal de predicción para el índice COLCAP, debido a que presenta el menor error de validación de todas las redes.

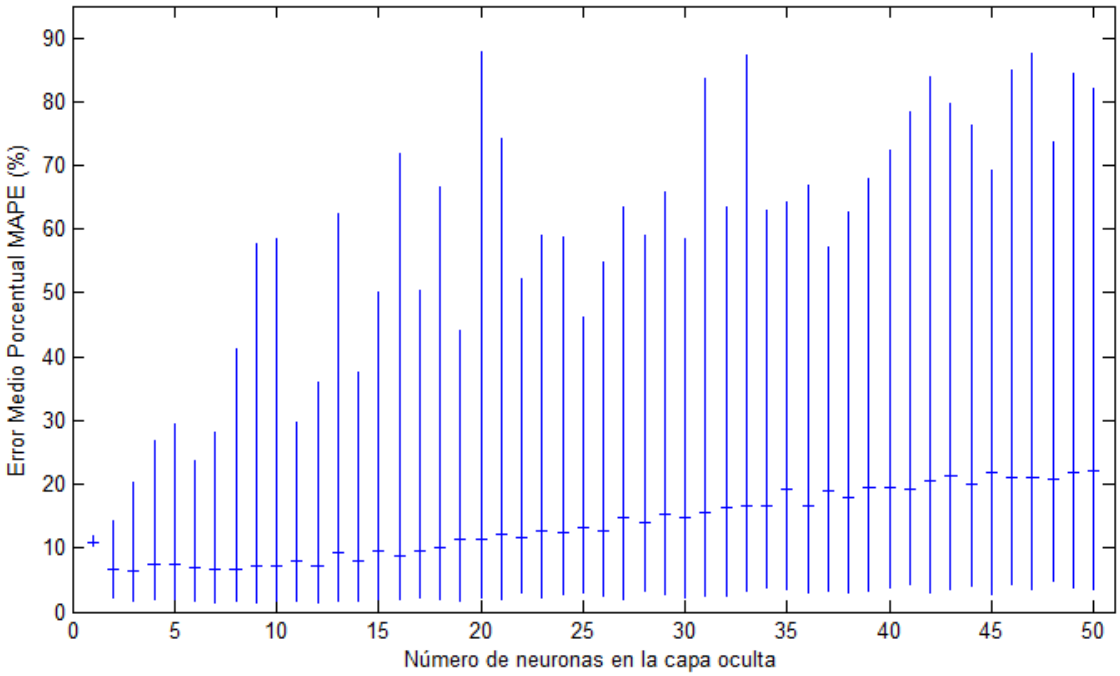


Figura 3.5. Error medio absoluto de validación de las redes neuronales

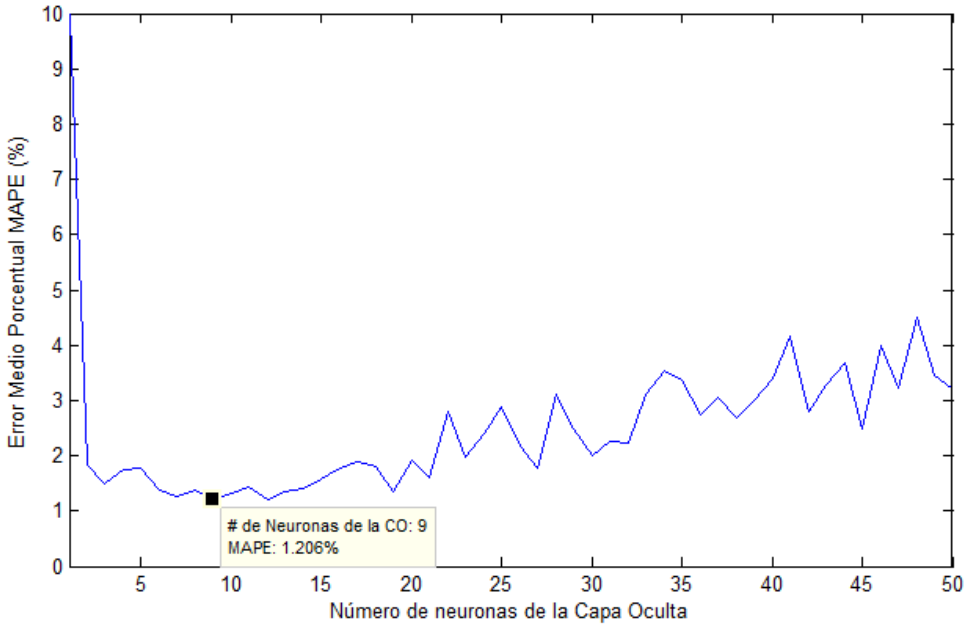


Figura 3.6. Error Medio Absoluto de las mejores redes

En la Figura 3.7 se presenta una comparación entre la serie original COLCAP y la predicción de la serie realizada por medio de la mejor red neuronal. El error medio absoluto (MAPE) entre estas series es de 1.206%, además se puede observar que la predicción hasta el mes de Mayo presenta error en seguir la serie, mientras a partir del mes de Mayo la predicción tiene un comportamiento más parecido al de la serie original.

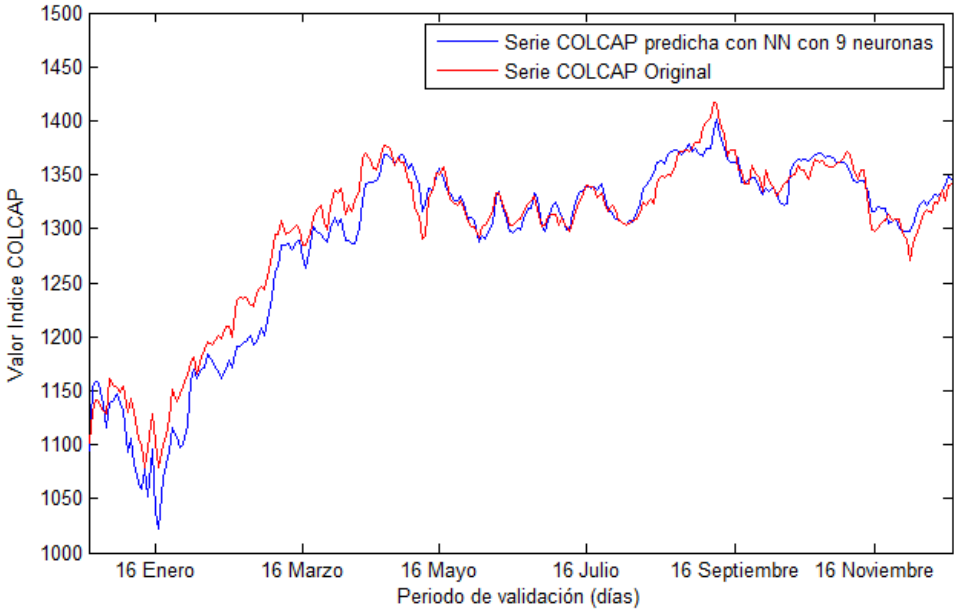


Figura 3.7. Comparación serie original y predicción realizada con la mejor red neuronal

3.5.2 Resultados obtenidos con AdaBoost.RT con φ auto-adaptativo

Para realizar la predicción del índice COLCAP por medio de AdaBoost se hizo uso de la función presentada en el Anexo 2, usando como Weak Learner el perceptrón con la función descrita en el Anexo 3. Dentro del algoritmo de AdaBoost.RT con φ auto-adaptativo están los parámetros β_t, que es la actualización del peso de las muestras, $\varphi_{inicial}$ que hace referencia al delimitador de muestras correctas y r que tiene efecto en la tasa de cambio del error RMSE, por este motivo, se realiza la variación de estos parámetros de manera separada para identificar el valor adecuado de cada uno de estos para el problema que se desea resolver.

3.5.2.1 Variación de la actualización del peso de las muestras correctas

Como se explicó en la sección 2.2.3, la actualización de la distribución de muestras D_i depende del factor β_t, calculado ε_t^n donde $n = 1, 2\ o\ 3$ (lineal, cuadrada o cúbica), donde ε es el error del aprendiz débil de la iteración t. De este factor no sólo depende la actualización de la distribución de muestras, también depende el peso del aprendiz débil en la iteración t, como se observa en (5). En la literatura no se define el método más efectivo de actualización, si lineal, cuadrada o cúbica, por este motivo se realizaron 200 pruebas para cada uno de los casos dejando constante el valor inicial de $\varphi = 0.2$ y $r = 0.5$, según las conclusiones presentadas en [38]. Al igual que en el caso del entrenamiento de redes neuronales, no es posible conocer de manera a priori el número de aprendices débiles que darán el mejor resultado, por este motivo, las pruebas fueron realizadas variando la cantidad de aprendices débiles desde 1 hasta 50.

- $\beta = error$

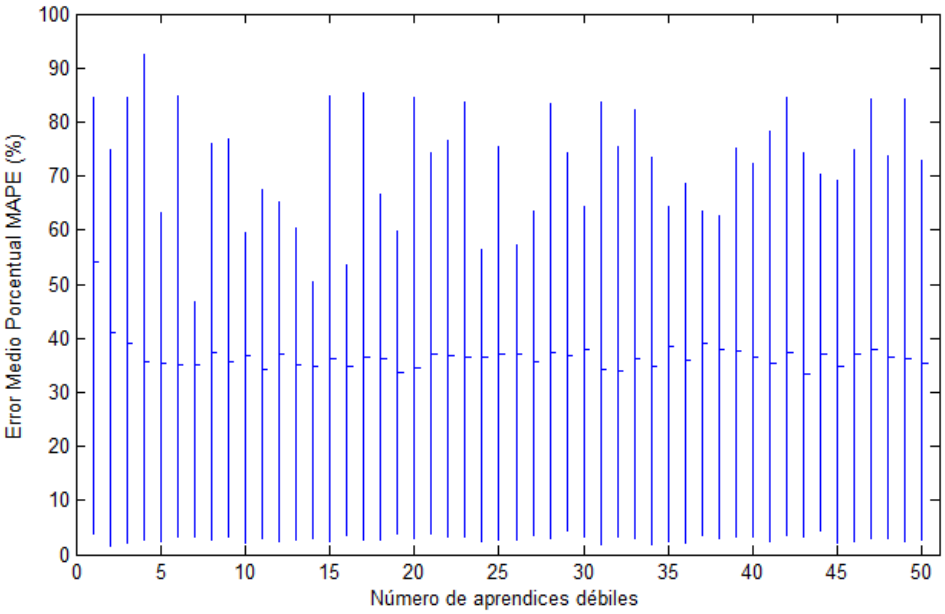


Figura 3.8. Error medio absoluto de validación de AdaBoost.RT con $\beta = error$

En la Figura 3.8 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\beta = error$. Se presentan 50 líneas verticales donde el valor inicial representa el error

mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se muestra una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.8 no Es posible analizar el comportamiento de este error, por este motivo se presentan los errores mínimos de cada variación de aprendices débiles en la Figura 3.9.

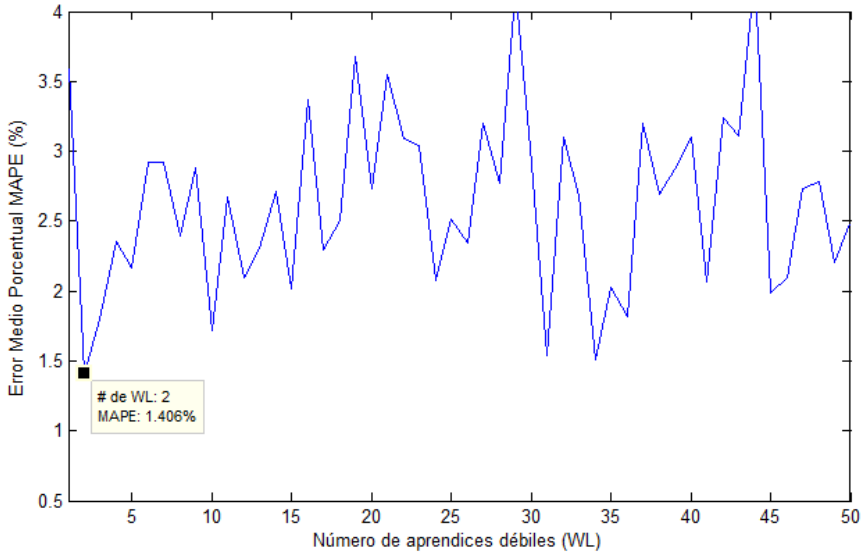


Figura 3.9. Error Medio Porcentual de los mejores modelos conjuntos con $\beta = error$

En la Figura 3.9 se presenta el error medio porcentual mínimo de los mejores modelos conjuntos variando el número de aprendices débiles. El menor error es de 1.406% y se presenta con el modelo de 2 aprendices débiles. No se observa una relación directa entre el error mínimo de las pruebas y el aumento de la cantidad de aprendices débiles.

- $\beta = error^2$

En la Figura 3.10 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\beta = error^2$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se muestra una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.10 no es posible analizar el comportamiento de este error, por este motivo se presentan los errores mínimos de cada variación de aprendices débiles en la Figura 3.11.

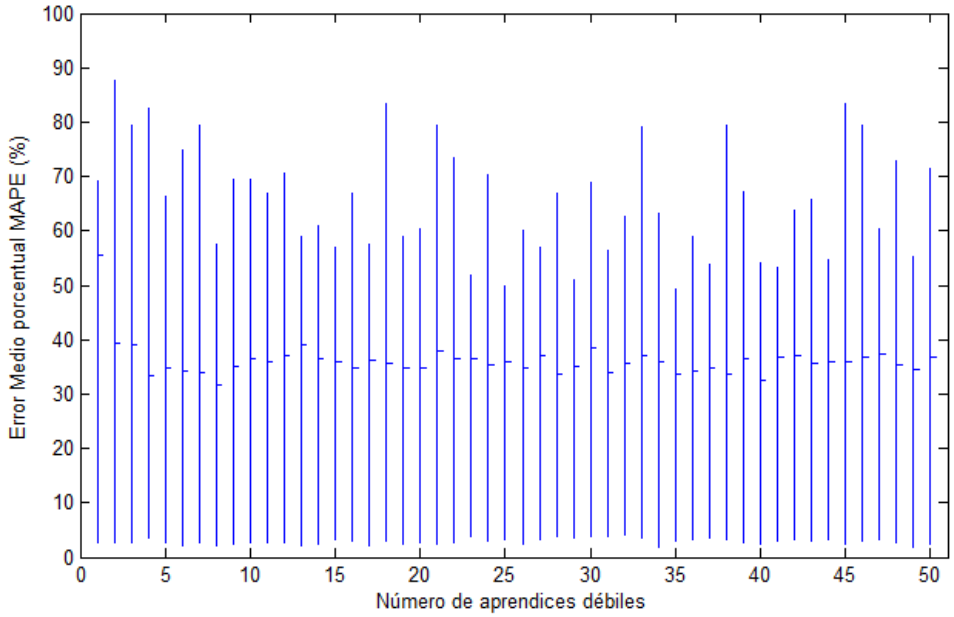


Figura 3.10. Error medio absoluto de validación de AdaBoost.RT con $\beta = error^2$

En la Figura 3.11 se presenta el error medio porcentual mínimo de los mejores modelos conjuntos variando el número de aprendices débiles con $\beta = error^2$. El menor error es de 1.522% y se presenta con el modelo de 34 aprendices débiles. No se observa una relación directa entre el error mínimo de las pruebas y el aumento de la cantidad de aprendices débiles.

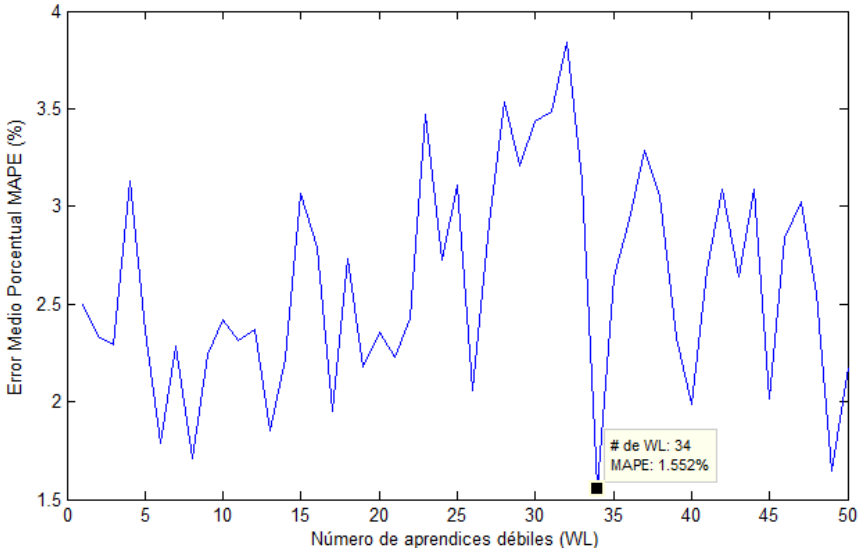


Figura 3.11. Error Medio Porcentual de los mejores modelos conjuntos con $\beta = error^2$

- $\beta = error^3$

En la Figura 3.12 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\beta = error^3$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada

número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se muestra una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.12 no es posible analizar el comportamiento de este error, por este motivo se presentan los errores mínimos de cada variación de aprendices débiles en la Figura 3.13.

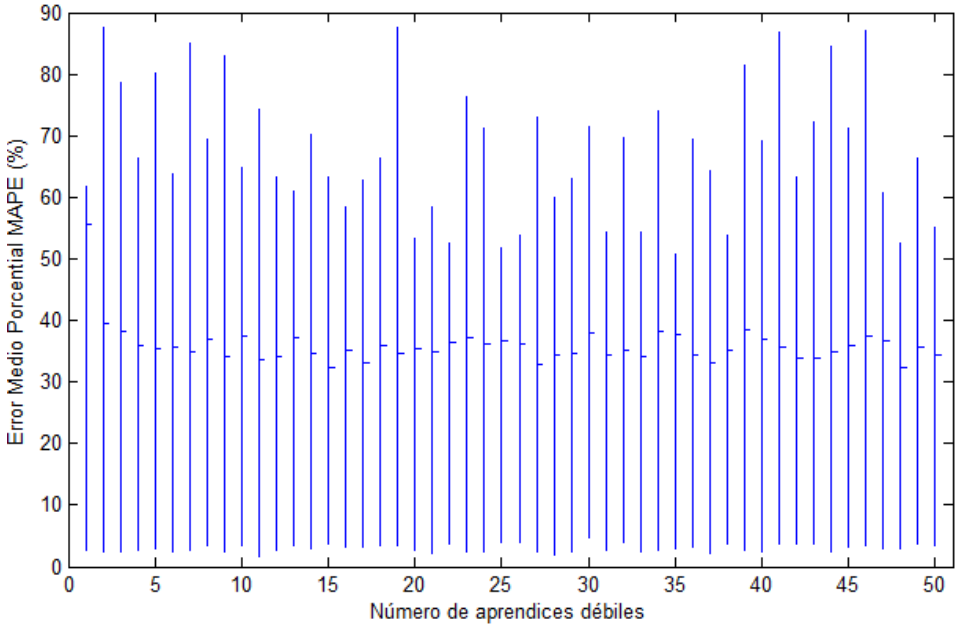


Figura 3.12. Error medio absoluto de validación de AdaBoost.RT con $\beta = error^3$

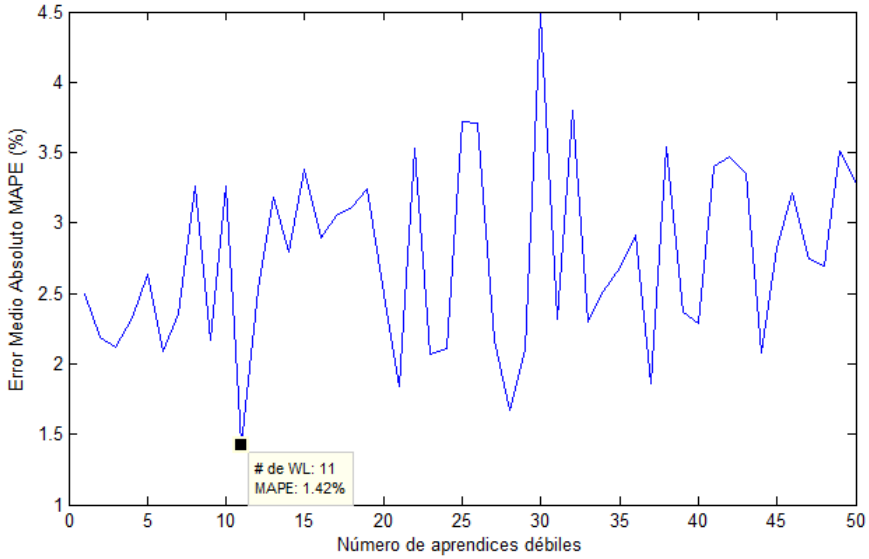


Figura 3.13. Error Medio Porcentual de los mejores modelos con $\beta = error^3$

En la Figura 3.13 se presenta el error medio porcentual mínimo de los mejores modelos conjuntos variando el número de aprendices débiles con $\beta = error^3$. El menor error es de 1.42% y se presenta

con el modelo de 11 aprendices débiles. No se observa una relación directa entre el error mínimo de las pruebas y el aumento de la cantidad de aprendices débiles.

En la Figura 3.14 se observa el error medio absoluto de los mejores modelos conjuntos obtenidos en las 200 pruebas realizadas variando el número de aprendices débiles con $\beta = error^n$, donde $n = 1,2,3$. A partir de la Figura 3.14 se puede observar que en general la variación del cálculo de β, no tiene una influencia marcada en el error medio de los modelos conjuntos variando el número de aprendices débiles. De igual manera no se observa una relación entre el error de los mejores modelos y el número de aprendices débiles.

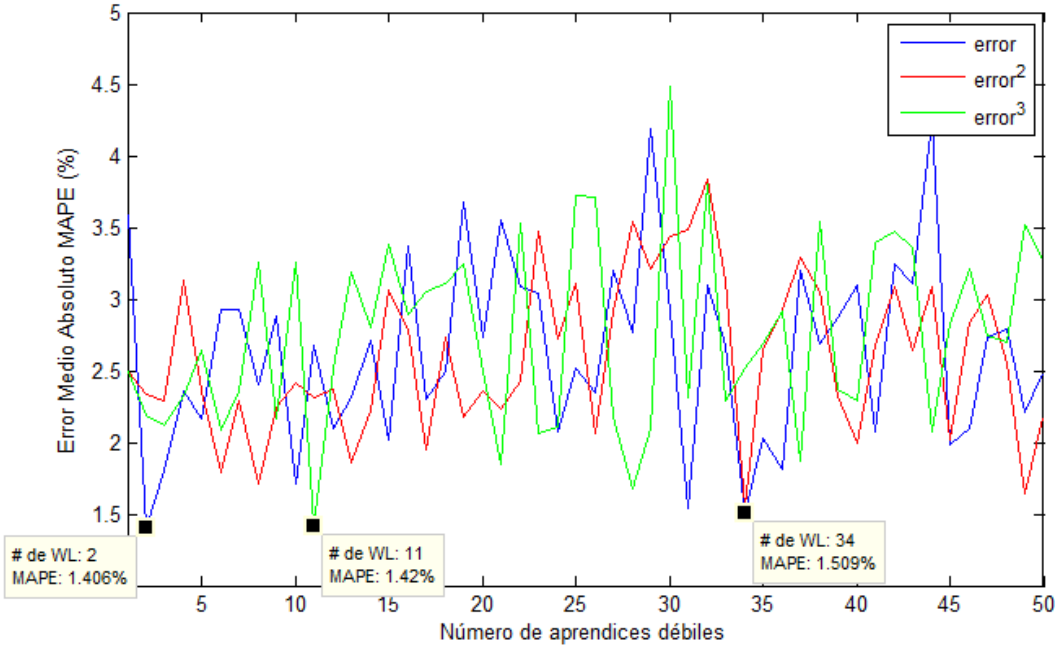


Figura 3.14. Error Medio Absoluto de AdaBoost.RT con variación de la actualización de pesos β_t

En la Tabla 3.11 se presenta el resumen de los mejores resultados de los modelos conjuntos variando el método de actualización de $\beta = error^n$, donde $n = 1,2,3$. El mejor modelo conjunto generado se presentó con $\beta = error$ y dos aprendices débiles, por este motivo se selecciona $\beta = error, n = 1$ como el método óptimo de actualización de β para la predicción del índice COLCAP.

Parámetro	# de Aprendices Débiles	Error
$\beta = error$	2	1.406%
$\beta = error^2$	34	1.509%
$\beta = error^3$	11	1.42%

Tabla 3.11.Resumen mejores modelos conjuntos variando $\beta = error^n$

3.5.2.2 Variación del valor inicial de φ

La mayor ventaja del algoritmo de AdaBoost utilizado en este proyecto, se basa en la adaptabilidad de φ dependiendo del error RMSE de cada iteración t, pero no se considera el valor inicial de este parámetro aunque en [38] se aconseja que se inicialice con un valor entre 0.2 y 0.4,

dado que con una inicialización mayor a 0.4 el algoritmo se vuelve inestable. Por este motivo, se considera variar el valor inicial de φ para encontrar el óptimo para el problema de regresión del índice COLCAP.

Para llevar a cabo esto, se realizaron 200 pruebas variando el valor inicial de φ, desde 0.1 hasta 0.5, variando la cantidad de aprendices débiles desde 1 hasta 50, donde se toma el mejor modelo conjunto de cada una de las variaciones y se escoge el valor óptimo de φ para el problema que se desea resolver.

- $\varphi_{inicial} = 0.1$

En la Figura 3.15 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\varphi_{\text{inicial}} = 0.1$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se presenta una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.15 no es posible analizar el comportamiento de este, motivo por el cual los errores mínimos de la variación de aprendices débiles con $\varphi_{\text{inicial}} = 0.1$ se presenta en la Figura 3.16.

En la Figura 3.16 se presenta el error medio porcentual de los mejores modelos conjuntos generados variando el número de aprendices débiles con $\varphi_{\text{inicial}} = 0.1$. No se observa una relación directa entre el error mínimo de las pruebas y la variación de la cantidad de aprendices débiles. El mejor modelo conjunto tiene un error de 1.604% y tiene 37 aprendices débiles.

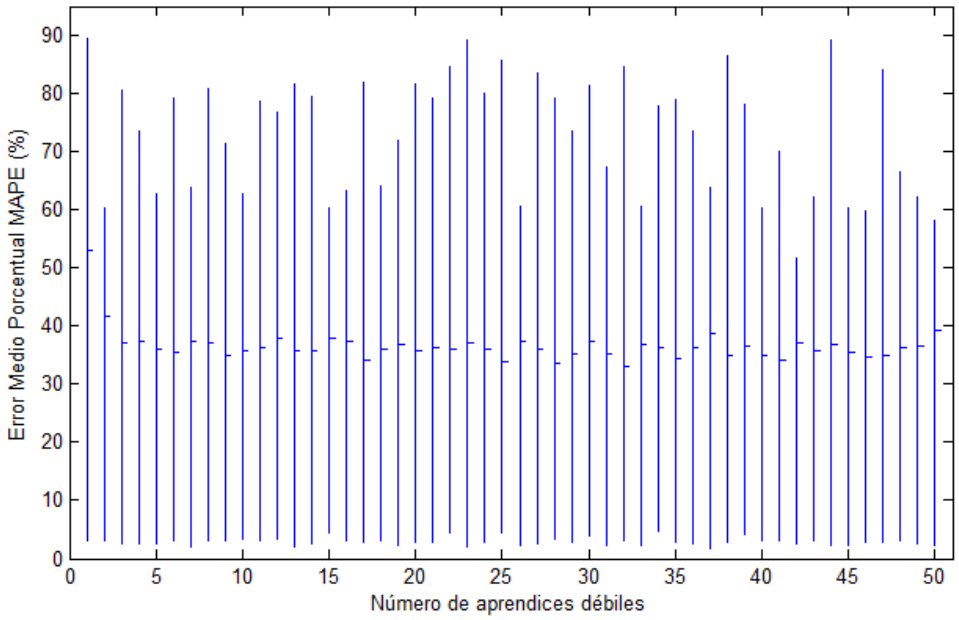


Figura 3.15. Error medio absoluto de validación de AdaBoost.RT con $\varphi_{inicial} = 0.1$

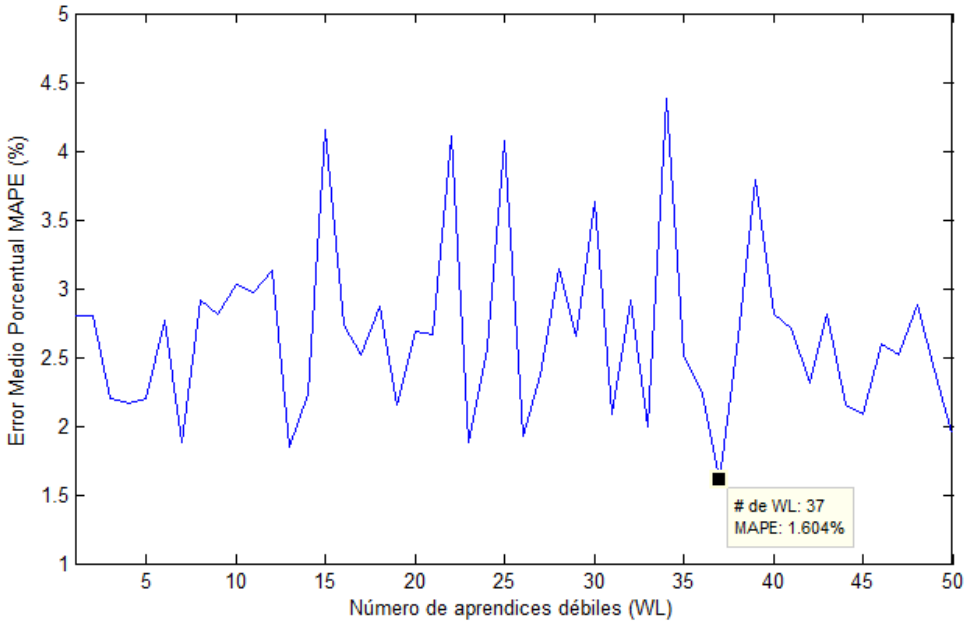


Figura 3.16. Error Medio Porcentual de los mejores modelos conjuntos con $\varphi_{inicial} = 0.1$

- $\varphi_{\text{inicial}} = 0.2$

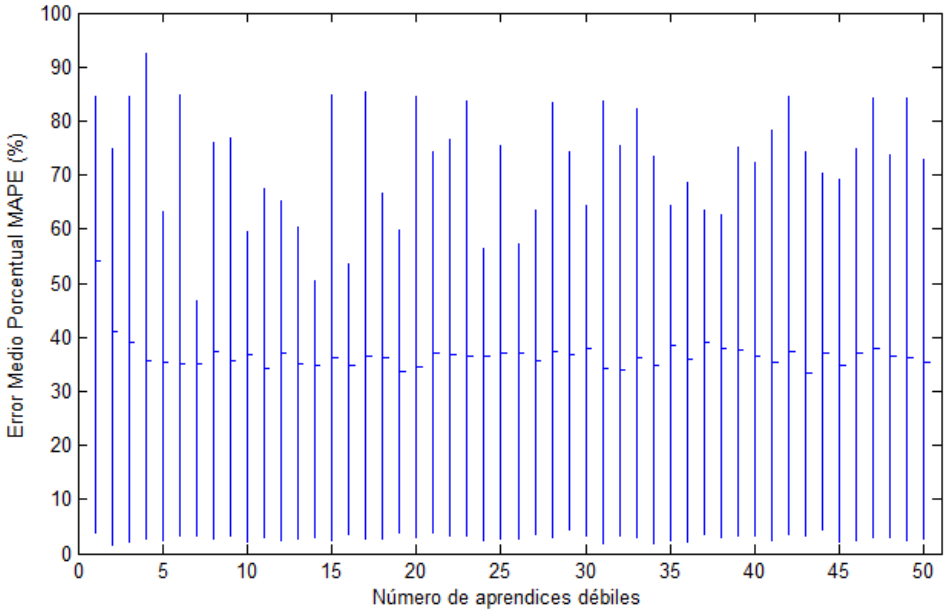


Figura 3.17. Error medio absoluto de validación con $\varphi_{inicial} = 0.2$

En la Figura 3.17 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\varphi_{\text{inicial}} = 0.2$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se muestra una relación clara entre el error

promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.17 no es posible analizar el comportamiento de este error, por este motivo se presentan los errores mínimos de cada variación de aprendices débiles en la Figura 3.18.

En la Figura 3.18 se presenta el error medio porcentual mínimo de los mejores modelos conjuntos variando el número de aprendices débiles. El menor error es de 1.406% y se presenta con el modelo de 2 aprendices débiles. No se observa una relación directa entre el error mínimo de las pruebas y el aumento de la cantidad de aprendices débiles.

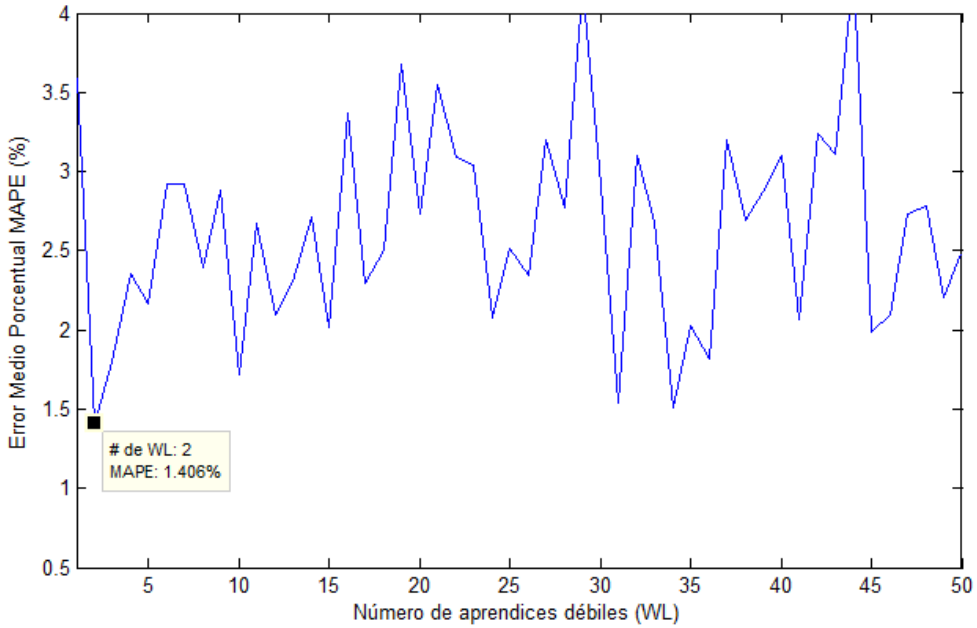


Figura 3.18. Error Medio Porcentual de los mejores modelos conjuntos con $\varphi_{inicial} = 0.2$

- $\varphi_{inicial} = 0.3$

En la Figura 3.19 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\varphi_{inicial} = 0.3$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se presenta una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.19 no es posible analizar el comportamiento de este, motivo por el cual los errores mínimos de la variación de aprendices débiles con $\varphi_{inicial} = 0.3$ se presenta en la Figura 3.20.

En la Figura 3.20 se presenta el error medio porcentual de los mejores modelos conjuntos generados variando el número de aprendices débiles con $\varphi_{\text{inicial}} = 0.3$. No se observa una relación directa entre el error mínimo de las pruebas y la variación de la cantidad de aprendices débiles. El mejor modelo conjunto tiene un error de 1.575% y tiene 26 aprendices débiles

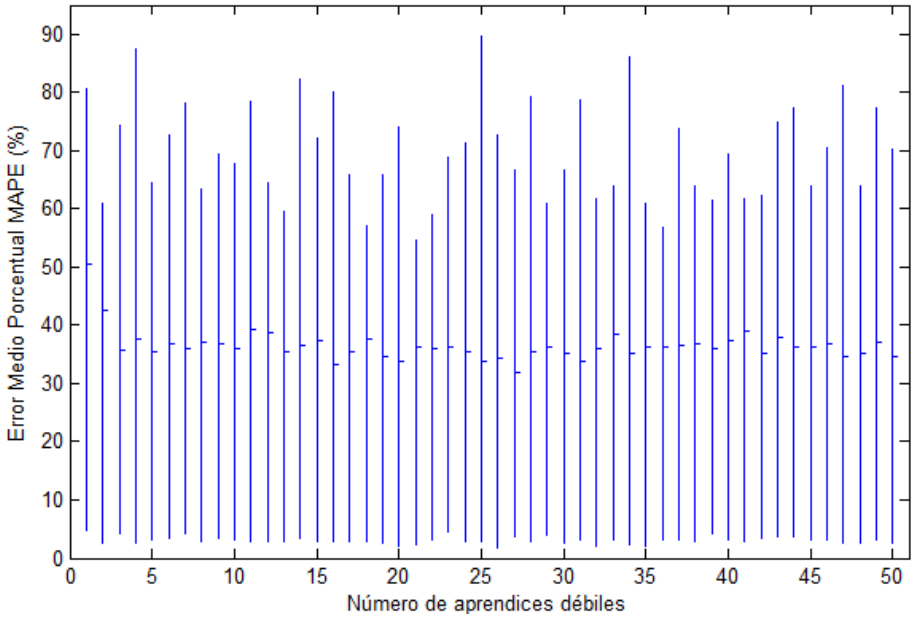


Figura 3.19. Error medio absoluto de validación de AdaBoost.RT con $\varphi_{inicial} = 0.3$

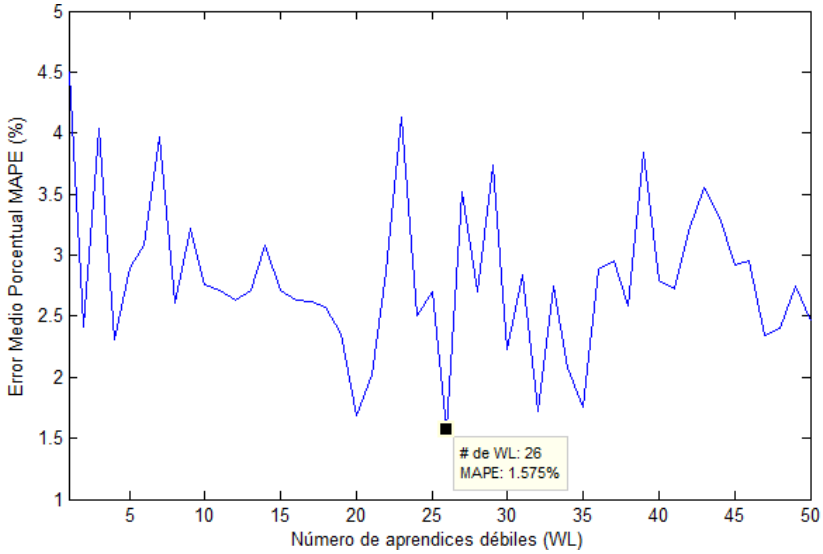


Figura 3.20. Error Medio Porcentual de los mejores modelos conjuntos con $\varphi_{inicial} = 0.3$

- $\varphi_{\text{inicial}} = 0.4$

En la Figura 3.21 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\varphi_{\text{inicial}} = 0.4$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se presenta una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir

de la Figura 3.21 no es posible analizar el comportamiento de este, motivo por el cual los errores mínimos de la variación de aprendices débiles con $\varphi_{\text{inicial}} = 0.4$ se presenta en la Figura 3.22.

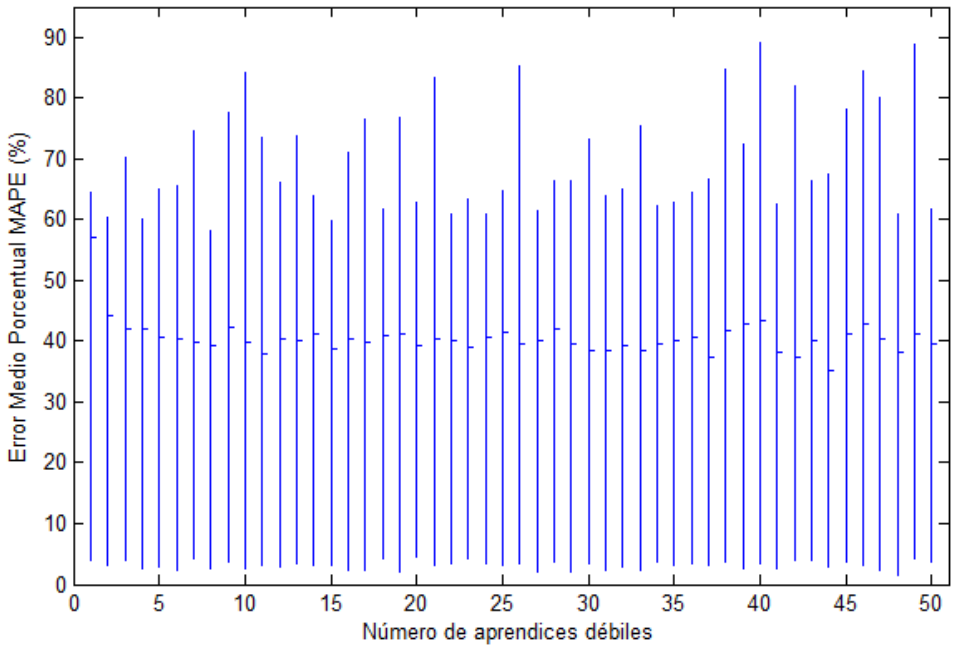


Figura 3.21. Error medio absoluto de validación de AdaBoost.RT con $\varphi_{inicial} = 0.4$

En la Figura 3.22 se presenta el error medio porcentual de los mejores modelos conjuntos generados variando el número de aprendices débiles con $\varphi_{\text{inicial}} = 0.4$. No se observa una relación directa entre el error mínimo de las pruebas y la variación de la cantidad de aprendices débiles. El mejor modelo conjunto tiene un error de 1.247% y tiene 48 aprendices débiles.

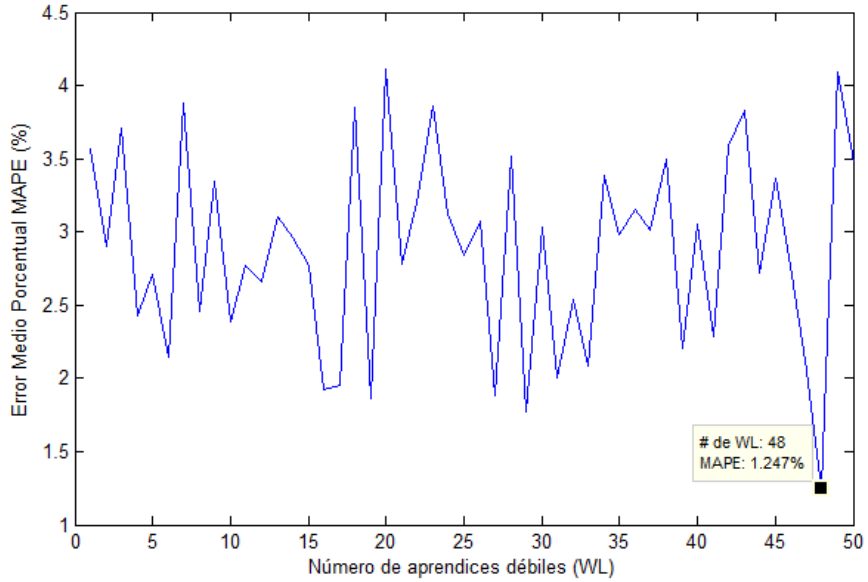


Figura 3.22. Error Medio Porcentual de los mejores modelos conjuntos con $\varphi_{inicial} = 0.4$

- $\varphi_{\text{inicial}} = 0.5$

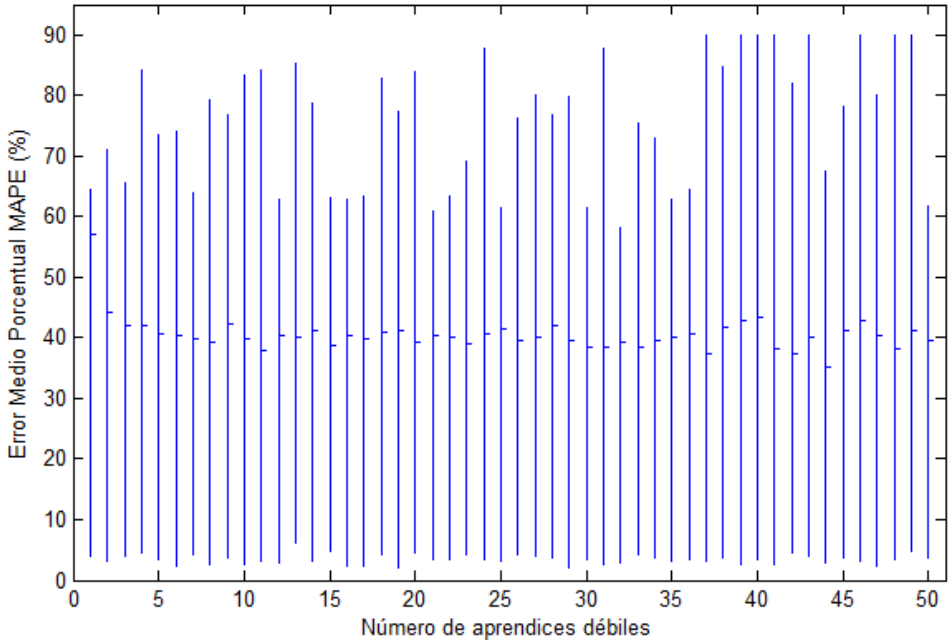


Figura 3.23. Error medio absoluto de validación de AdaBoost.RT con $\varphi_{inicial} = 0.5$

En la Figura 3.23 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\varphi_{\text{inicial}} = 0.5$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se presenta una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.23 no es posible analizar el comportamiento de este, motivo por el cual los errores mínimos de la variación de aprendices débiles con $\varphi_{\text{inicial}} = 0.5$ se presenta en la Figura 3.24.

En la Figura 3.24 se presenta el error medio porcentual de los mejores modelos conjuntos generados variando el número de aprendices débiles con $\varphi_{\text{inicial}} = 0.5$. No se observa una relación directa entre el error mínimo de las pruebas y la variación de la cantidad de aprendices débiles. El mejor modelo conjunto tiene un error de 1.775% y 29 aprendices débiles.

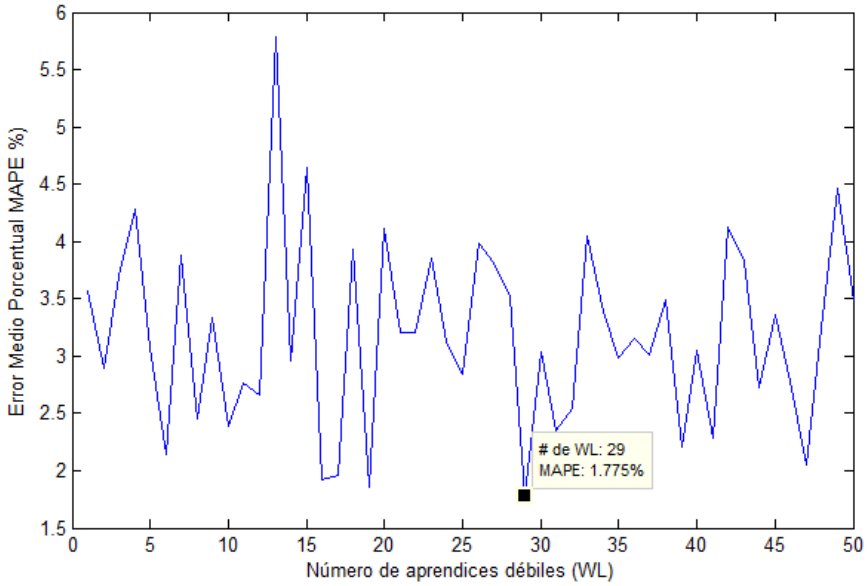


Figura 3.24. Error Medio Porcentual de los mejores modelos conjuntos con $\varphi_{inicial} = 0.5$

En la Figura 3.25 se presenta el error medio absoluto de los mejores modelos conjuntos obtenidos en las 200 pruebas realizadas variando el número de aprendices débiles y el valor de φ_{inicial}, desde 0.1 hasta 0.5.

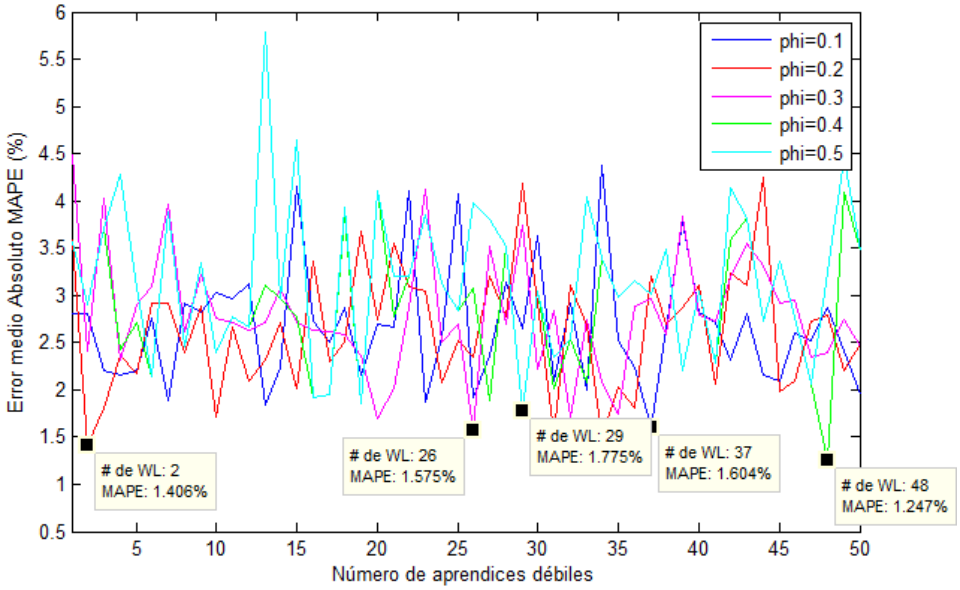


Figura 3.25. Error Medio Absoluto de AdaBoost.RT con variación de $\varphi_{inicial}$

A partir de la Figura 3.25 se puede observar que en general la variación del valor inicial de φ, no tiene una influencia marcada en el error medio de los modelos conjuntos variando el número de aprendices débiles. En la Tabla 3.12 se presenta el resumen de los mejores resultado de los modelos conjuntos variando el método de actualización de φ_{inicial}. El mejor modelo conjunto generado se presentó con $\varphi_{\text{inicial}} = 0.4$ y 48 aprendices débiles, por este motivo se selecciona $\varphi_{\text{inicial}} = 0.4$ como el valor óptimo de φ_{inicial} para la predicción del índice COLCAP.

Parámetro	# de Aprendices Débiles	Error
$\varphi_{\text{inicial}} = 0.1$	37	1.604%
$\varphi_{\text{inicial}} = 0.2$	2	1.406%
$\varphi_{\text{inicial}} = 0.3$	26	1.575%
$\varphi_{\text{inicial}} = 0.4$	48	1.247%
$\varphi_{\text{inicial}} = 0.5$	29	1.775%

Tabla 3.12. Resumen mejores modelos conjuntos variando $\varphi_{inicial}$

3.5.2.3 Variación del parámetro r

El último parámetro que es posible variar y que depende del problema es r, un factor de ajuste de la tasa de cambio del RMSE usado para la actualización de pesos de las muestras en el algoritmo AdaBoost.RT que se presenta en la (8). En [38], se sugiere $r = 0.5$ pero de igual manera se indica que puede ser seleccionado dependiendo de la aplicación.

Para este problema se decide realizar pruebas con $r = 0.3, 0.5$ y 0.7 para ver el efecto de este en el algoritmo, para cada caso se varió la cantidad de aprendices débiles de 1 a 50 y se realizaron 200 pruebas donde se toma el mejor modelo conjunto de cada una de las variaciones y se escoge el valor óptimo de r para el problema a resolver.

- $r = 0.3$

En la Figura 3.26 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $\mathrm{r} = 0.3$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales, representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se presenta una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.26 no es posible analizar el comportamiento de este, motivo por el cual los errores mínimos de la variación de aprendices débiles con $\mathrm{r} = 0.3$ se presenta en la Figura 3.27.

En la Figura 3.27 se presenta el error medio porcentual de los mejores modelos conjuntos generados variando el número de aprendices débiles con $\mathrm{r} = 0.3$. No se observa una relación directa entre el error mínimo de las pruebas y la variación de la cantidad de aprendices débiles. El mejor modelo conjunto tiene un error de 1.353% y 14 aprendices débiles.

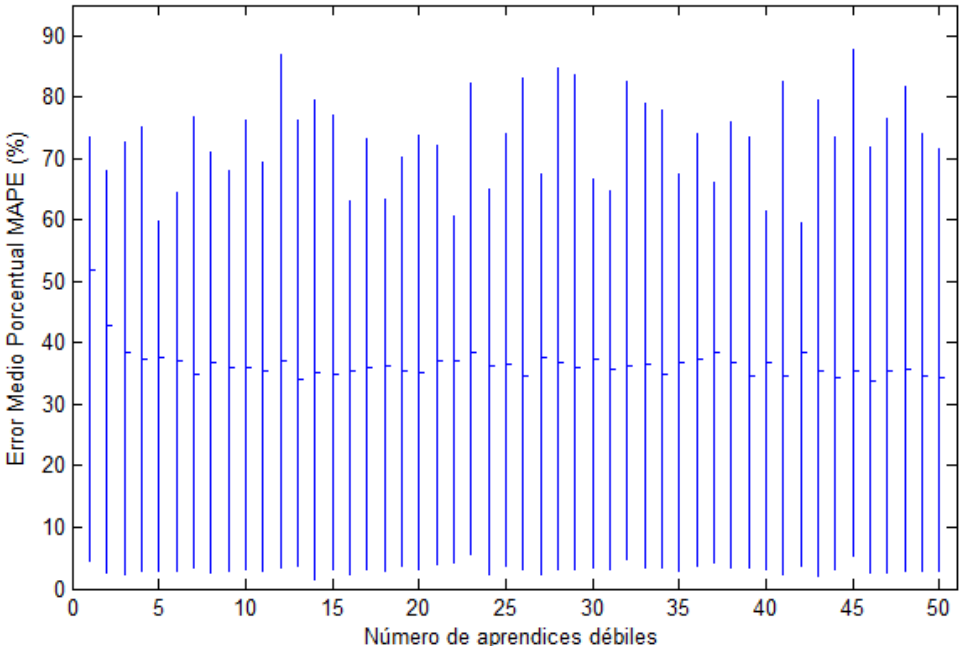


Figura 3.26. Error medio absoluto de validación de AdaBoost.RT con $r = 0.3$

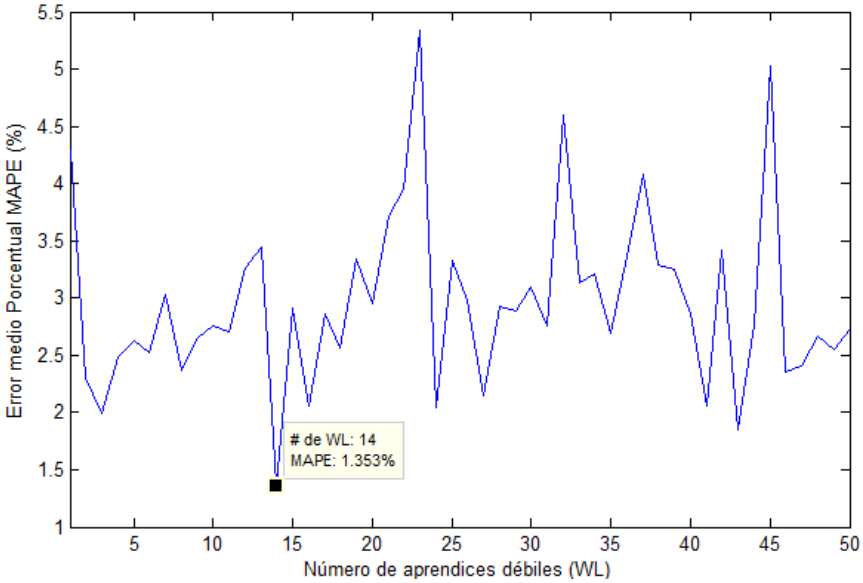


Figura 3.27. Error Medio Porcentual de los mejores modelos conjuntos con $r = 0.3$

- $r = 0.5$

En la Figura 3.28 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $r = 0.5$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se presenta una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura

3.28 no es posible analizar el comportamiento de este, motivo por el cual los errores mínimos de la variación de aprendices débiles con r = 0.5 se presenta en la Figura 3.29.

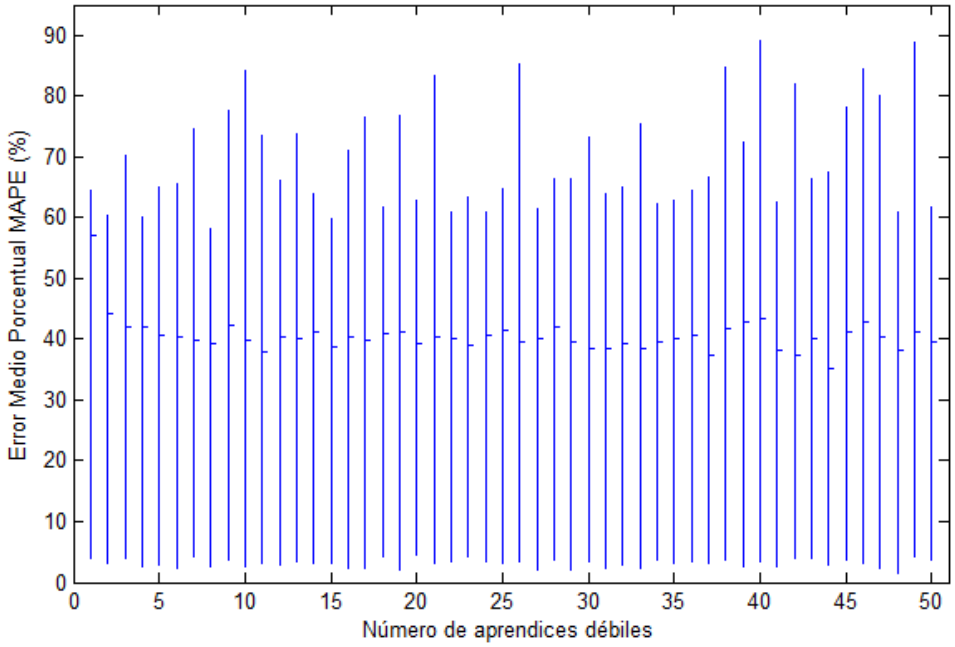


Figura 3.28. Error medio absoluto de validación de AdaBoost.RT con $r = 0.5$

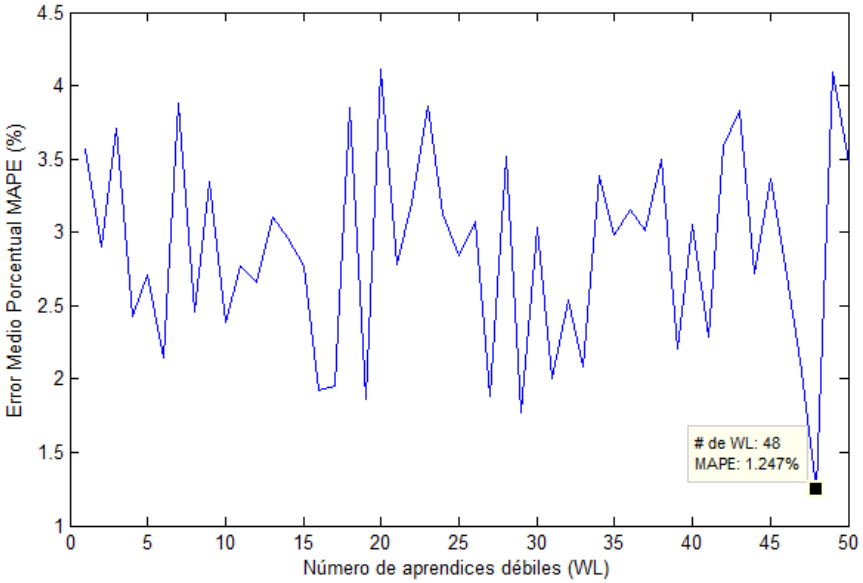


Figura 3.29. Error Medio Porcentual de los mejores modelos conjuntos con $r = 0.5$

En la Figura 3.29 se presenta el error medio porcentual de los mejores modelos conjuntos generados variando el número de aprendices débiles con r = 0.5. No se observa una relación directa entre el error mínimo de las pruebas y la variación de la cantidad de aprendices débiles. El mejor modelo conjunto tiene un error de 1.247% y tiene 48 aprendices débiles.

- $r = 0.7$

En la Figura 3.30 se presenta el error medio absoluto de los modelos conjuntos de AdaBoost entrenados y validados con $r = 0.7$. Se presentan 50 líneas verticales donde el valor inicial representa el error mínimo, el valor final representa el error máximo y la línea horizontal que cruza cada una de las líneas verticales representa el error promedio de las 200 pruebas realizadas para cada número de aprendices débiles. Se puede observar, que el comportamiento del error máximo no tiene patrones marcados que se puedan relacionar con el aumento de los aprendices débiles, lo mismo sucede con el error promedio de las pruebas, dado que no se presenta una relación clara entre el error promedio y el número de aprendices débiles del modelo conjunto. En cuanto al error mínimo, a partir de la Figura 3.30 no es posible analizar el comportamiento de este, motivo por el cual los errores mínimos de la variación de aprendices débiles con $r = 0.7$ se presenta en la Figura 3.31.

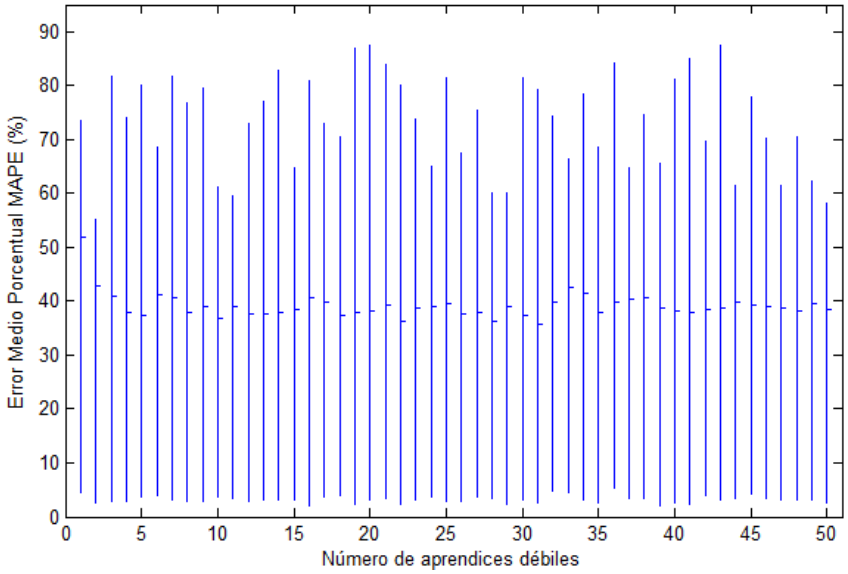


Figura 3.30. Error medio absoluto de validación de AdaBoost.RT con $r = 0.7$

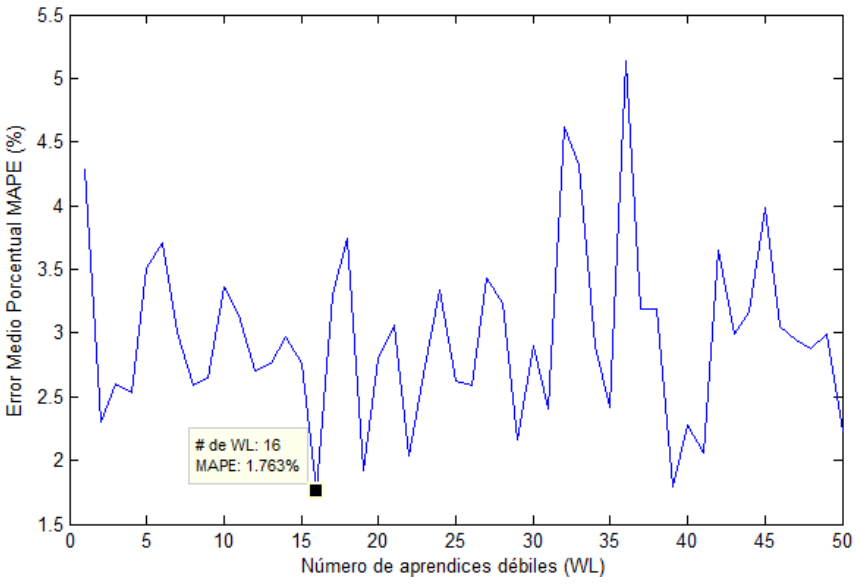


Figura 3.31. Error Medio Porcentual de los mejores modelos conjuntos con $r = 0.7$

En la Figura 3.31 se presenta el error medio porcentual de los mejores modelos conjuntos generados variando el número de aprendices débiles con r = 0.7. No se observa una relación directa entre el error mínimo de las pruebas y la variación de la cantidad de aprendices débiles. El mejor modelo conjunto tiene un error de 1.763% y tiene 16 aprendices débiles

En la Figura 3.32 se presenta el error medio absoluto de los mejores modelos conjunto obtenidos en las 200 pruebas realizadas variando el número de aprendices débiles y el valor de r = 0.3, 0.5 y 0.7.

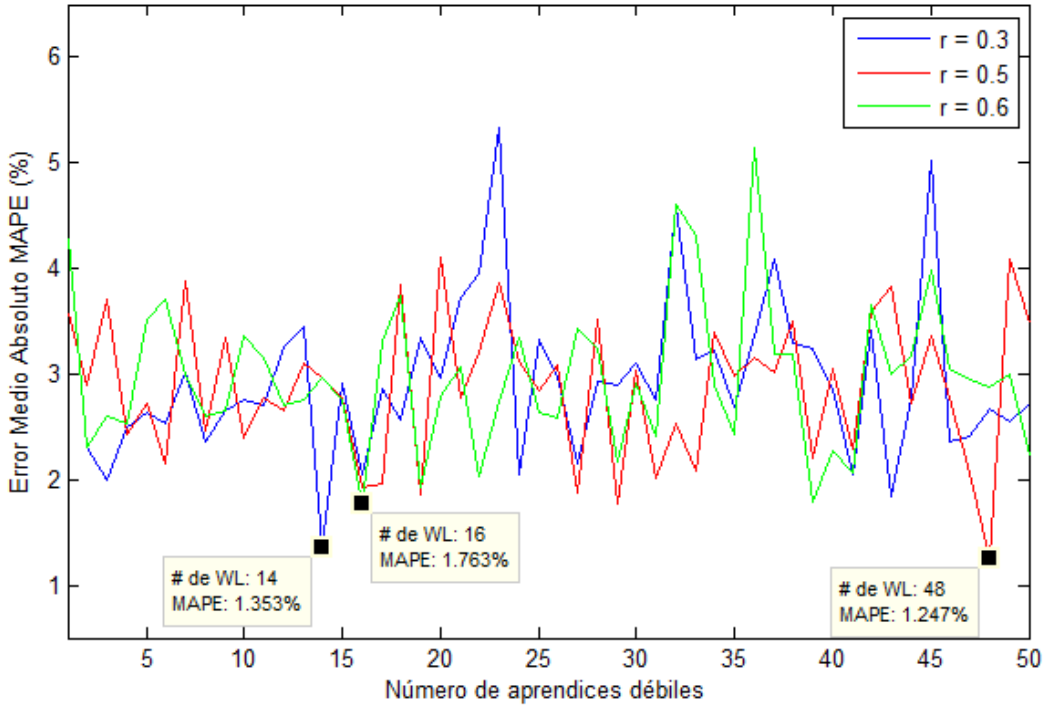


Figura 3.32. Error Medio Absoluto de AdaBoost.RT con variación de r

A partir de la Figura 3.32 se puede observar que en general la variación de r, no tiene una influencia marcada en el error medio de los modelos conjuntos variando el número de aprendices débiles. En la Tabla 3.13 se presenta el resumen de los mejores resultado de los modelos conjuntos variando el método de actualización de r. El mejor modelo conjunto generado se presentó con r = 0.5 y 48 aprendices débiles, por este motivo se selecciona r= 0.5 como el valor óptimo de r para la predicción del índice COLCAP.

Parámetro	# de Aprendices Débiles	Error
r = 0.3	14	1.353%
r = 0.5	48	1.247%
r = 0.7	16	1.763%

Tabla 3.13. Resumen mejores modelos conjuntos variando *r*

3.5.2.4 Modelo óptimo de predicción del índice COLCAP con AdaBoost.RT con φ auto − adaptativo

Luego de analizar los resultados de las variaciones de los parámetros $\beta, \varphi_{\text{inicial}}$ y r, se determinó que el mejor modelo conjunto se obtuvo con $\beta = \text{error}, \varphi_{\text{inicial}} = 0.4$ y $r = 0.5$.

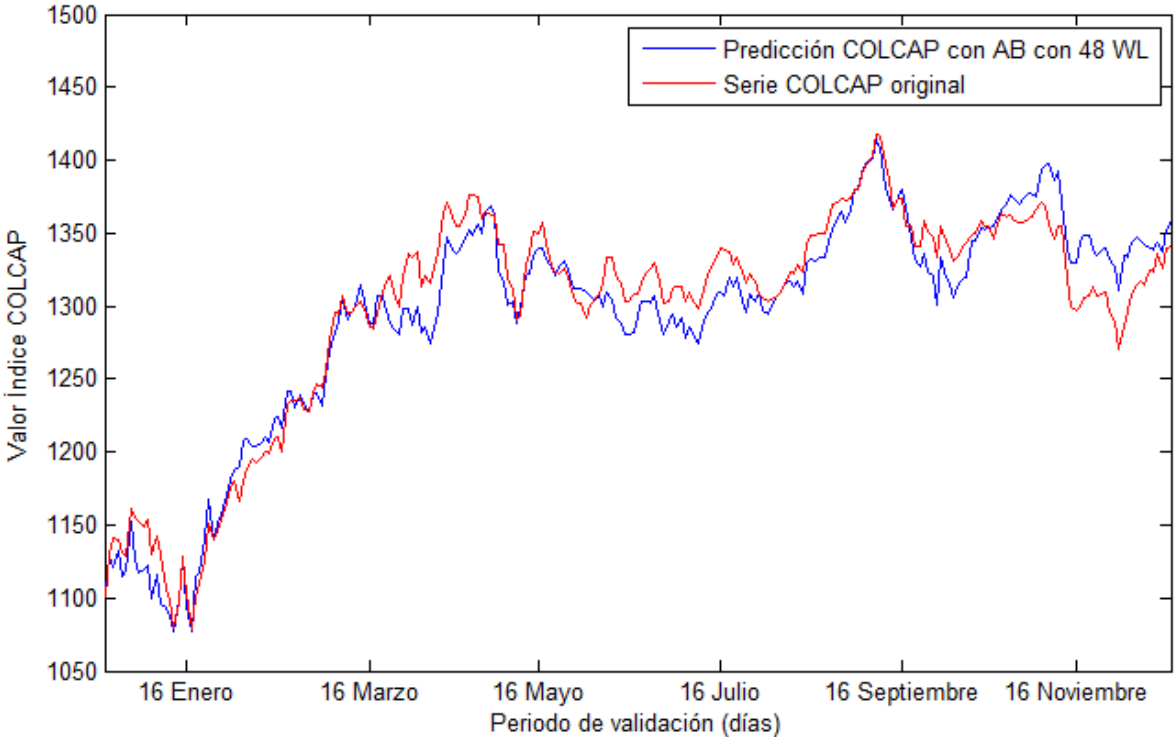


Figura 3.33. Comparación serie original y predicción realizada con el mejor modelo conjunto AdaBoost

En la Figura 3.33 se presenta una comparación entre la serie original COLCAP y la predicción de la serie realizada por medio del mejor modelo conjunto AdaBoost que cuenta con 48 aprendices débiles. El error medio absoluto (MAPE) entre estas series es 1.247%, además se puede observar que la predicción tiene un comportamiento parecido a la serie hasta el mes de Marzo, luego empieza a presentar error al seguir la serie aunque continúa teniendo un comportamiento similar al de la serie original.

4 PÉNDULO INVERTIDO

En este capítulo se genera el modelo del péndulo invertido a partir de las ecuaciones de variables de estado. Luego se presenta el controlador de referencia usado para la generación de la base de datos, se genera el modelo del controlador de referencia y se evalúa el desempeño del sistema realimentado. Se generan tres bases de datos introduciendo ruido con diferentes potencias, a partir de las cuales se generan los controladores basados en redes neuronales y AdaBoost. Se define el método utilizado para obtener los modelos de control, así como las características usadas para el modelo de redes neuronales y de AdaBoost.RT con φ auto-adaptativo. Finalmente se presentan los resultados obtenidos a través de las pruebas realizadas, se realiza la comparación de los modelos para cada potencia de ruido y se comparan con el controlador de referencia.

4.1 SIMULACIÓN DEL SISTEMA

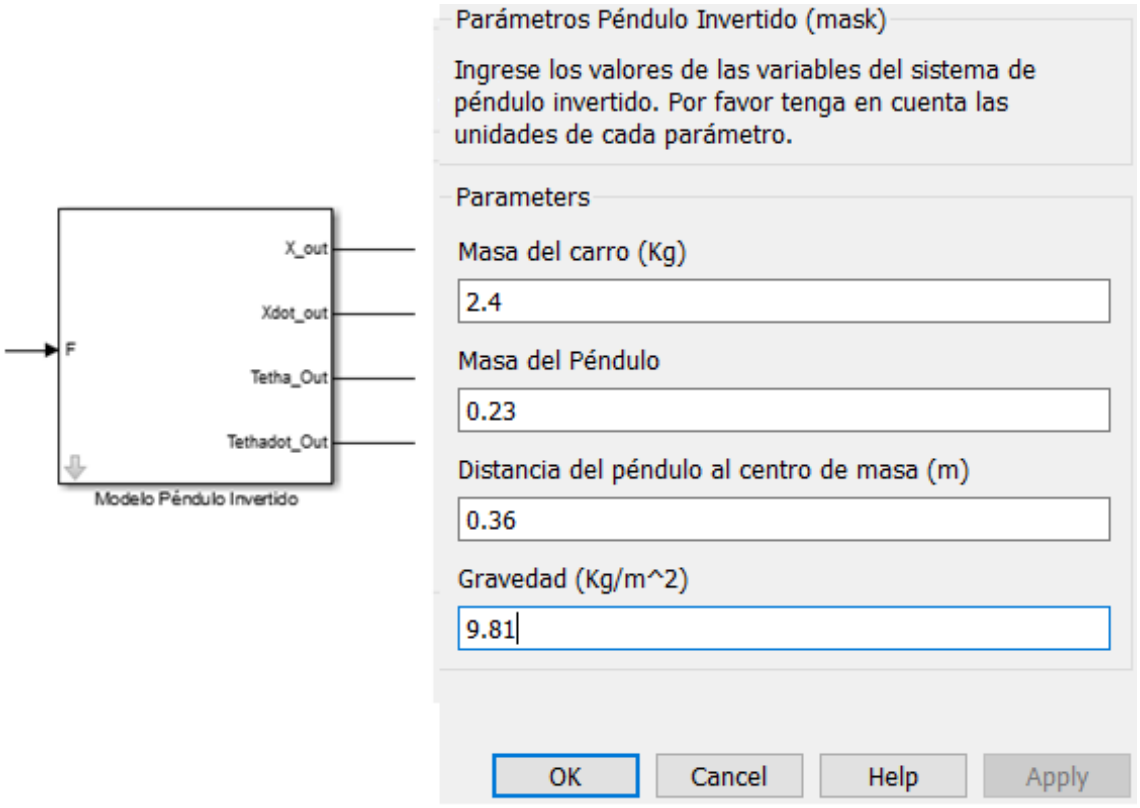


Figura 4.1. Máscara Péndulo Invertido

En la sección 2.4 se presenta el desarrollo matemático para llegar a (19) y (20) que son las que describen el comportamiento del sistema en términos de las variables de estado que son la posición del carro (x), su derivada $(\dot{x})$, el ángulo que forma el péndulo con el eje vertical (θ) y su respectiva derivada $(\dot{\theta})$. A partir de estas ecuaciones se realiza el modelo en Simulink®. El modelo utilizado para el desarrollo del proyecto se presenta en el Anexo 4, el cual fue construido por medio de

integradores, sumadores, bloques de ganancia, entre otros bloques matemáticos con los que cuenta la herramienta.

.

En la Figura 3.31 se presenta el error medio porcentual de los mejores modelos conjuntos generados variando el número de aprendices débiles con $\mathrm{r} = 0.7$. No se observa una relación directa entre el error mínimo de las pruebas y la variación de la cantidad de aprendices débiles. El mejor modelo conjunto tiene un error de 1.763% y tiene 16 aprendices débiles

4.2 MODELO DE CONTROL DEL PÉNDULO INVERTIDO

Puesto que en la revisión bibliográfica realizada no se encontró aplicaciones en control del algoritmo AdaBoost y por tanto no se tiene conocimiento sobre si el algoritmo tiene la capacidad de generar un modelo capaz de tener un comportamiento no lineal para llevar el control de un sistema, se escogió el modelo más sencillo de control para llevar a cabo la generación del modelo. Se escogió el método de control clásico, como se presenta en la Figura 4.2.

Para llevar a cabo este modelo de control con métodos de Inteligencia Artificial se puede hacer uso de algoritmos de aprendizaje supervisado y no supervisado para el desarrollo del controlador y se llegó a la conclusión de que tener conocimientos a priori del proceso controlado, (aprendizaje supervisado) conlleva mejores resultados en el control [54]. Para hacer aprendizaje supervisado es necesario contar con un controlador de referencia que tenga el comportamiento que se desea que tenga el sistema.

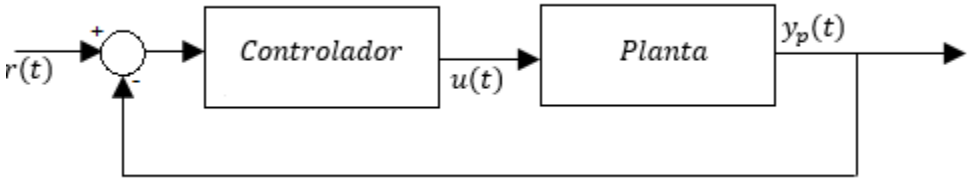


Figura 4.2 Modelo de control clásico

Luego de que se tiene el controlador y se comprueba el correcto funcionamiento del mismo, se genera una base de datos de entrada-salida de la planta controlada y a partir de esta entrenar el controlador basado en Inteligencia Artificial como se presenta en la *Figura 4.3*, donde $x_m(t)$ son los parámetros de entrada de la base de datos de entrenamiento, $y_m(t)$ el comportamiento que se desea que tenga el sistema, $y_{IA}(t)$ es la señal controlada por medio del controlador basado en Inteligencia Artificial, $e(t)$ es la diferencia entre la señal del controlador de referencia y el controlador basado en IA y finalmente el mecanismo de ajuste se refiere al método usado para realizar el entrenamiento del controlador basado en IA.

Puesto que el proceso de entrenamiento se lleva a cabo de manera offline, es decir, el modelo de control se entrena a partir del control de referencia, pero no tiene influencia sobre el sistema de la planta en el proceso de entrenamiento, luego se escoge el mejor controlador y se aplica sobre la planta.

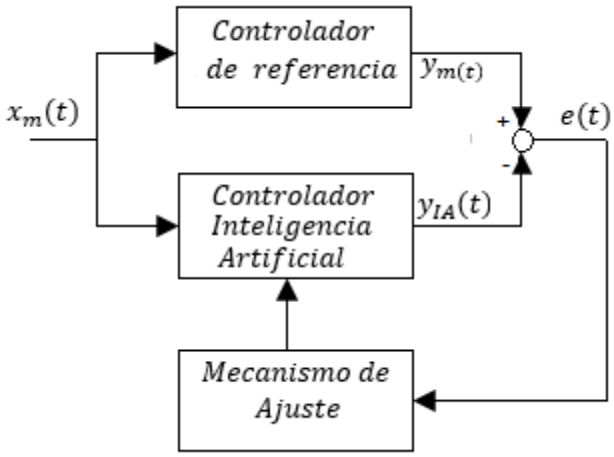


Figura 4.3 Entrenamiento del controlador basado en Inteligencia Artificial

4.2.1 Generación de la base de datos de entrenamiento

El péndulo invertido es un sistema inestable en lazo abierto, lo que indica que para poder generar la base de datos de entrenamiento requerida es necesario estabilizar el sistema por medio de un controlador de realimentación. Dado que el objetivo de este proyecto radica en controlar el sistema de péndulo invertido por medio de Redes Neuronales y AdaBoost.RT con φ auto-adaptativo se hace uso de un controlador encontrado en la literatura [54], el cual se presenta a continuación.

Se hace uso de un controlador diseñado por medio de Linealización de Realimentación y Transformada de Desacoplamiento (FLDT), que se usa para cancelar las no linealidades del sistema y transformarlo en un sistema lineal controlado. Las dinámicas del sistema parecen ser lineales y pueden ser controladas por medio de teorías de control lineal. Luego de FLDT, la regla de control usada por el controlador queda descrita por (23).

$$U = \frac{f_2}{h_2}\left[h_1 + k_1(\theta - \theta_d) + k_2\dot{\theta} + c_1(x - x_d) + c_2\dot{x}\right] - f_1 \tag{23}$$

Donde

$$h_1 = \frac{3}{4l} g \sin\theta$$

$$f_1 = m\left(l \sin\theta\, \dot{\theta}^2 - \frac{3}{8} g \sin 2\theta\right)$$

$$h_2 = \frac{3}{4l}\cos\theta$$

$$f_2 = M + m\left(l - \frac{3}{4}\cos^2\theta\right)$$

$$k_1 = 25, \qquad k_2 = 10, \qquad c_1 = 1, \qquad c_2 = 26$$

θ_d es el valor deseado de θ, que en nuestro caso $\theta_d = 0$.
x_d es el valor deseado de la posición del carro, para el caso $x_d = 0$.

Haciendo uso de la regla de control presentada en (23), se realiza el diseño del modelo de control en Simulink®, el cual se presenta en el Anexo 5. Al igual que en el caso del modelo del péndulo invertido, se crea un subsistema, cuyas entradas son las variables de estado del péndulo invertido $(x, \dot{x}, \theta, \dot{\theta})$ y la salida es la fuerza necesaria para mantener el sistema balanceado. De la misma manera, se enmascara el sistema, para en caso de necesitar variar los parámetros del péndulo invertido sea más fácil que hacer los cambios en el modelo que se presenta en el diagrama de bloques del péndulo invertido controlado se presenta en la Figura 4.4.

En la Figura 4.4 se pueden observar los subsistemas del modelo del péndulo invertido y del controlador de referencia, la señal θ está medida en radianes, por eso es necesario realizar la conversión de radianes a grados, que es el bloque de ganancia que se presenta en la parte superior de la figura. El bloque Inputs se encarga de capturar la historia de las 4 variables de estado del péndulo invertido y guardarlas en una variable de MatLab llamada Inputs, el bloque Output cumple la misma función pero guarda la salida del controlador, es decir, la fuerza necesaria para mantener estable el sistema.

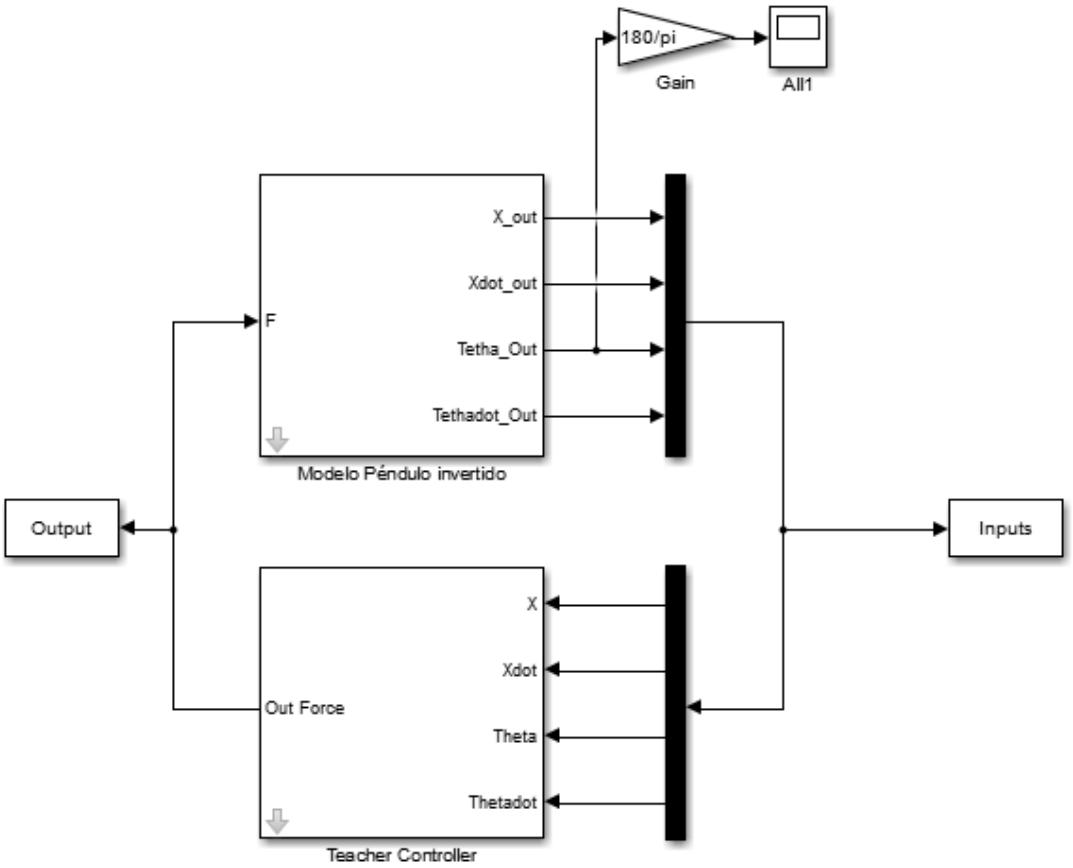


Figura 4.4. Diagrama de bloques con controlador de referencia

Antes de generar la base de datos de entrenamiento, se verificó el funcionamiento del controlador. Los valores de los parámetros del modelo del péndulo invertido utilizados para la verificación del control son masa del carro $Mc = 2.4\ Kg$, masa del péndulo $m = 0.23Kg$, longitud del péndulo $l = 0.36\ m$ y finalmente $g = 9{,}81\ Kg/m^2$, estos valores se utilizaron tanto para el modelo del péndulo,

como para el controlador. Además se aplicó un impulso unitario al sistema realimentado, de tal manera que sacara del reposo al péndulo invertido y se esperó a que el sistema se estabilizara. En la Figura 4.5 se presenta la respuesta del sistema realimentado usando como controlador el modelo de referencia. Se puede observar que tiene un rango de Overshoot de -1.729 a 1.721 y un tiempo de establecimiento de $5.4s$, a partir de lo cual se puede concluir que presenta buenos resultados dado que cumple el objetivo de estabilizar el sistema del péndulo. Debido a los resultados obtenidos por el controlador de referencia y a los encontrados en la literatura [72], [75], se considera el ruido como una variable de entrada que se suma a la fuerza calculada por el controlador. El ruido introducido al sistema es ruido blanco Gaussiano, cuya potencia se variará para comprobar el efecto de la potencia del ruido en el entrenamiento y desempeño de los controladores a diseñar.

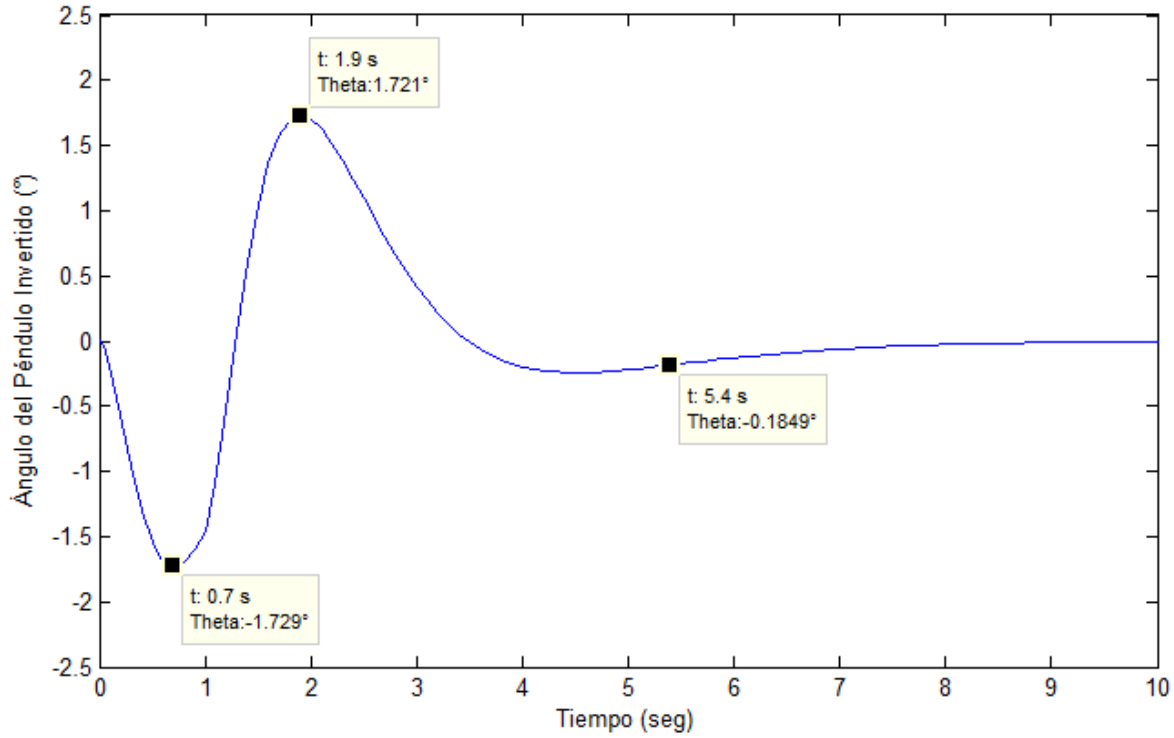


Figura 4.5. Respuesta del sistema realimentado con controlador de referencia

Dado de que se desea variar el ruido introducido al sistema, es necesario generar bases de datos para cada potencia del ruido, dado que dependiendo de la potencia, varía la señal de control (Fuerza para mantener el equilibrio del sistema). Para la generación de las bases de datos, se usan los mismos parámetros descritos en la Figura 4.1.

Tabla 4.1. Parámetros péndulo invertido

Parámetro	**Valor**
Masa del Carro (M_c)	2.4 Kg
Masa del péndulo (m)	0.23 Kg
Longitud del péndulo (l)	0.36 m
Gravedad (g)	9.81 Kg/m^2

- Potencia del ruido de 0.1%

En la Figura 4.6 se presenta la respuesta del sistema del controlador de referencia con ruido blanco con potencia del 0.1%. Se observa que el sistema presenta un Overshoot en el rango de $-1.77°$ a $1.839°$ y aunque no llega a estabilizarse y mantenerse en $\theta = 0$ como se presenta en el caso en el que no posee ruido, el sistema se mantiene en un rango entre $-0.2864°$ y $0.246°$. °, el error de estabilización del sistema es del 15.55% con respecto al Overshoot del sistema. La señal mostrada es usada como señal de control para llevar a cabo el entrenamiento de los modelos.

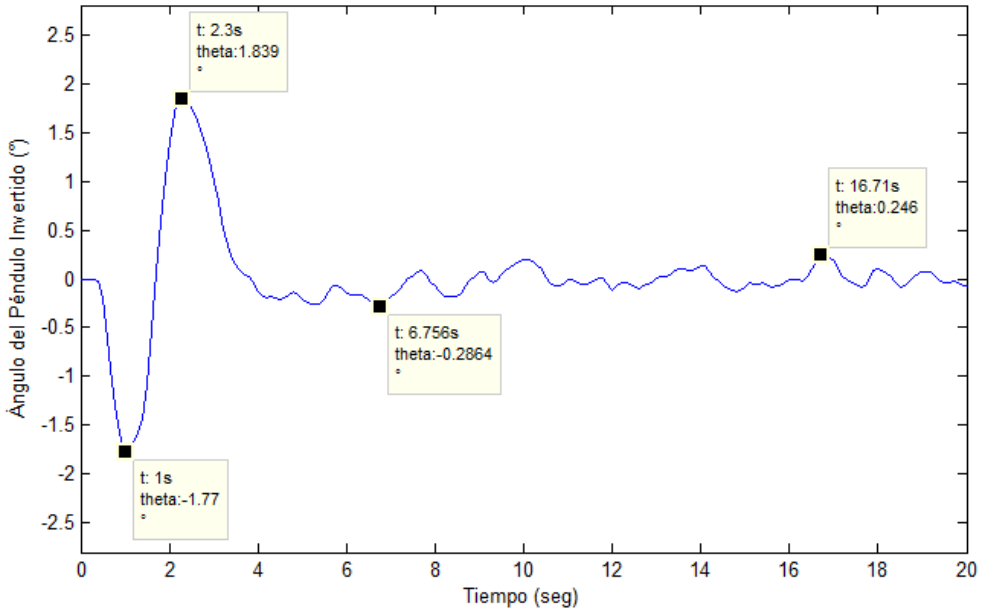


Figura 4.6. Respuesta del sistema realimentado con controlador de referencia con ruido blanco con potencia 0.1%

- Potencia del ruido de 1%

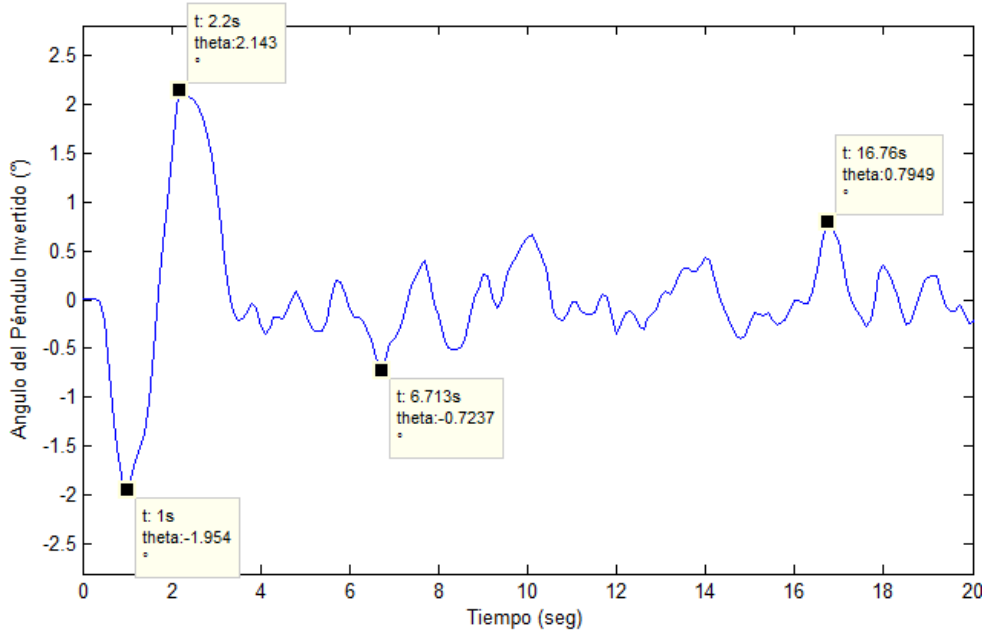


Figura 4.7. Respuesta del sistema realimentado con controlador de referencia con ruido blanco con potencia 1%

En la Figura 4.7 se presenta la respuesta del sistema del controlador de referencia con ruido blanco con potencia del 1%. Se observa que el sistema presenta un Overshoot en el rango del $-1.954°$ a $2.143°$ y aunque no llega a estabilizarse y mantenerse en $\theta = 0$ como se presenta en el caso en el que no posee ruido, el sistema se mantiene en un rango entre $-0.7237°$ y $0.7949°$, el error de estabilización del sistema es del 37.1% con respecto al Overshoot del sistema y es mayor al que se presenta con ruido de potencia 0.1% que es del 15.55%. La señal mostrada es usada como señal de control para llevar a cabo el entrenamiento de los modelos.

- Potencia del ruido de 5%

En la Figura 4.8 se presenta la respuesta del sistema del controlador de referencia" con ruido blanco con potencia del 5%. Se observa que el sistema presenta un Overshoot en el rango del $-2.293°$ a $2.679°$ y aunque no llega a estabilizarse y mantenerse en $\theta = 0$ como se presenta en el caso en el que no posee ruido, el sistema se mantiene en un rango entre $-1.488°$ y $1.768°$, el error de estabilización del sistema es del 65.99% con respecto al Overshoot del sistema y es mayor al que se presenta con ruido de potencia 0.1% que es del 15.55% y del ruido con potencia 1% que es del 37.1%. La señal mostrada es usada como señal de control para llevar a cabo el entrenamiento de los modelos. Aunque se consideraría que el sistema no es estable debido al alto porcentaje del rango de estabilización, de acuerdo a la literatura [73], [74], [76], se puede considerar un rango de estabilidad dentro de 5° a -5°, por lo cual si el sistema se mantiene dentro de este rango se considera estable. En este caso, tanto el rango de Overshoot como el de estabilidad se mantienen dentro de este rango lo que permite considerar el sistema controlado.

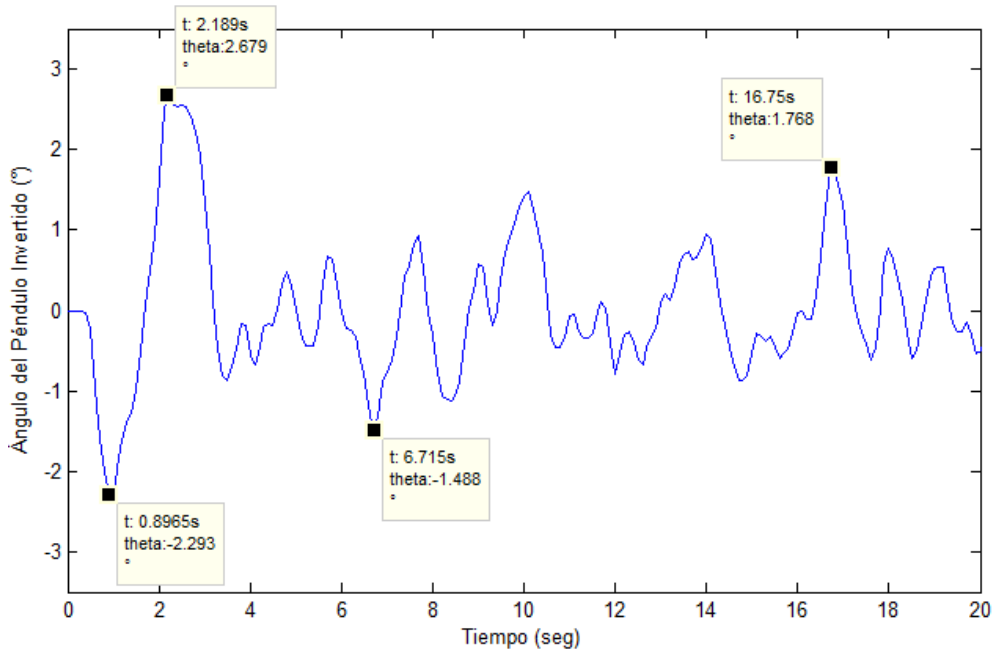


Figura 4.8. Respuesta del sistema realimentado con controlador de referencia con ruido blanco con potencia 5%

4.2.2 Modelo de control del péndulo invertido con redes neuronales.

Para realizar el entrenamiento de las redes neuronales que serán usadas para el control del sistema del péndulo invertido se hace uso de MatLab®. En cuanto a la arquitectura de la red, se decidió contar con una sola capa oculta dado que una red neuronal con una capa oculta es considerada un aproximador universal [50]. Además, dado que no es posible de manera a priori conocer el número de neuronas necesarias para aprender el comportamiento del controlador de referencia, se diseñó un algoritmo que variaba el número de neuronas en la capa oculta entre 1 y 50 y se repitió este procedimiento 200 veces de tal manera que se tuviera una base de datos lo suficientemente grande y variada dado que las redes neuronales dependen mucho de la inicialización de los pesos que se realiza de manera aleatoria.

Los parámetros utilizados para el entrenamiento de las redes neuronales se presentan en la Tabla 4.2. En la Figura 4.9 se presenta la estructura general de los modelos creados a partir de redes neuronales. Son 4 entradas, cada una representa una variable de estado del sistema normalizadas entre 0 y 1, NNCO es el número de neuronas de la capa oculta que será variado entre 1 y 50 para encontrar la arquitectura con mejor desempeño, la función de activación de las neuronas de la capa oculta es Sigmoidea Tangencial Hiperbólica, la función de entrenamiento de la red para actualizar el valor de los pesos y el bias de cada neurona se lleva a cabo de acuerdo al algoritmo de optimización Levenberg-Marquardt, se cuenta con una neurona en la capa de salida con función de activación limitador duro que representa el valor de la fuerza necesaria para mantener el péndulo estable. Finalmente, es necesario resaltar que la inicialización de las sinapsis de la red neuronal se hace de manera aleatoria.

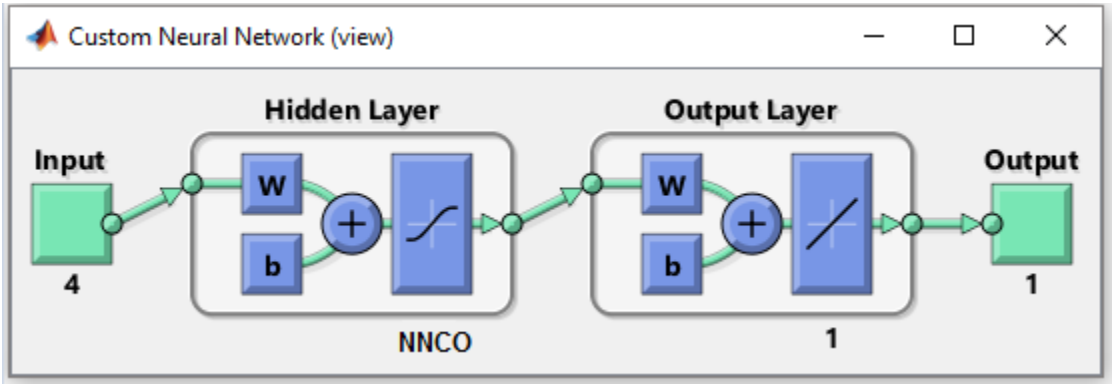


Figura 4.9. Estructura general Red Neuronal para el control del péndulo invertido

Tabla 4.2. Parámetros de entrenamiento de las redes neuronales

Parámetro	Valor
Número de Capas Ocultas	1
Número de neuronas de la capa oculta	1-50
Función de activación neuronas capa oculta	Sigmoidea Tangencial Hiperbólica
Función de activación capa de salida	Limitador duro
Función de entrenamiento de la red	Levenberg - Marquardt
Número máximo de Épocas	500
Tiempo máximo de entrenamiento	Infinito
Desempeño deseado	$1 * 10^{-7}$
Gradiente mínimo	$1 * 10^{-7}$
Número de pruebas de validación	6

4.2.3 Modelo de control del péndulo invertido por medio de AdaBoost.RT con φ auto-adaptativo.

Para realizar el entrenamiento del modelo conjunto del algoritmo AdaBoost para controlar el péndulo invertido, se hace uso del algoritmo AdaBoost.RT descrito en la Figura 2.8 con el modelo auto-adaptativo de φ presentado en la sección 2.2.4. En la sección 3.4.2 se explicaron las funciones adaboost_reg, que se encuentra en el Anexo 2 y weak_learner cuyo código se presenta en el Anexo 3, las cuales son utilizadas para llevar a cabo el entrenamiento de los modelos conjuntos empleados para controlar el sistema del péndulo invertido. En la Tabla 4.3, se presentan los parámetros del algoritmo AdaBoost.RT con φ auto-adaptativo utilizados para el entrenamiento del modelo de control del péndulo invertido. Es necesario resaltar que el número de aprendices débiles (t) se refiere a la cantidad de aprendices débiles que conforman el modelo conjunto AdaBoost y es diferente al número de iteraciones del aprendiz débil (it) que como su nombre lo indica se refiere al número de iteraciones que realiza el perceptrón para encontrar el mejor regresor que será el aprendiz débil de la iteración t, además, la inicialización de las sinapsis del algoritmo del perceptrón se realiza de manera aleatoria.

Tabla 4.3. Parámetros del algoritmo AdaBoost.RT con φ auto-adaptativo

Parámetro	**Valor**
Algoritmo de Aprendiz Débil	Perceptrón
Número de aprendices débiles (t)	1-50
Número iteraciones aprendiz débil (it)	100
Criterios de evaluación	$MAPE$
$\varphi_{inicial}$	0.1-0.5
r	0.3, 0.5, 0.7
$\beta = error^n$	$n = 1, 2, 3$

4.3 RESULTADOS DE CONTROL DEL PÉNDULO INVERTIDO

4.3.1 Resultados de control del péndulo invertido obtenidos con Redes Neuronales

Se realizaron 200 pruebas variando la cantidad de neuronas de la capa oculta desde 1 hasta 50, dado que no es posible conocer de manera a priori la cantidad de neuronas que darán el mejor resultado. Primero se entrenaron las redes neuronales a partir de las bases de datos generadas previamente y se usó como criterio de evaluación para la selección de la mejor red neuronal el error medio cuadrático MAPE.

4.3.1.1 Ruido de 0.1%

En la Figura 4.10 se presenta el error medio absoluto MAPE de las mejores redes neuronales para cada número de neuronas, las cuales son aquellas que presentan el mínimo error de las 200 pruebas realizadas. Se puede observar como el error a partir de una neurona en la capa oculta empieza a disminuir, aparentemente la red con 2 neuronas presenta el menor error, pero no es así, el menor error lo presenta la red con 11 neuronas en la capa oculta que es de 1.186%. Entre 2 y 11 neuronas el comportamiento del error MAPE es variante y no se presenta un comportamiento relevante, en

cambio, a partir de las 11 neuronas de la capa oculta el error de las mejores redes neuronales empieza a aumentar a medida que aumenta el número de neuronas de la capa oculta, hasta llegar al error máximo con 47 neuronas que es de 6.353%.

Se toma la red con 11 neuronas en la capa oculta como la mejor red neuronal para el control del sistema de péndulo invertido, debido a que presenta el menor error de entrenamiento de todas las redes que fueron entrenadas con la base de datos con ruido del 0.1%.

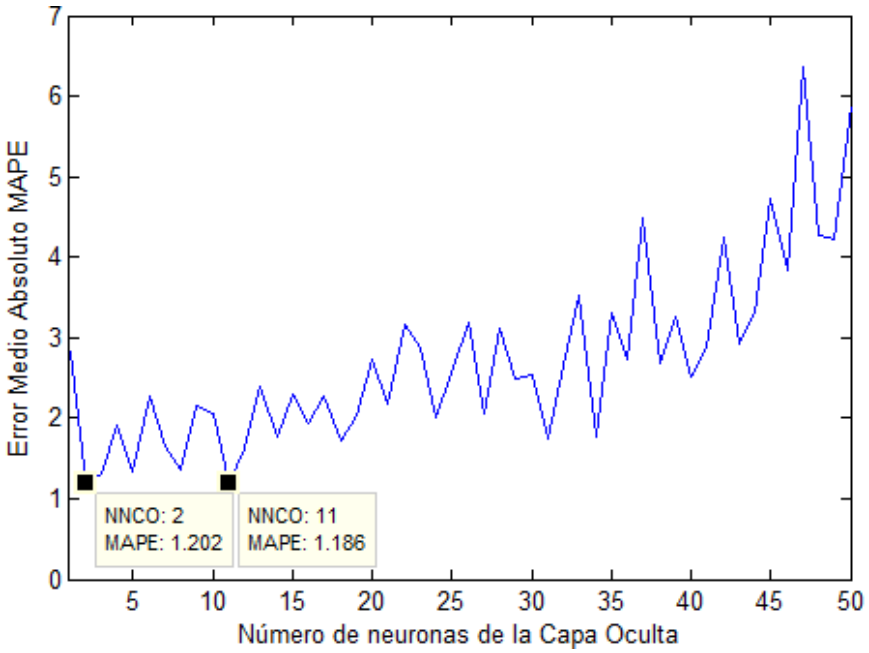


Figura 4.10. Error medio cuadrático de las mejores redes neuronales con ruido 0.1%

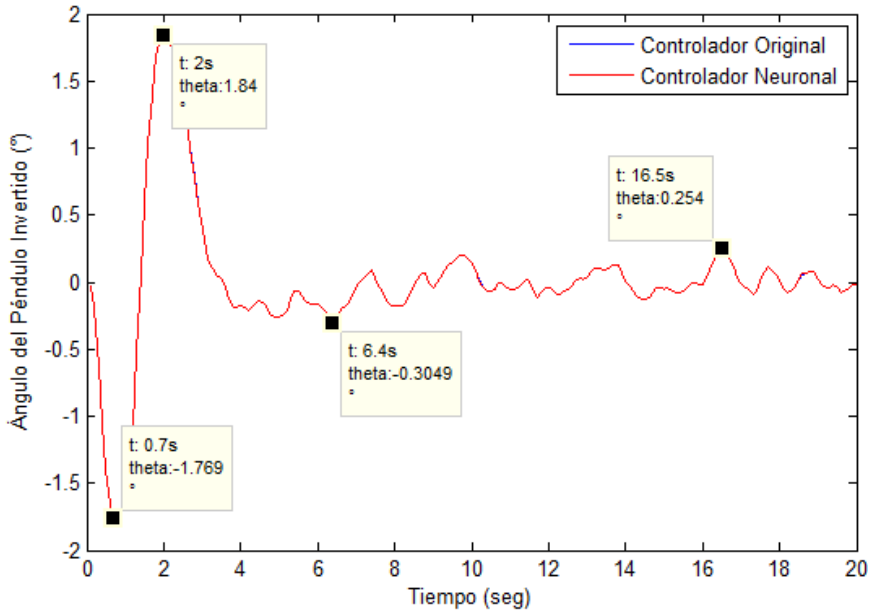


Figura 4.11. Desempeño del controlador diseñado por redes neuronales con ruido 0.1%

En la Figura 4.11 se presenta el desempeño del controlador diseñado por redes neuronales comparado con el controlador usado para el entrenamiento, en la figura solo se observa la gráfica de estabilización del controlador neuronal dado que presenta el mismo comportamiento que el controlador original. El Overshoot de los controladores se mantienen en el rango de $-1.769°$ a 1.84 y aunque no llega a estabilizarse y mantenerse en $\theta = 0$ como se presenta en la Figura 4.5, el sistema se mantiene en un rango entre $-0.3049°$ y $0.254°$. Estos rangos se presentan de forma más clara en la Tabla 4.4 Parámetros de desempeño del controlador neuronal con ruido 0.1%

Tabla 4.4 Parámetros de desempeño del controlador neuronal con ruido 0.1%

	Controlador de Referencia	Controlador Neuronal
Overshoot	$-1.769°$ a $1.84°$	$1.769°$ a $1.84°$
Rango Estabilización	$-0.3049°$ a $0.254°$	$-0.3049°$ a $0.254°$

4.3.1.2 Ruido del 1%

En la Figura 4.12 se presenta el error medio absoluto MAPE de las mejores redes neuronales con la variación del número de neuronas, las cuales son aquellas que presentan el mínimo error de las 200 pruebas realizadas para cada caso. Se puede observar como el error a partir de una neurona en la capa oculta no presenta variaciones marcadas hasta las 10 neuronas, donde se presenta el error de 1.734% que es el menor de todas las pruebas. A partir de las 10 neuronas ocultas el error aumenta aunque no se presenta relación con el número de neuronas, el error máximo se presenta en el caso de la red con 43 neuronas con 13.3%.

Se toma la red con 10 neuronas en la capa oculta como la mejor red neuronal para el control del sistema de péndulo invertido, debido a que presenta el menor error de entrenamiento de todas las redes que fueron entrenadas con la base de datos con ruido del 1%.

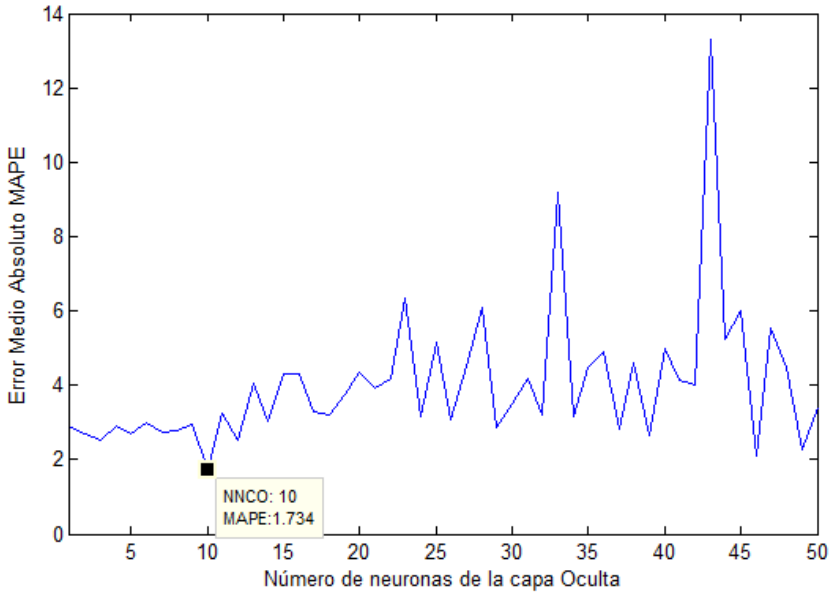


Figura 4.12. Error medio cuadrático de las mejores redes neuronales con ruido 1%

En la Figura 4.13 se presenta el desempeño del controlador diseñado por redes neuronales comparado con el controlador de referencia usado para el entrenamiento, en la figura solo se observa la gráfica de estabilización del controlador neuronal dado que presenta el mismo comportamiento que

el controlador original. El Overshoot de los controladores se mantiene en el rango de $-1.954°$ a $2.14°$ y aunque no llega a estabilizarse y mantenerse en $\theta = 0$ como se presenta en la Figura 4.5, el sistema controlado se mantiene en un rango entre $-0.737°$ y $0.8031°$. Estos rangos se presentan más claramente en la Tabla 4.5.

Tabla 4.5. Parámetros de desempeño del controlador neuronal con ruido 1%

	Controlador "Profesor"	Controlador AdaBoost
Overshoot	$-1.954°$ a $2.14°$	$-1.954°$ a $2.14°$
Rango Estabilización	$-0.737°$ y $0.8031°$	$-0.737°$ y $0.8031°$

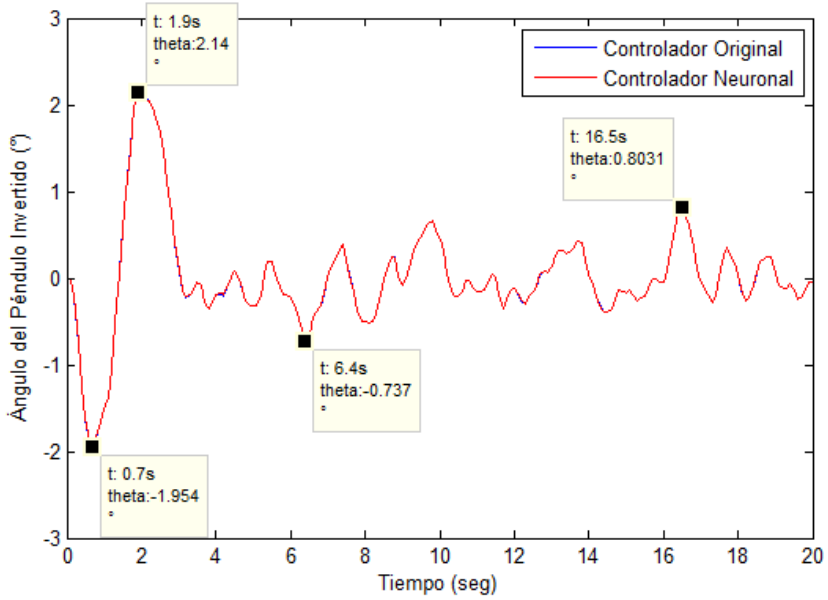


Figura 4.13. Desempeño del controlador diseñado por redes neuronales con ruido 1%

4.3.1.3 Ruido del 5%

En la Figura 4.14 se presenta el error medio absoluto MAPE de las mejores redes neuronales para cada número de neuronas, las cuales son aquellas que presentan el mínimo error de las 200 pruebas realizadas. Se puede observar como el error comienza en el máximo de las pruebas y disminuye a medida que se aumenta el número de neuronas hasta las 5 donde se obtiene el menor error que es de 0.13%, a partir de las 5 neuronas el error tiende a aumentar a medida que se aumenta el número de neuronas.

Se toma la red con 5 neuronas en la capa oculta como la mejor red neuronal para el control del sistema de péndulo invertido, debido a que presenta el menor error de entrenamiento de todas las redes entrenadas con la base de datos con ruido del 5%.

Tabla 4.6. Parámetros de desempeño del controlador neuronal con ruido 5%

	Controlador de referencia	Controlador AdaBoost
Overshoot	$-1.846°$ a $2.033°$	$-2.073°$ a $2.296°$
Rango Estabilización	$-1.571°$ a $0.7929°$	$-1.839°$ a $1.015°$

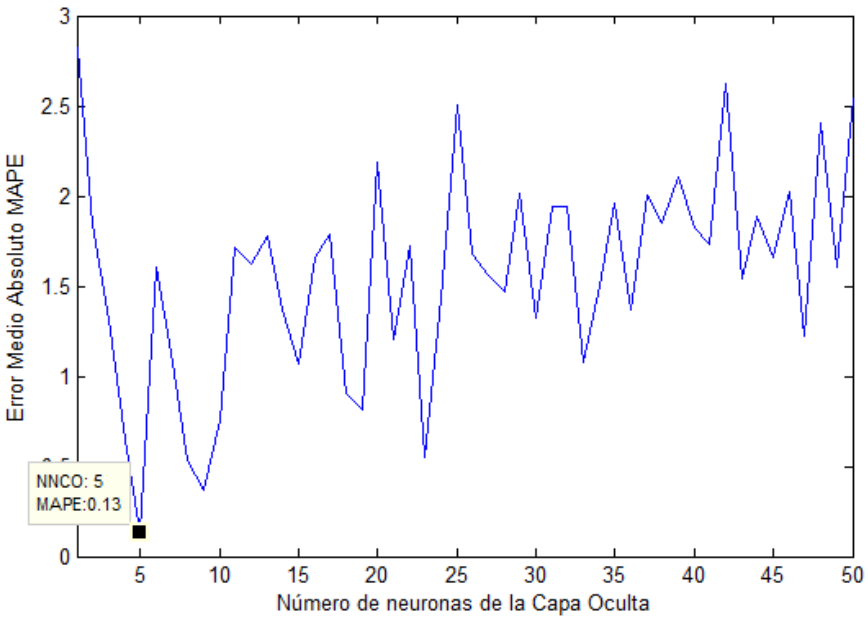


Figura 4.14 . Error medio cuadrático de las mejores redes neuronales con ruido 5%

En la Figura 4.15 se presenta el desempeño del controlador diseñado por redes neuronales comparado con el controlador usado para el entrenamiento, en la figura solo se observa la gráfica de estabilización del controlador neuronal dado que presenta el mismo comportamiento que el controlador original. El Overshoot del controlador se mantiene en el rango de $-2.288°$ a $2.703°$ y aunque no llega a estabilizarse y mantenerse en $\theta = 0$ como se presenta en la Figura 4.5, el sistema se mantiene en un rango entre $-1.516°$ y $1.795°$.

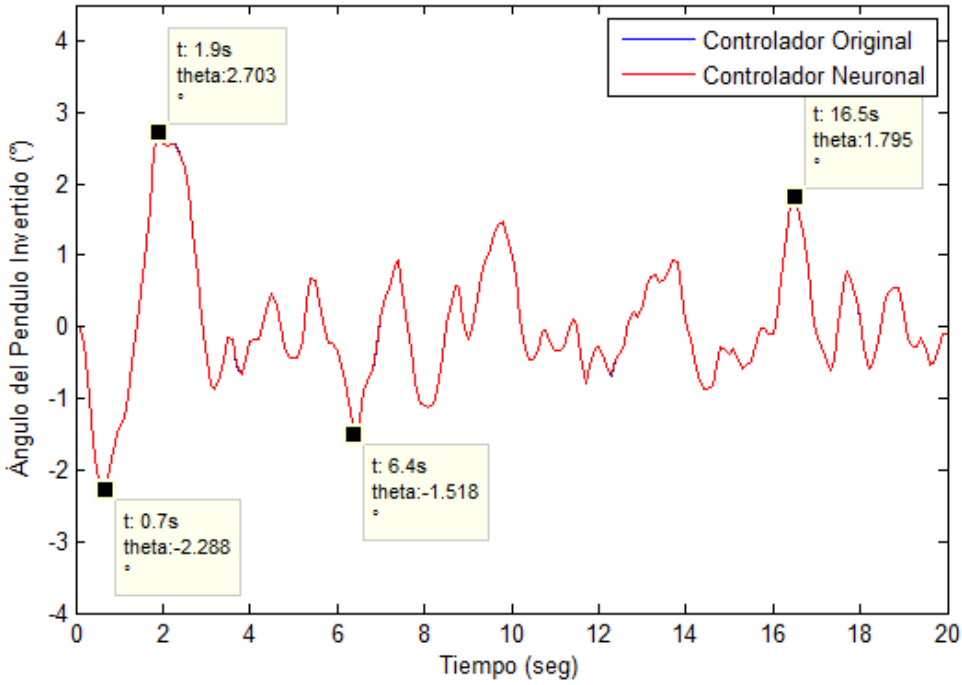


Figura 4.15. Desempeño del controlador diseñado por redes neuronales con ruido 5%

4.3.2 Resultados de control del péndulo invertido obtenidos con AdaBoost.RT con φ auto-adaptativo

Para realizar el entrenamiento del modelo conjunto que es usado para el control del sistema del péndulo invertido se hizo uso de la función presentada en el Anexo 2, usando como Weak Learner el perceptrón con la función descrita en el Anexo 3. Dentro del algoritmo de AdaBoost.RT con φ auto-adaptativo hay parámetros que pueden ser variados dependiendo del problema, por este motivo, se realiza la variación de estos parámetros para identificar el valor adecuado de cada uno de estos para el problema que se desea resolver.

En la literatura no se ha encontrado aplicación del algoritmo en problemas de control, pero dado que el entrenamiento se hace por medio de aprendizaje supervisado y se espera que el algoritmo aprenda el comportamiento de un controlador de referencia, se realiza la variación de los parámetros β_t, que es la actualización del peso de las muestras, $\varphi_{inicial}$ que hace referencia al delimitador de muestras correctas y r que tiene efecto en la tasa de cambio del error RMSE

4.3.2.1 Variación de la actualización del peso de las muestras correctas

Como se explicó en la sección 2.2.3, la actualización de la distribución de muestras D_i depende del factor β_t que se calcula como ε_t^n donde $n = 1, 2\ o\ 3$ (lineal, cuadrada o cúbica). De este factor no sólo depende la actualización de la distribución de muestras, también depende el peso del aprendiz débil en la iteración t, como se observa en (5).

Se realizaron 200 pruebas variando la manera de actualización de distribución entre lineal, cuadrada, cúbica, dejando constante el valor inicial de $\varphi = 0.2$ y $r = 0.5$, según las conclusiones presentadas en [38].

Al igual que en el caso del entrenamiento de redes neuronales, no es posible conocer de manera a priori el número de aprendices débiles que darán el mejor resultado, por este motivo, las pruebas fueron realizadas variando la cantidad de aprendices débiles desde 1 hasta 50.

- **Ruido de 0.1%**

En la Figura 4.16 se presenta el error medio absoluto (MAPE) de los mejores modelos conjuntos obtenidos en las 200 pruebas realizadas variando el número de aprendices débiles. A partir de la figura se puede observar que en general el método de actualización del peso de las muestras no conlleva a la disminución o aumento del error medio absoluto. El mejor modelo obtenido con esta variación se presenta con $\beta_t = \varepsilon_t^2$ actualización cuadrada y 32 aprendices débiles con un error del 1.204%. En la Tabla 4.7 se presenta un resumen de los mejores modelos conjuntos para cada tipo de actualización.

Tabla 4.7 Mejores modelos conjuntos de control con $\beta = \varepsilon^n$ y ruido 0.1%

Parámetro	# Aprendices débiles	MAPE (%)
$\beta = \varepsilon$	46	1.259
$\beta = \varepsilon^2$	32	1.204
$\beta = \varepsilon^3$	44	1.456

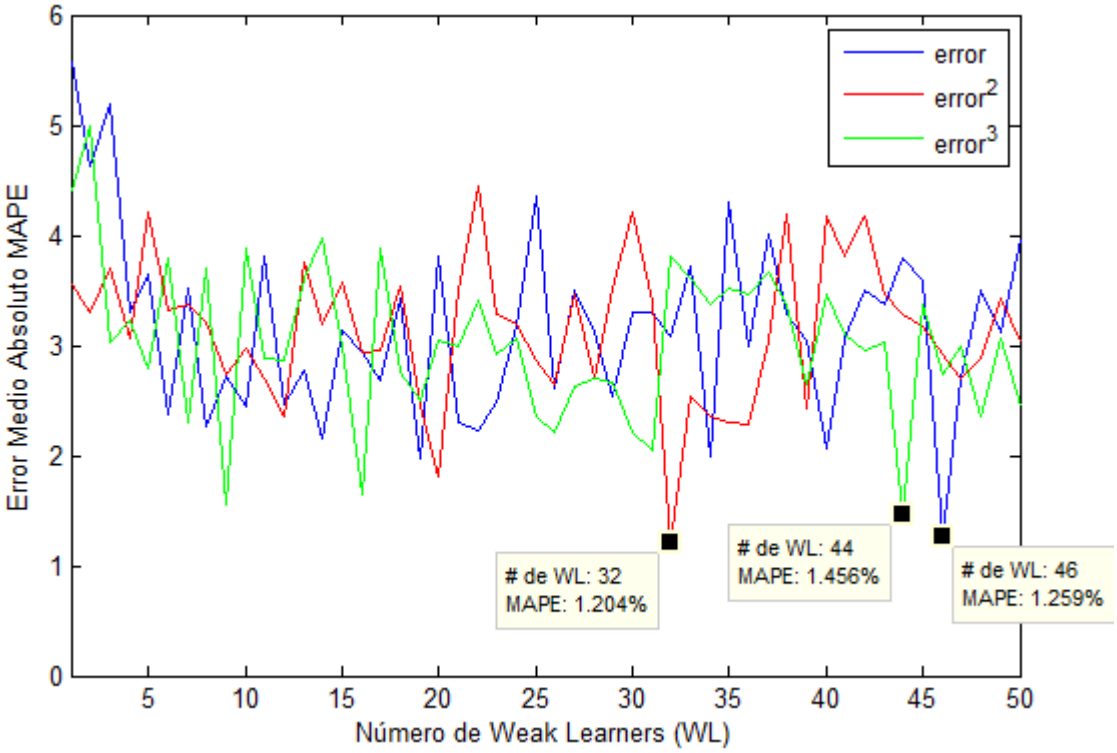


Figura 4.16 Error de AdaBoost.RT con variación de la actualización de pesos β_t para control con ruido 0.1%

- **Ruido de 1%**

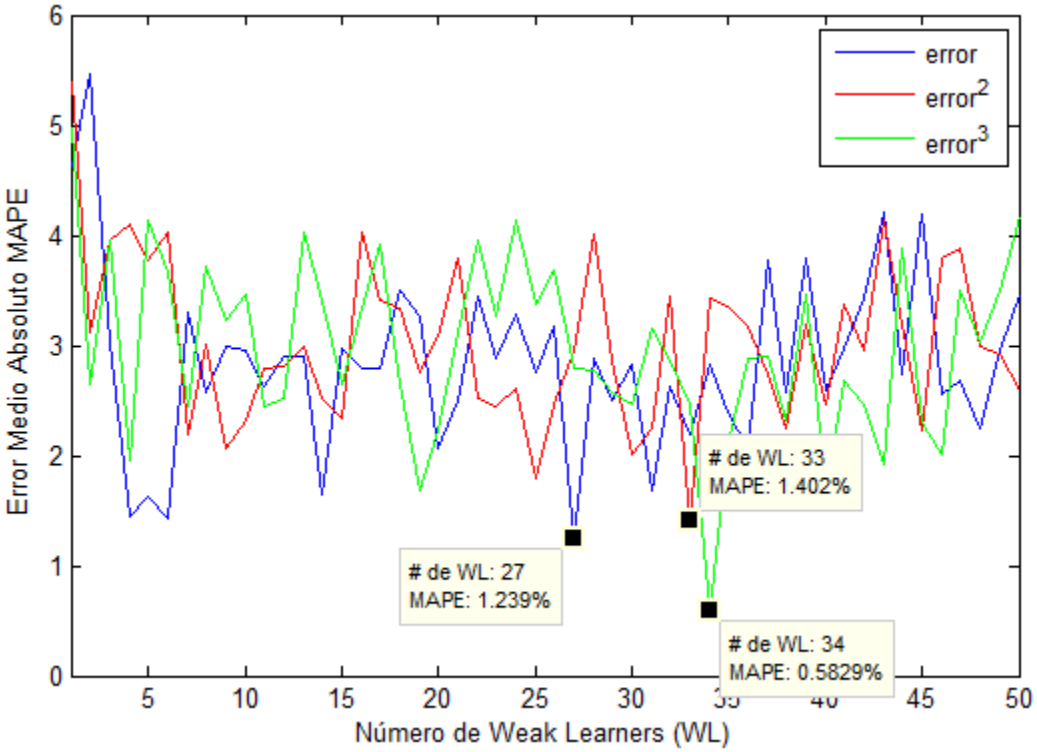


Figura 4.17 Error de AdaBoost.RT con variación de la actualización de pesos β_t para control con ruido 1%

En la Figura 4.17 se presenta el error medio absoluto (MAPE) de los mejores modelos conjuntos obtenidos en las 200 pruebas realizadas variando el número de aprendices débiles. A partir de la figura se puede observar que en general el método de actualización del peso de las muestras no conlleva a la disminución o aumento del error medio absoluto. El mejor modelo obtenido con esta variación se presenta con $\beta_t = \varepsilon_t^3$ actualización cúbica y 34 aprendices débiles con un error del 0.5829%. En la Tabla 4.8 se presenta un resumen de los mejores modelos conjuntos para cada tipo de actualización.

Tabla 4.8 Mejores modelos conjuntos de control con $\beta = \varepsilon^n$ y ruido 1%

Parámetro	# Aprendices débiles	MAPE (%)
$\beta = \varepsilon$	27	1.239
$\beta = \varepsilon^2$	33	1.402
$\beta = \varepsilon^3$	34	0.582

- **Ruido de 5%**

En la Figura 4.18 se presenta el error medio absoluto (MAPE) de los mejores modelos conjuntos obtenidos en las 200 pruebas realizadas variando el número de aprendices débiles. A partir de la figura se puede observar que en general el error posee un comportamiento similar en los tres casos, por lo menos hasta los 10 aprendices débiles, a partir de los 10 aprendices los errores presentan comportamientos diversos pero no se presenta una relación en ninguno de los casos entre el error y el método de actualización del peso de las muestras con la variación de aprendices débiles. El mejor modelo obtenido con esta variación se presenta con $\beta_t = \varepsilon_t$ actualización lineal y 15 aprendices débiles con un error del 1.085%. En la Tabla 4.9 se presenta un resumen de los mejores modelos conjuntos para cada tipo de actualización.

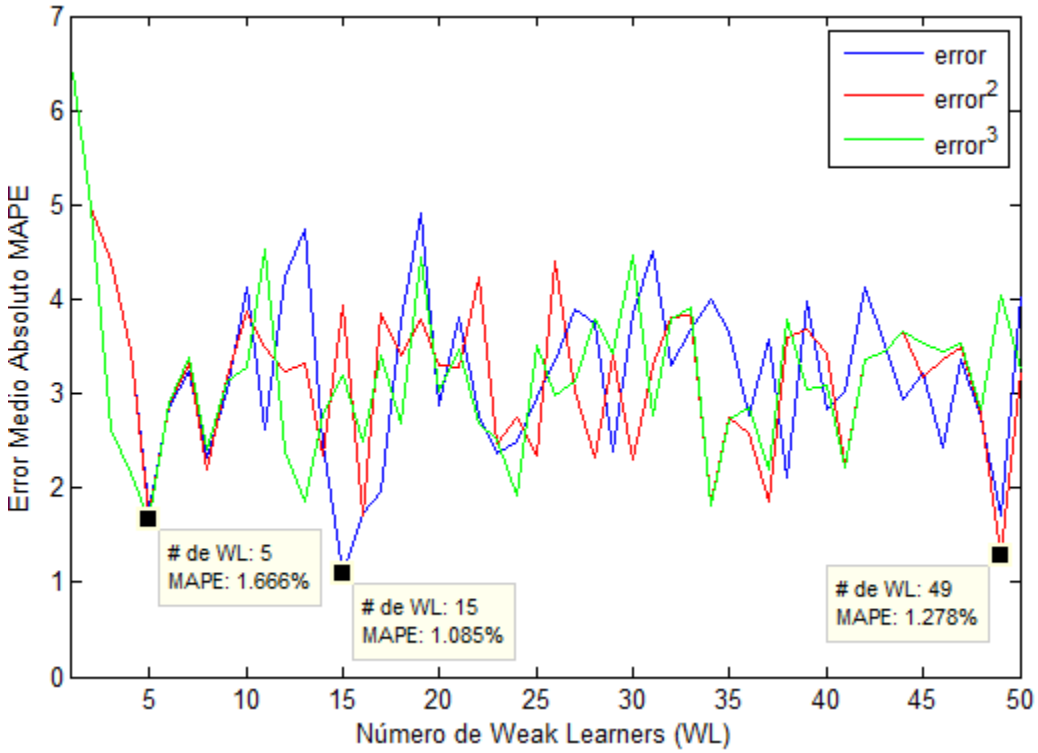


Figura 4.18. Error de AdaBoost.RT con variación de la actualización de pesos β_t para control con ruido 5%

Tabla 4.9 Mejores modelos conjuntos de control con $\beta = \varepsilon^n$ y ruido 5%

Parámetro	# Aprendices débiles	MAPE (%)
$\beta = \varepsilon$	15	1.085
$\beta = \varepsilon^2$	49	1.278
$\beta = \varepsilon^3$	5	1.666

4.3.2.2 Variación del valor inicial de φ

La mayor ventaja e innovación del algoritmo de AdaBoost utilizado en este proyecto, se basa en la adaptabilidad de φ dependiendo del error del problema, pero no se considera el valor inicial del parámetro, aunque en [38] se aconseja que se inicialice con un valor entre 0.2 y 0.4,dado que con una inicialización mayor a 0.4, el algoritmo se vuelve inestable. De acuerdo a este, se varía el valor inicial de φ para encontrar el óptimo para el problema de control propuesto.

Para llevar a cabo esto, se realizaron 200 pruebas variando el valor inicial de φ, desde 0.1 hasta 0.5, variando la cantidad de aprendices débiles desde 1 hasta 50, donde se toma el mejor modelo conjunto de cada una de las variaciones y se escoge el valor óptimo de φ para el problema que se desea resolver.

- **Ruido 0.1%**

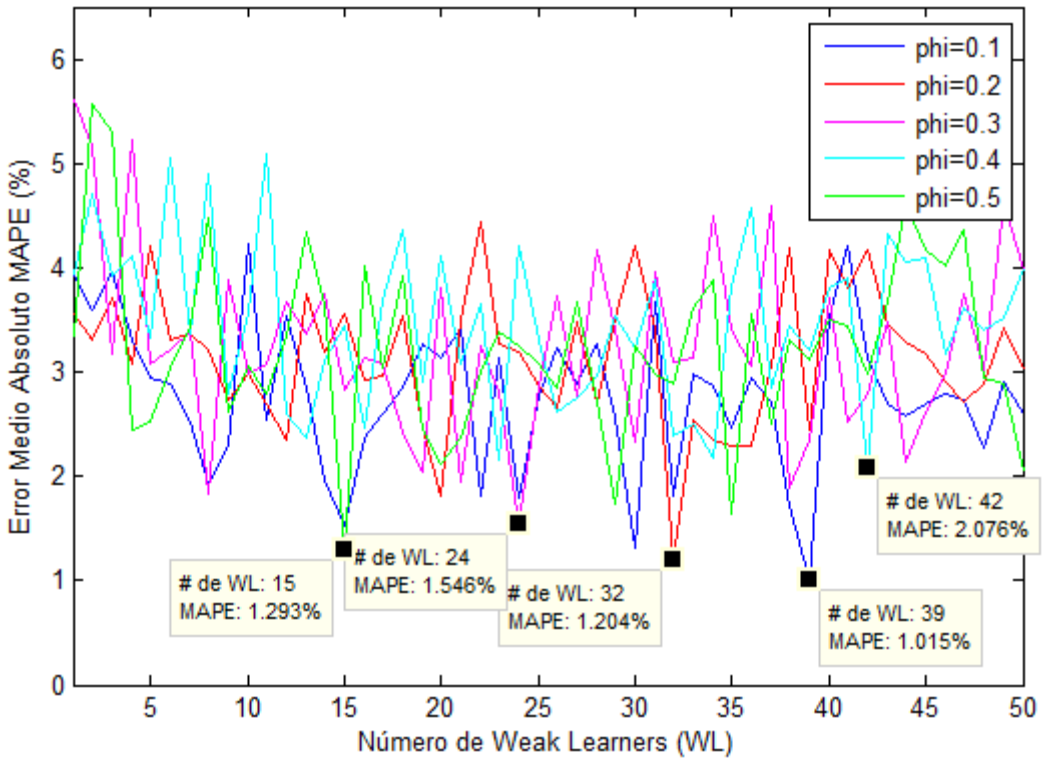


Figura 4.19 Error de AdaBoost.RT con variación de $\varphi_{inicial}$ para control con ruido 0.1%

En la Figura 4.19 se presenta el error medio absoluto MAPE de los mejores modelos conjuntos, donde no se observa relación entre el valor inicial de φ y el error o el error y el aumento del número

de aprendices débiles que conforman el modelo conjunto. El mejor modelo conjunto se presentó con $\varphi_{\text{inicial}} = 0.1$ con un error de 1.015%. En la

Tabla *4.10* se presenta el resultado del mejor modelo conjunto para cada φ_{inicial}.

Tabla 4.10 Mejores modelos conjuntos de control variando $\varphi_{inicial}$ y ruido 0.1%

Parámetro	# Aprendices débiles	MAPE (%)
$\varphi_{\text{inicial}} = 0.1$	39	1.015
$\varphi_{inicial} = 0.2$	32	1.204
$\varphi_{inicial} = 0.3$	24	1.546
$\varphi_{inicial} = 0.4$	42	2.076
$\varphi_{inicial} = 0.5$	15	1.293

- **Ruido 1%**

En la Figura 4.20 se presenta el error medio absoluto MAPE de los mejores modelos conjuntos, donde no se observa relación entre el valor inicial de φ y el error o el error y el aumento del número de aprendices débiles que conforman el modelo conjunto. El mejor modelo conjunto se presentó con $\varphi_{\text{inicial}} = 0.2$ con un error de 0.583%. En la Tabla 4.11 se presenta el resultado del mejor modelo conjunto para cada φ_{inicial}.

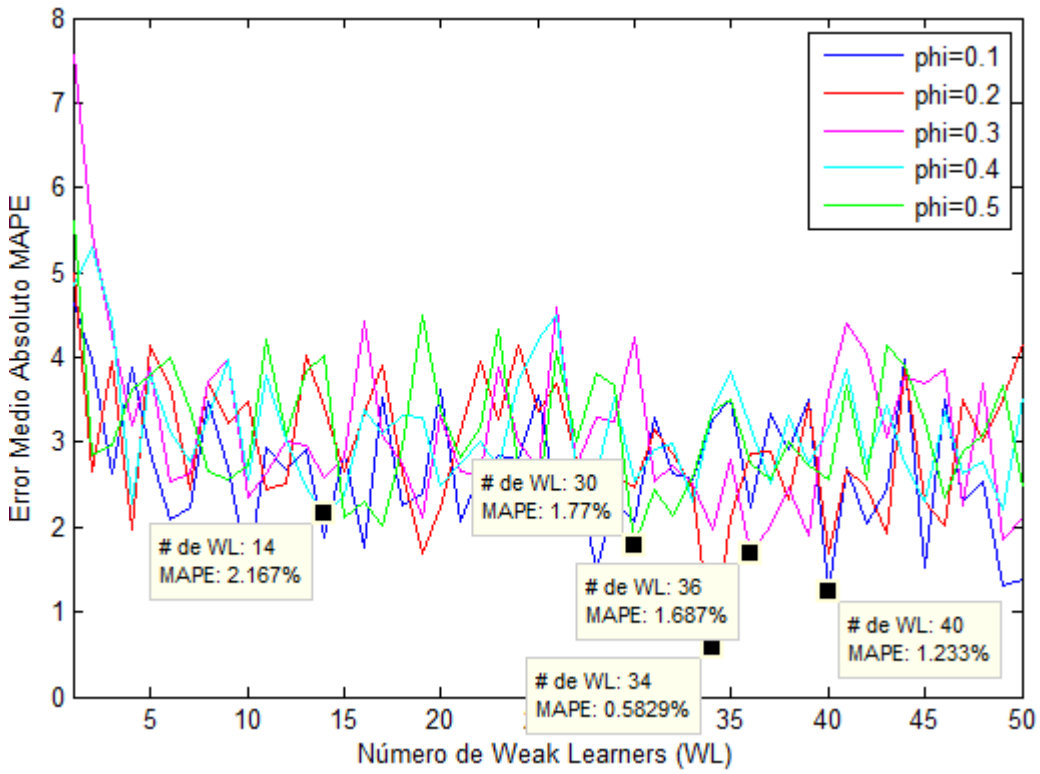


Figura 4.20. Error de AdaBoost.RT con variación de $\varphi_{inicial}$ para control con ruido 1%

Tabla 4.11 Mejores modelos conjuntos de control variando $\varphi_{inicial}$ con ruido 1%

Parámetro	# Aprendices débiles	MAPE (%)
$\varphi_{\text{inicial}} = 0.1$	40	1.233

$\varphi_{inicial} = 0.2$	34	0.583
$\varphi_{inicial} = 0.3$	36	1.687
$\varphi_{inicial} = 0.4$	14	2.167
$\varphi_{inicial} = 0.5$	30	1.77

- **Ruido 5%**

En la Figura 4.21 se presenta el error medio absoluto MAPE de los mejores modelos conjuntos, donde no se observa relación entre el valor inicial de φ y el error o el error y el aumento del número de aprendices débiles que conforman el modelo conjunto. El mejor modelo conjunto se presentó con $\varphi_{\text{inicial}} = 0.4$ con un error de 0.923%. En la Tabla 4.12 se presenta el resultado del mejor modelo conjunto para cada φ_{inicial}.

Tabla 4.12 Mejores modelos conjuntos de control variando $\varphi_{inicial}$ con ruido 5%

Parámetro	# Aprendices débiles	MAPE (%)
$\varphi_{\text{inicial}} = 0.1$	16	1.418
$\varphi_{\text{inicial}} = 0.2$	15	1.085
$\varphi_{inicial} = 0.3$	47	1.421
$\varphi_{inicial} = 0.4$	34	0.923
$\varphi_{inicial} = 0.5$	37	1.51

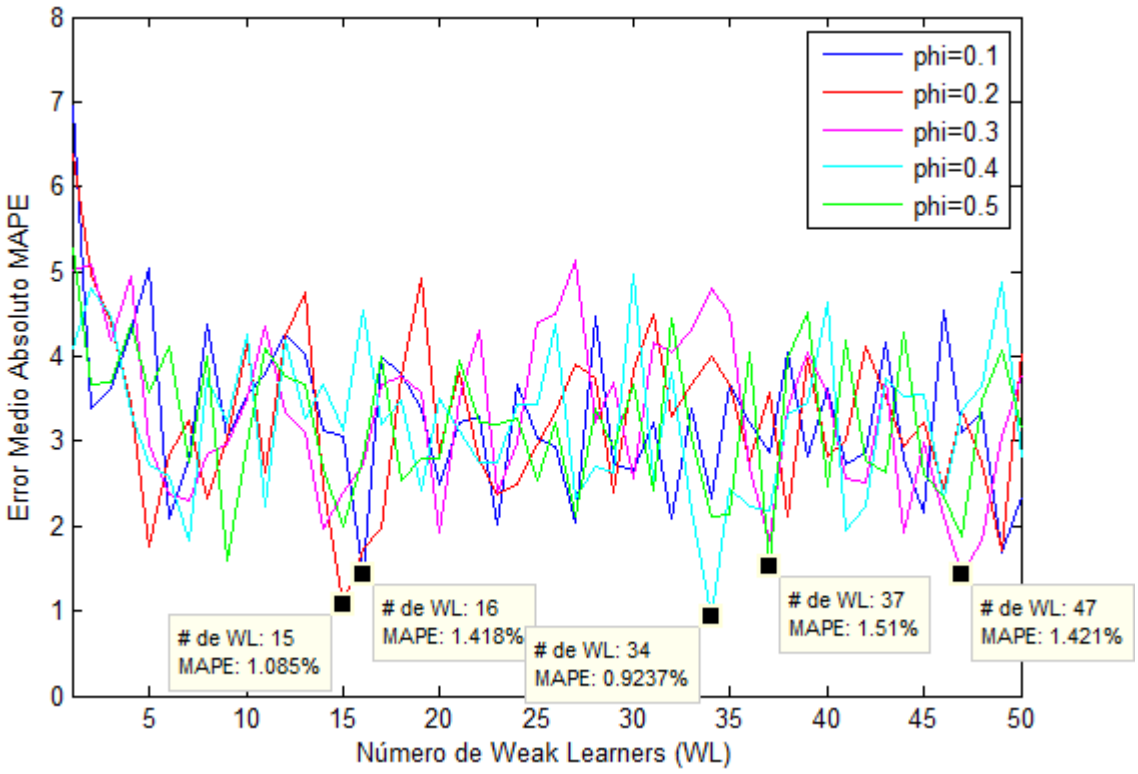


Figura 4.21 Error de AdaBoost.RT con variación de $\varphi_{inicial}$ para control con ruido 5%

4.3.2.3 Variación del parámetro r

El último parámetro que es posible variar y que puede depender del problema es r, un factor de ajuste de la tasa de cambio del RMSE usado para la actualización de pesos de las muestras en el algoritmo AdaBoost.RT que se presenta en (8). En [38], se sugiere $r = 0.5$ pero de igual manera se indica que puede ser seleccionado dependiendo de la aplicación.

Para este problema se decide realizar pruebas con $r = 0.3, 0.5$ y 0.7 para ver el efecto de este en el algoritmo, para cada caso se varió la cantidad de aprendices débiles de 1 a 50 y se realizaron 200 pruebas donde se toma el mejor modelo conjunto de cada una de las variaciones y se escoge el valor óptimo de r para el problema a resolver.

- **Ruido 0.1%**

En la Figura 4.22 se presenta el error medio absoluto MAPE de los mejores modelos conjuntos, donde no se observa relación entre el valor de r y el error o el error y el aumento del número de aprendices débiles que conforman el modelo conjunto. El mejor modelo conjunto se presentó con $r = 0.5$ con un error de 1.015%. En la Tabla 4.13 se presenta el resultado del mejor modelo conjunto para cada r.

Tabla 4.13 Mejores modelos conjuntos de control con variación de r con ruido 0.1%

Parámetro	# Aprendices débiles	MAPE (%)
$r = 0.3$	17	1.168
$r = 0.5$	39	1.015
$r = 0.7$	31	1.249

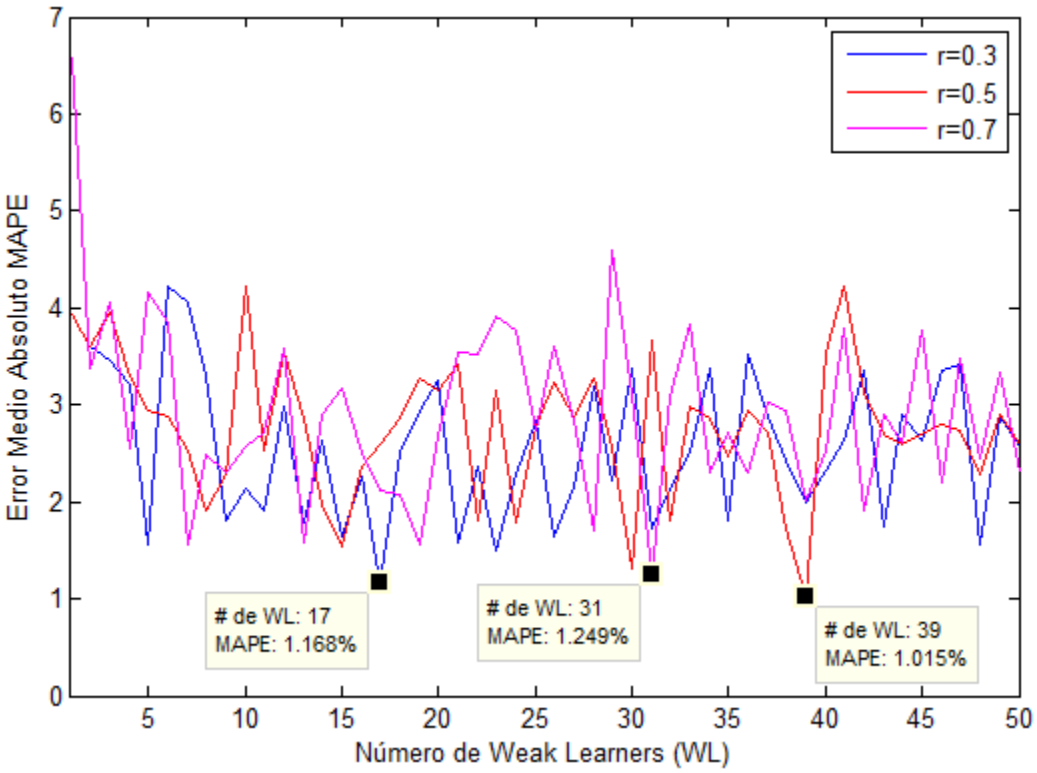


Figura 4.22 Error de AdaBoost.RT con variación de r para control con ruido 0.1%

- **Ruido 1%**

En la Figura 4.23 se presenta el error medio absoluto MAPE de los mejores modelos conjuntos, donde no se observa relación entre el valor de r y el error o el error y el aumento del número de aprendices débiles que conforman el modelo conjunto. El mejor modelo conjunto se presentó con r = 0.5 con un error de 0.583%. En la Tabla 4.14 se presenta el resultado del mejor modelo conjunto para cada r.

Tabla 4.14 Mejores modelos conjuntos de control con variación de r con ruido 1%

Parámetro	# Aprendices débiles	MAPE (%)
r = 0.3	45	0.709
r = 0.5	34	0.583
r = 0.7	19	1.345

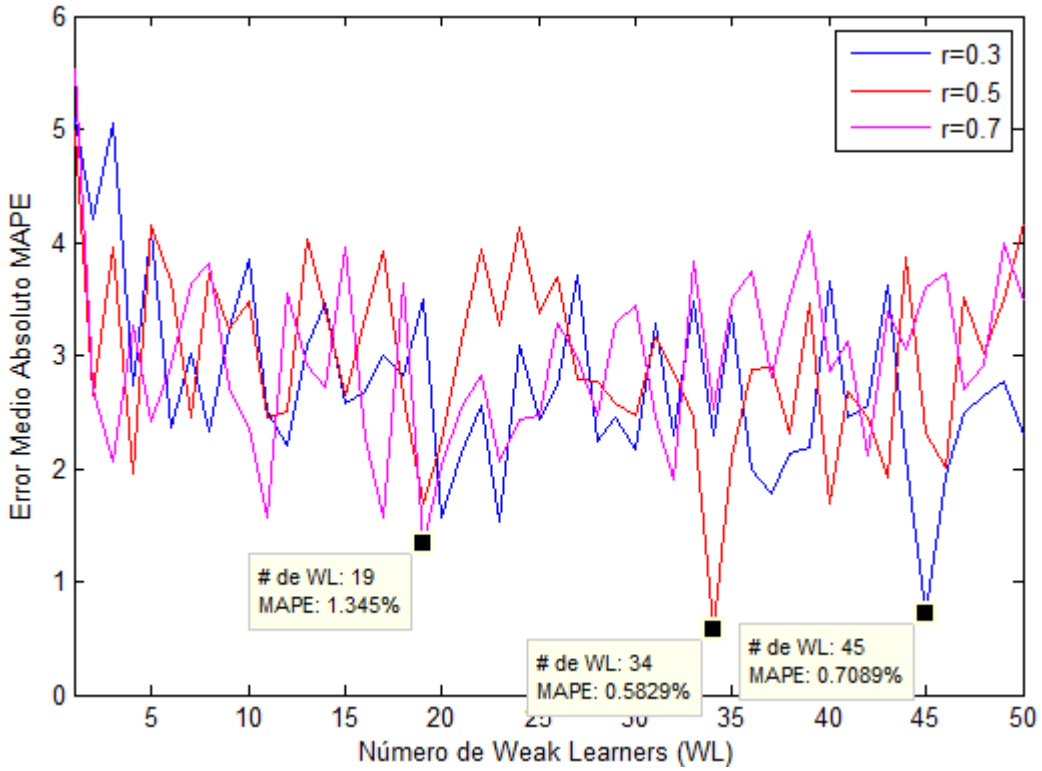


Figura 4.23 Error de AdaBoost.RT con variación de r para control con ruido 1%

- **Ruido 5%**

En la Figura 4.24 se presenta el error medio absoluto MAPE de los mejores modelos conjuntos, donde no se observa relación entre el valor de r y el error o el error y el aumento del número de aprendices débiles que conforman el modelo conjunto. El mejor modelo conjunto se presentó con r = 0.5 con un error de 0.923%. En la Tabla 4.15 se presenta el resultado del mejor modelo conjunto para cada r.

Tabla 4.15 Mejores modelos conjuntos de control con variación de r con ruido 5%

Parámetro	# Aprendices débiles	MAPE (%)
r = 0.3	36	1.372
r = 0.5	34	0.923
r = 0.7	41	1.493

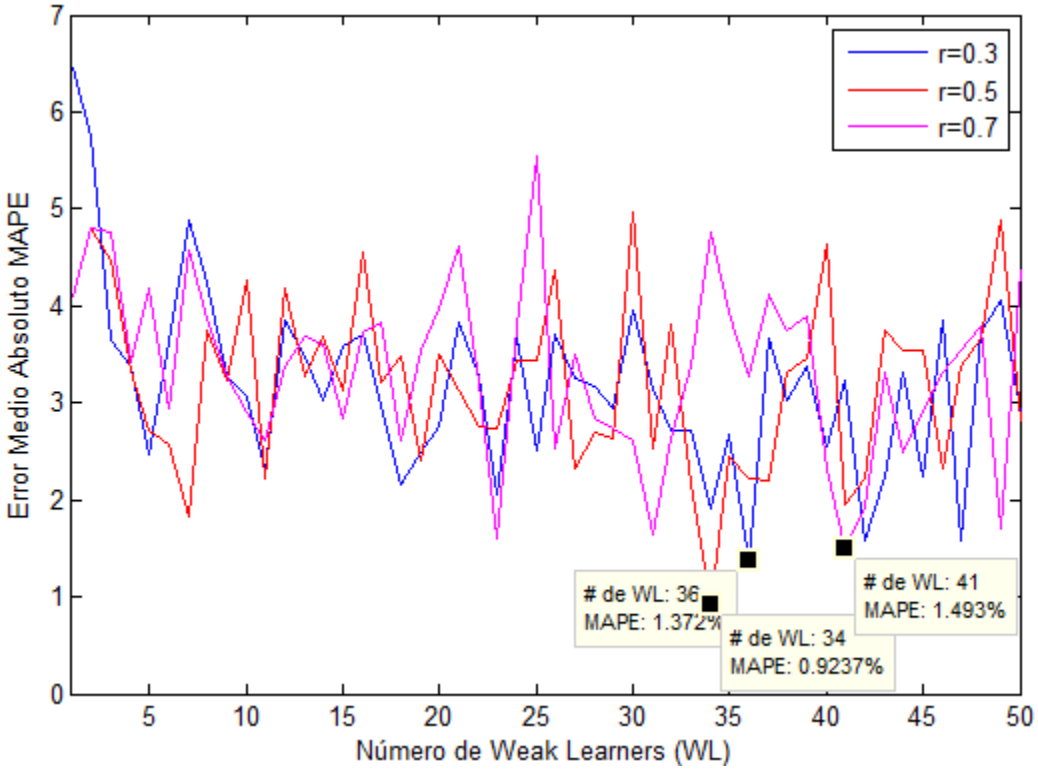


Figura 4.24 Error de AdaBoost.RT con variación de r para control con ruido 5%

4.3.2.4 Modelo óptimo de control del índice COLCAP con AdaBoost.RT con φ auto-adaptativo

Luego de analizar los resultados de las variaciones de los parámetros β, φ_{inicial} y r, se determinó el mejor modelo conjunto para el control del sistema del péndulo invertido.

- **Ruido 0.1%**

El modelo conjunto óptimo para el control del péndulo invertido entrenado con la base de datos con ruido del 0.1% se obtuvo con $\beta = error$, es decir, $n = 1$, $\varphi_{inicial} = 0.4$ y $r = 0.5$.

En la Figura 4.25 se presenta la comparación entre el controlador original o "profesor" y el controlador diseñado con base en AdaBoost y base de entrenamiento con ruido del 0.1%. El Overshoot del controlador de referencia está en un rango de $-1.716°$ a $1.604°$, mientras el Overshoot del controlador AdaBoost está en un rango de $-2.103°$ a $1.604°$. En cuanto al rango de estabilización, los dos controladores tienen prácticamente el mismo rango estable que se encuentra entre $-0.236°$ a $0.149°$. Estos rangos se presentan más claramente en la Tabla 4.16

Tabla 4.16 Parámetros de desempeño del controlador basado en AdaBoost con ruido 0.1%

	Controlador de refencia"	Controlador AdaBoost
Overshoot	$-1.716°$ a $1.604°$	$-2.103°$ a $1.604°$
Rango Estabilización	$-0.236°$ a $0.149°$	$-0.236°$ a $0.149°$.

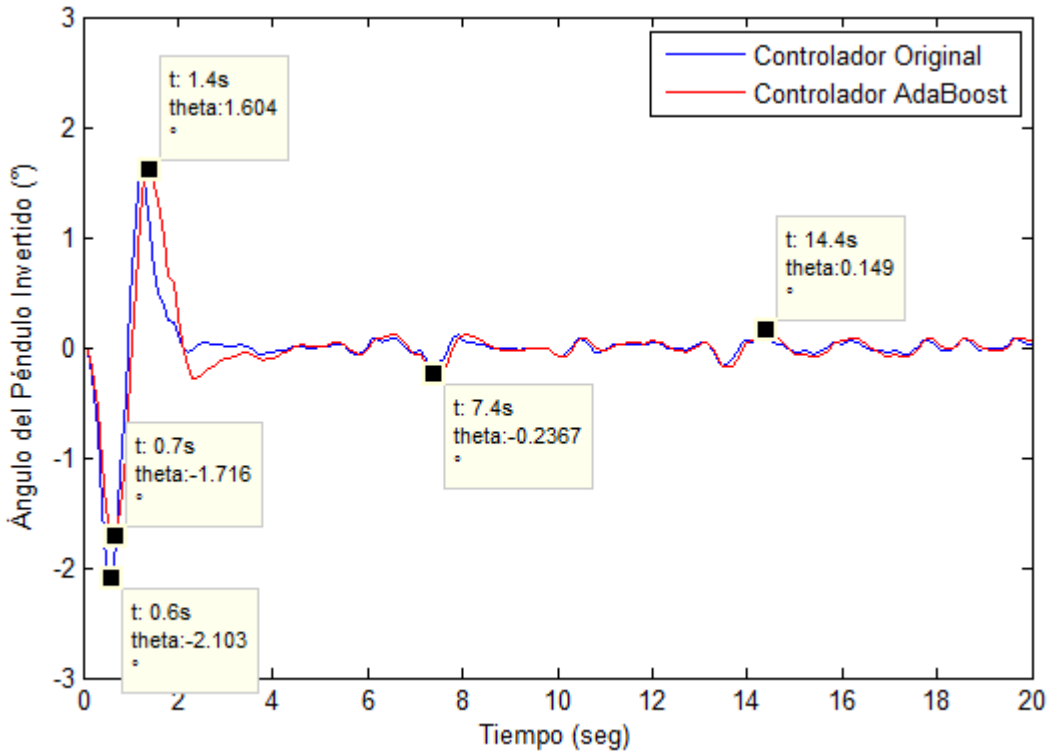


Figura 4.25 Desempeño del controlador diseñado por AdaBoost con ruido 0.1%

- **Ruido 1%**

El modelo conjunto óptimo para el control del péndulo invertido entrenado con la base de datos con ruido 1% se obtuvo con $\beta = error^3$, es decir, $n = 3$, $\varphi_{inicial} = 0.2$ y $r = 0.5$.

En la Figura 4.26 se presenta la comparación entre el controlador original o "profesor" y el controlador diseñado con base en AdaBoost y base de entrenamiento con ruido del 1%. El Overshoot del controlador de referencia está en un rango de $-1.666°$ a $1.309°$, mientras el Overshoot del controlador AdaBoost está en un rango de $-1.666°$ a $1.535°$. En cuanto al rango de estabilización, los dos controladores tienen prácticamente el mismo rango estable que se encuentra entre $-0.683°$ a $0.458°$. Estos rangos se presentan más claramente en la Tabla 4.17

Tabla 4.17 Parámetros de desempeño del controlador basado en AdaBoost con ruido 1%

	Controlador de referencia	Controlador AdaBoost
Overshoot	$-1.666°$ a $1.309°$	$-1.666°$ a $1.535°$
Rango Estabilización	$-0.683°$ a $0.458°$	$-0.683°$ a $0.458°$

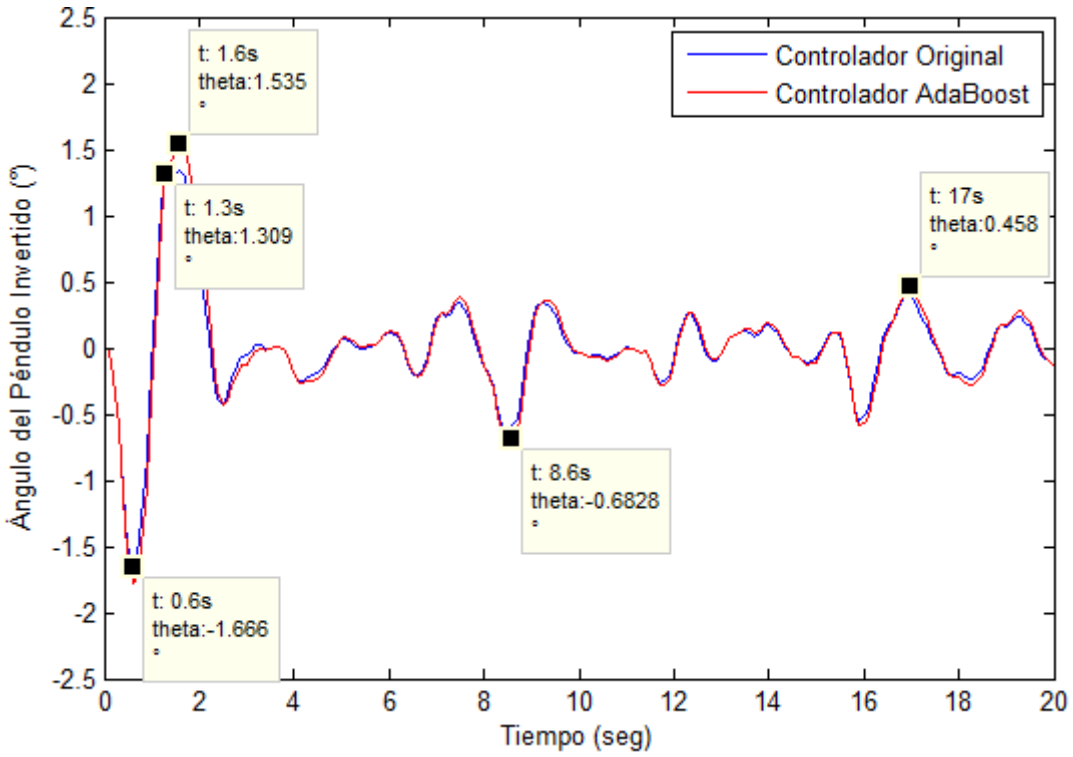


Figura 4.26. Desempeño del controlador diseñado por AdaBoost con ruido 1%

- **Ruido 5%**

El modelo conjunto óptimo para el control del péndulo invertido entrenado con la base de datos con ruido 5% se obtuvo con $\beta = error^2$, es decir, $n = 2$, $\varphi_{inicial} = 0.1$ y $r = 0.5$.

En la Figura 4.27 se pres

enta la comparación entre el controlador de referencia y el controlador diseñado con base en AdaBoost y base de entrenamiento con ruido del 5%. El Overshoot del controlador "profesor" está en un rango de $-1.846°$ a $2.033°$, mientras el Overshoot del controlador AdaBoost está en un rango de $-2.073°$ a $2.296°$. En cuanto al rango de estabilización, los dos controladores difieren muy poco, el rango estable del controlador de referencia se encuentra entre $-1.571°$ a $0.7929°$ y para el caso del controlador AdaBoost entre aún asi el rango tienen prácticamente el mismo rango estable que se encuentra entre $-1.839°$ a $1.015°$. Estos rangos se presentan más claramente en la Tabla 4.18

Tabla 4.18 Parámetros de desempeño del controlador basado en AdaBoost con ruido 5%

	Controlador de referencia	Controlador AdaBoost
Overshoot	$-1.846°$ a $2.033°$	$-2.073°$ a $2.296°$
Rango Estabilización	$-1.571°$ a $0.7929°$	$-1.839°$ a $1.015°$

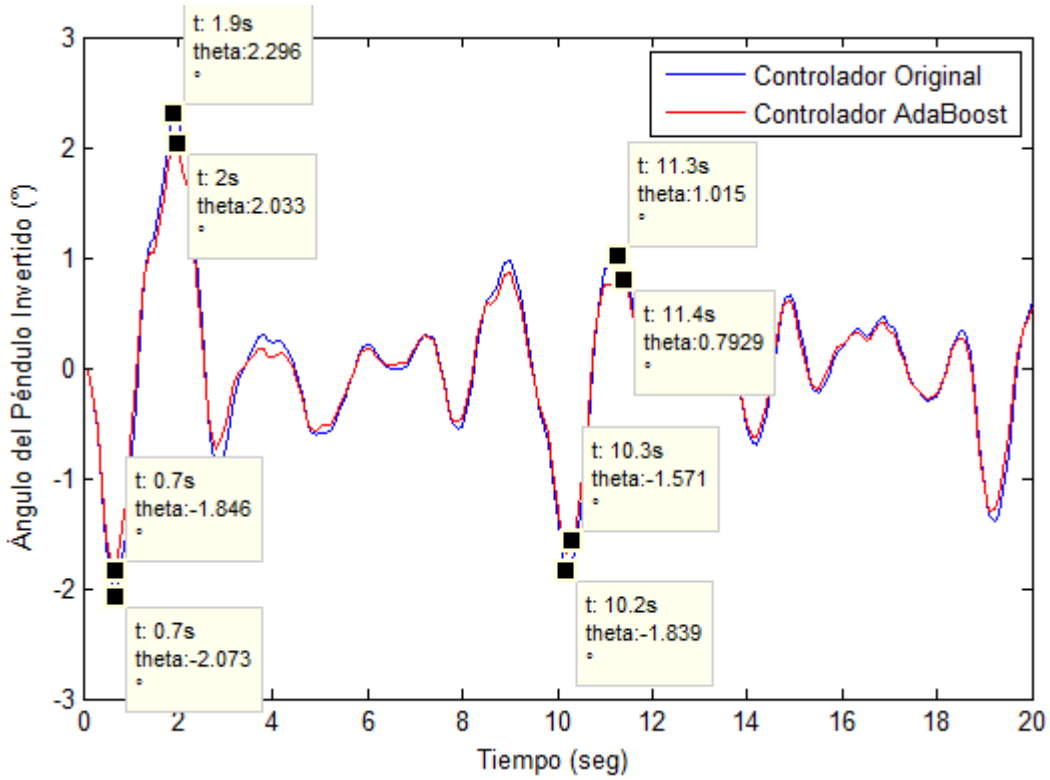


Figura 4.27. Desempeño del controlador diseñado por AdaBoost con ruido 5%

5. ANÁLISIS DE RESULTADOS

En este capítulo se presenta el análisis de los resultados obtenidos en los dos capítulos anteriores y se realiza una comparación de los modelos óptimos obtenidos en este proyecto con resultados obtenidos en la literatura para validar los resultados del proyecto.

5.1 ANÁLISIS DE RESULTADOS

5.1.1 Predicción Índice COLCAP

La predicción del índice COLCAP se llevó a cabo por medio de Redes Neuronales y AdaBoost.RT con φ auto-adaptativo. Los resultados de la predicción del índice por Redes Neuronales se presentan en la sección 3.5.1 y el modelo óptimo basado en AdaBoost.RT se presenta en la sección 3.5.2.4. A partir de los resultados presentados en las secciones se puede observar que el modelo con redes neuronales presenta un error del 1.206% con 9 neuronas en la capa oculta mientras el modelo conjunto de AdaBoost.RT presenta un error del 1.247% y cuenta con 48 aprendices débiles. En la *Tabla 5.1* se presentan los parámetros de los modelos de predicción. En la *Figura 5.1* se presenta la comparación de la serie original, la predicción realizada por redes neuronales y la realizada con AdaBoost.RT.

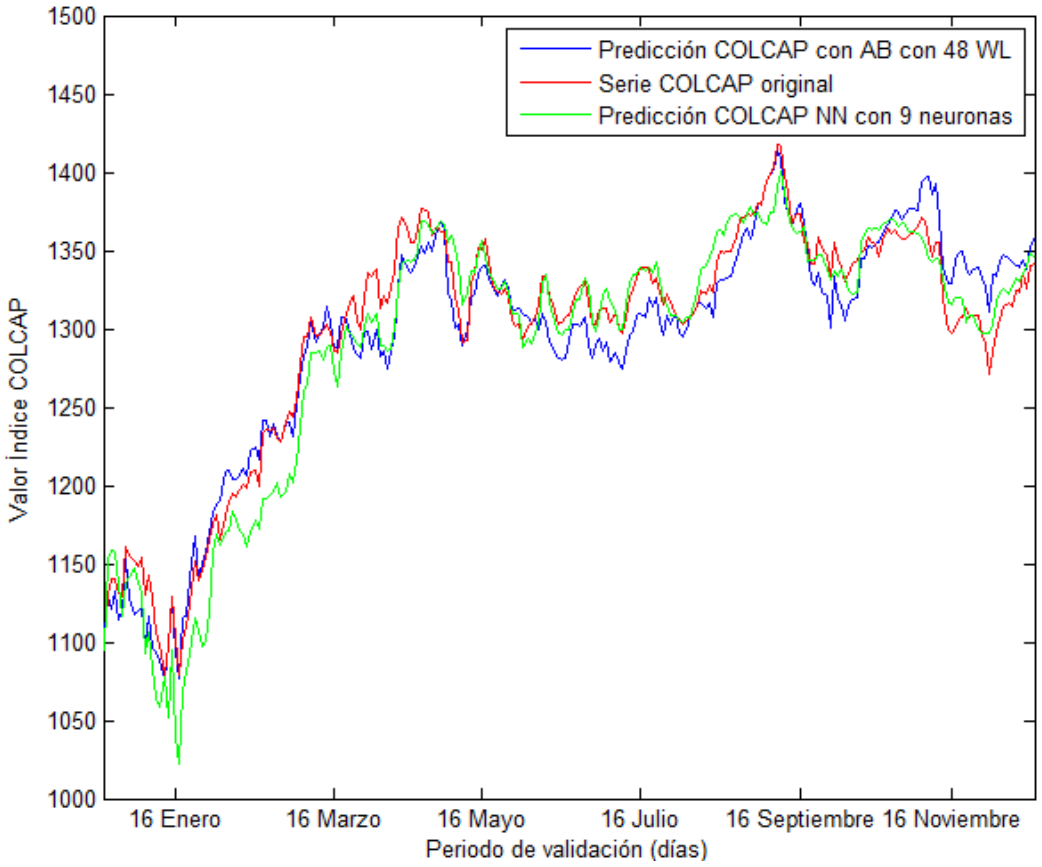


Figura 5.1 Comparación serie original y predicciones realizadas por medio de redes neuronales y AdaBoost.RT

En la *Figura 5.1* se puede observar que hasta Marzo, la predicción con AdaBoost sigue con mayor exactitud a la serie original, luego aproximadamente hasta Mayo las dos predicciones tienen un comportamiento similar al de la serie original pero no son exactas y a partir de Mayo la predicción con Redes Neuronales sigue mejor a la serie original que la realizada con AdaBoost.

Tabla 5.1 Parámetros de los modelos de predicción

Modelo	MAPE	RMSE	Coeficiente de Correlación	Complejidad
AdaBoost	1.247%	19.87	0.97	48 WL
Neuronal	1.206%	19.91	0.99	9 neuronas

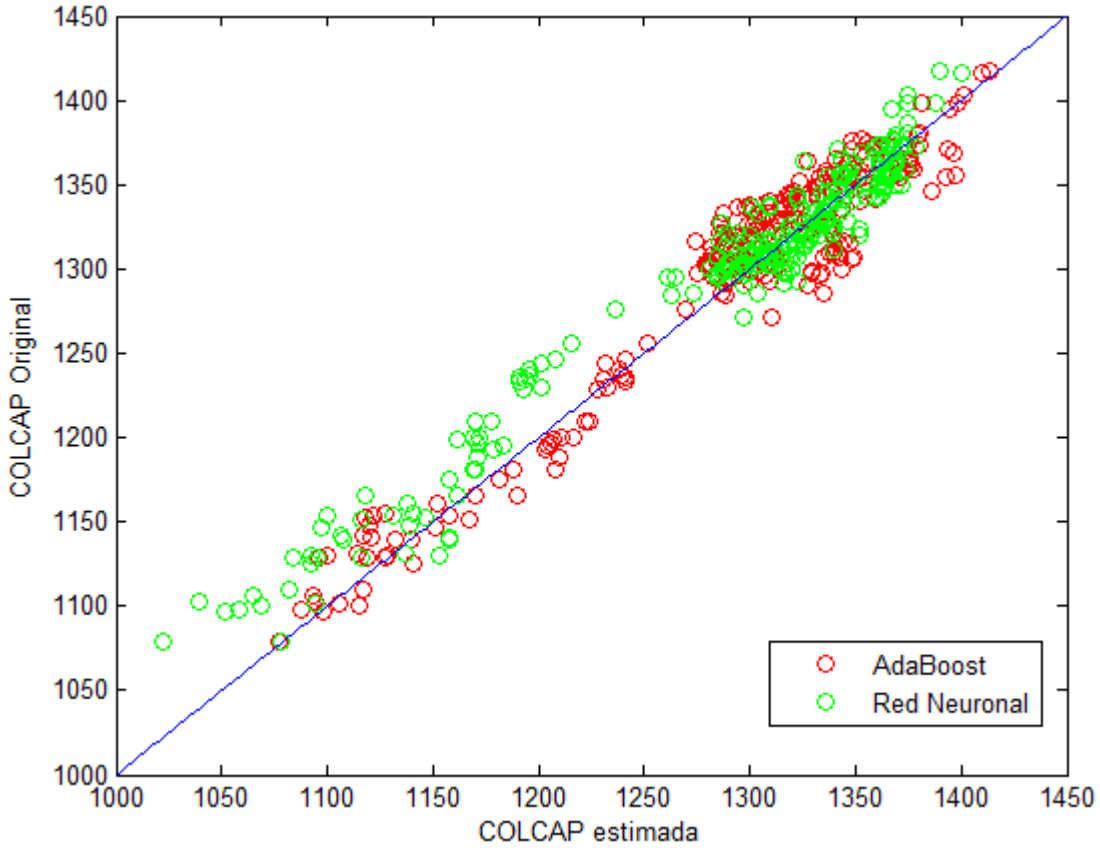


Figura 5.2 Diagrama de Dispersión de las estimaciones neuronal y AdaBoost de la serie COLCAP

En la *Figura 5.2* se presenta el diagrama de dispersión de la serie COLCAP estimada por medio de AdaBoost y Redes neuronales, en donde se puede observar la relación que se presenta entre la serie estimada y la serie original, de tal manera que entre mayor sea esta, más cerca se encuentran los respectivos puntos a la línea azul. A partir de los criterios de desempeño presentados en la *Tabla 5.1* y de la Figura 5.1 se puede determinar que el modelo neuronal posee mejor desempeño que el modelo conjunto AdaBoost.

Además con respecto a la variación del error máximo, promedio y mínimo, que en el caso de redes neuronales se encuentra en la Figura 3.5 se puede observar que los errores promedios que se encuentra cerca del valor mínimo y tienen un comportamiento en el cual disminuye hasta llegar al

error promedio mínimo y luego empieza a ascender a medida que se aumentan las neuronas de la capa oculta, en el caso de AdaBoost, que se presentan en las Figuras Figura 3.8, Figura 3.10, Figura 3.12,Figura 3.15, Figura 3.17, Figura 3.19, Figura 3.21, Figura 3.23, Figura 3.26, Figura 3.28 y Figura 3.30 se observa que el modelo es altamente variable puesto que el error promedio tiene un comportamiento que no tiene relación con la cantidad de número de aprendices débiles y el valor promedio se encuentra aproximadamente en la mitad o más cercano al error máximo que al mínimo, lo que muestra que los modelos generados a partir de AdaBoost.RT dependen de las condiciones iniciales del experimento, las cuales son determinadas aleatoriamente.

Los resultados de la predicción realizada por medio de AdaBoost se presentan en la sección 3.5.2 en donde se varió la actualización de los pesos de las muestras correctas β_t, el valor inicial de φ y el parámetro r.

A partir de los resultados de la variación de los pesos de las muestras correctas β_t presentados en la Tabla 3.11 se observó que el mejor desempeño para este problema se presentó con $\beta = error$ (lineal), aunque los modelos diseñados con los otros dos tipos de actualización (cuadrada y cuadrática) tienen error de predicción menor al 2% lo que se considera un buen desempeño de predicción. La mayor diferencia presente entre los métodos de actualización se basa en la cantidad de aprendices débiles del modelo conjunto, dado que el método de actualización que presenta menor error posee dos aprendices débiles y el modelo que le sigue presenta un aumento del error menor al 0.1% con 11 aprendices débiles.

En cuanto a los resultados de la variación del valor inicial de φ presentados en la Tabla 3.12 se observó que el mejor modelo conjunto se obtuvo con $\varphi_{inicial} = 0.4$ y tiene un error del 1.247%. Al igual que en el caso anterior los demás modelos no presentan un error mayor al 2%. No se muestra relación entre el valor de φ inicial y la cantidad de aprendices débiles del modelo, puesto que el modelo con mejor desempeño tiene 48 aprendices débiles mientras el modelo con el segundo mejor desempeño posee tan sólo 2 aprendices débiles. En este caso, se considera que el mejor modelo es el que presenta menor error aunque tenga mayor complejidad dado que no se tiene en cuenta como criterio de selección del mejor modelo la complejidad del mismo, en caso de ser necesario implementar el modelo en un sistema físico o se busque un modelo sencillo con buen desempeño, el modelo con 2 aprendices débiles puede ser escogido como el mejor, dado que no incurre en una disminución significativa del desempeño (menor al 0.2%).

Adicionalmente, con base en los resultados de la variación del parámetro r presentados en la Tabla 3.13 no se observa relación entre el parámetro r y el desempeño de los modelos. El mejor modelo se obtuvo con $r = 0.5$, de la misma manera que en los casos anteriores los demás modelos no poseen un error superior al 2%. No se observa relación entre el número de aprendices débiles y el valor de r, debido a que el mejor modelo consta de 48 aprendices débiles mientras el modelo que le sigue en desempeño entrenado con $r = 0.3$ consta de 14 aprendices y la disminución del desempeño es menor al 0.1%, de tal manera que en caso de tener en cuenta la complejidad al escoger el mejor modelo, el obtenido con $r = 0.3$ se consideraría el mejor debido a su baja complejidad y desempeño.

Debido a lo mostrado anteriormente, se hace necesario llevar a cabo la experimentación realizada puesto que el espacio de búsqueda no es simple y no es posible establecer de manera a priori los parámetros del algoritmo $(\beta,\ \varphi_{inicial}, r)$ como una condición única y particular en el momento de inicializar AdaBoost.

5.1.2 Control del sistema del péndulo Invertido

- **Ruido 0.1%**

Para evaluar el desempeño de los controladores diseñados se aumenta el ruido introducido al sistema en un 100% con respecto al original, es decir, la potencia del ruido es del 0.2%. En la Figura 5.3 se presenta la comparación y en la En comparación con los resultados obtenidos con la base de datos original contaminada con ruido del 0.1% presentados en la Tabla 4.4 en el caso de redes neuronas y Tabla 4.16 para AdaBoost se puede observar que aumentó el Overshoot para ambos casos en un 8.22% y el rango de estabilización para redes neuronales aumentó en un 52.18% y para AdaBoost en un 96.66%, a pesar del aumento en los rangos el sistema se considera estable dado que se mantiene en el rango de -5° a 5°, sin embargo se observa que el controlador neuronal para este caso tiene más robustez puesto que presenta un porcentaje menor de variación de rango de estabilización ante la variación del ruido del sistema con respecto al modelo basado en AdaBoost además la complejidad del controlador neuronal es menor a la del modelo basado en AdaBoost.

se presenta el desempeño de los controladores diseñados con el aumento de la potencia mencionada anteriormente. Se observa que aunque se aumentó la potencia del ruido en un 100%, los controladores diseñados logran estabilizar el sistema en un rango de $-0.464°$ a 0.562. Además, en la Figura 5.4 se presenta la gráfica de dispersión del controlador de referencia versus los controladores entrenados por medio de AdaBoost y red neuronal, se puede observar que el controlador neuronal no presenta dispersión, lo que se refuerza con el coeficiente de correlación de este controlador que es de 1, en el caso de controlador basado en AdaBoost se observa que se presenta dispersión principalmente en los extremos pero en los puntos cercanos a cero, que son los correspondientes al estado de estabilización se observa que la dispersión es menor.

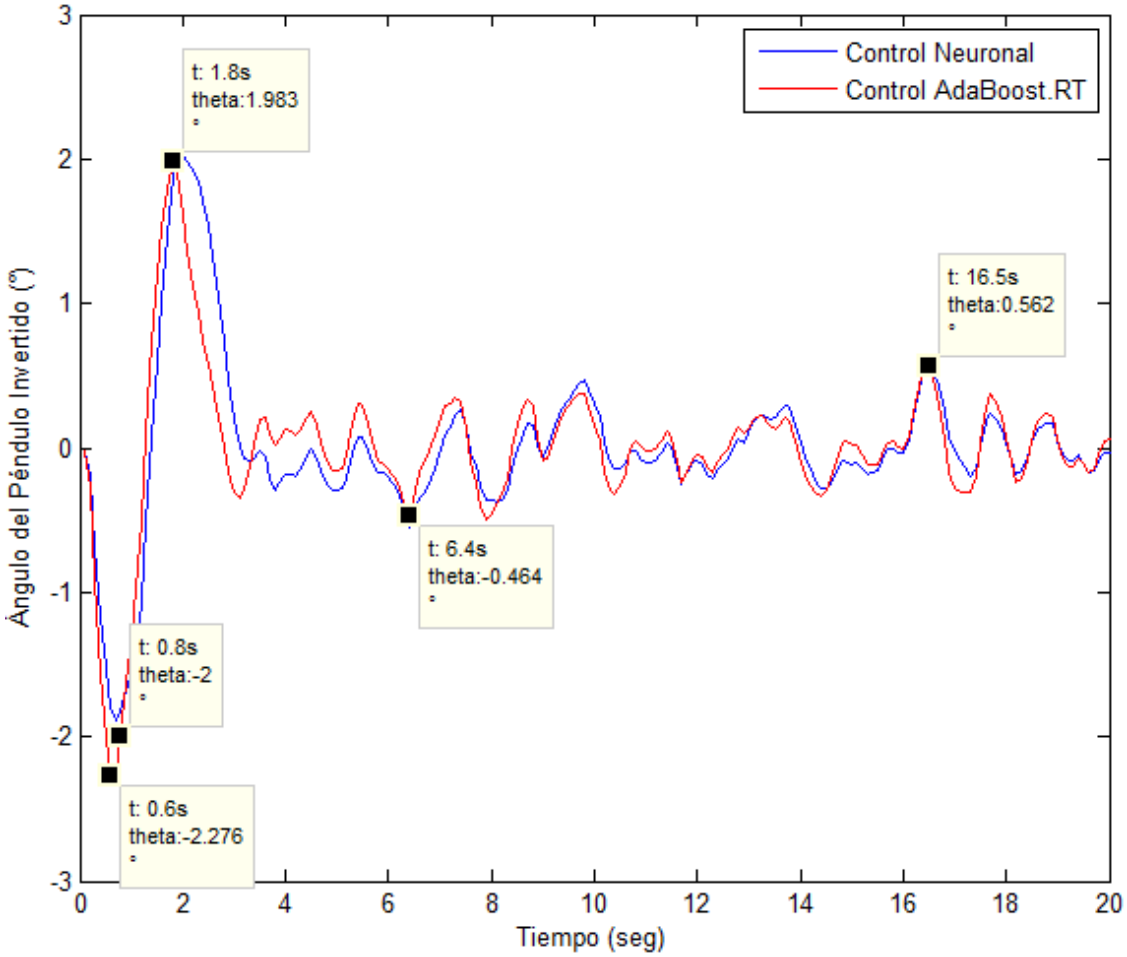


Figura 5.3 Comparación controlador neuronal y AdaBoost.RT ruido 0.2%

En comparación con los resultados obtenidos con la base de datos original contaminada con ruido del 0.1% presentados en la Tabla 4.4 en el caso de redes neuronas y Tabla 4.16 para AdaBoost se puede observar que aumentó el Overshoot para ambos casos en un 8.22% y el rango de estabilización para redes neuronales aumentó en un 52.18% y para AdaBoost en un 96.66%, a pesar del aumento en los rangos el sistema se considera estable dado que se mantiene en el rango de -5° a 5°, sin embargo se observa que el controlador neuronal para este caso tiene más robustez puesto que presenta un porcentaje menor de variación de rango de estabilización ante la variación del ruido del sistema con respecto al modelo basado en AdaBoost además la complejidad del controlador neuronal es menor a la del modelo basado en AdaBoost.

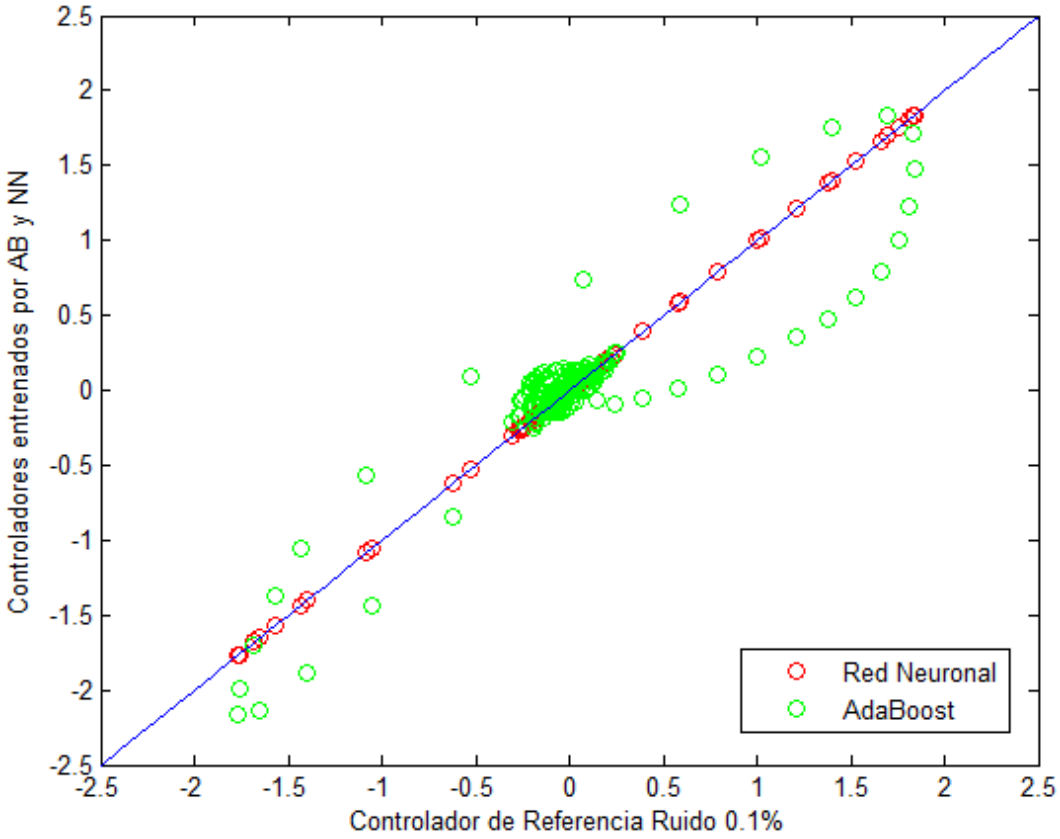


Figura 5.4 Gráfica de dispersión de los controladores entrenados por medio de AdaBoost y Redes Neuronales para ruido del 0.1%.

Tabla 5.2 Desempeño de los controladores diseñados con ruido del 0.1% con aumento del ruido del 100%

	Controlador Neuronal	Controlador AdaBoost
Overshoot	$-2°$ a $1.983°$	$-2.276°$ a $1.983°$
% Variación OV	8.22%	8.22%
Rango Estabilización	$-0.464°$ a $0.562°$	$-0.464°$ a $0.562°$
% Variación RE	52.18%	96.66%
Coeficiente de correlación	1	0.898
Complejidad	11 neuronas	39 aprendices débiles

- **Ruido 1%**

Para evaluar el desempeño de los controladores diseñados se aumenta el ruido introducido al sistema en un 100% con respecto al original, es decir, la potencia del ruido es del 2%. En la Figura 5.5 se presenta la comparación y en la Tabla 5.3 el desempeño de los controladores diseñados con el aumento de la potencia mencionada anteriormente. Se observa que aunque se aumentó la potencia del ruido en un 100%, los controladores diseñados logran estabilizar el sistema en un rango de $-0.9537°$ a $1.096°$. Además, en la Figura 5.6 se presenta la gráfica de dispersión del controlador de referencia versus los controladores entrenados por medio de AdaBoost y red neuronal, se puede observar que el controlador neuronal no presenta dispersión, lo que se refuerza con el coeficiente de correlación de

este controlador que es de 1, en el caso de controlador basado en AdaBoost se observa que se presenta dispersión principalmente en los extremos pero en los puntos cercanos a cero, que son los correspondientes al estado de estabilización se observa que la dispersión es menor.

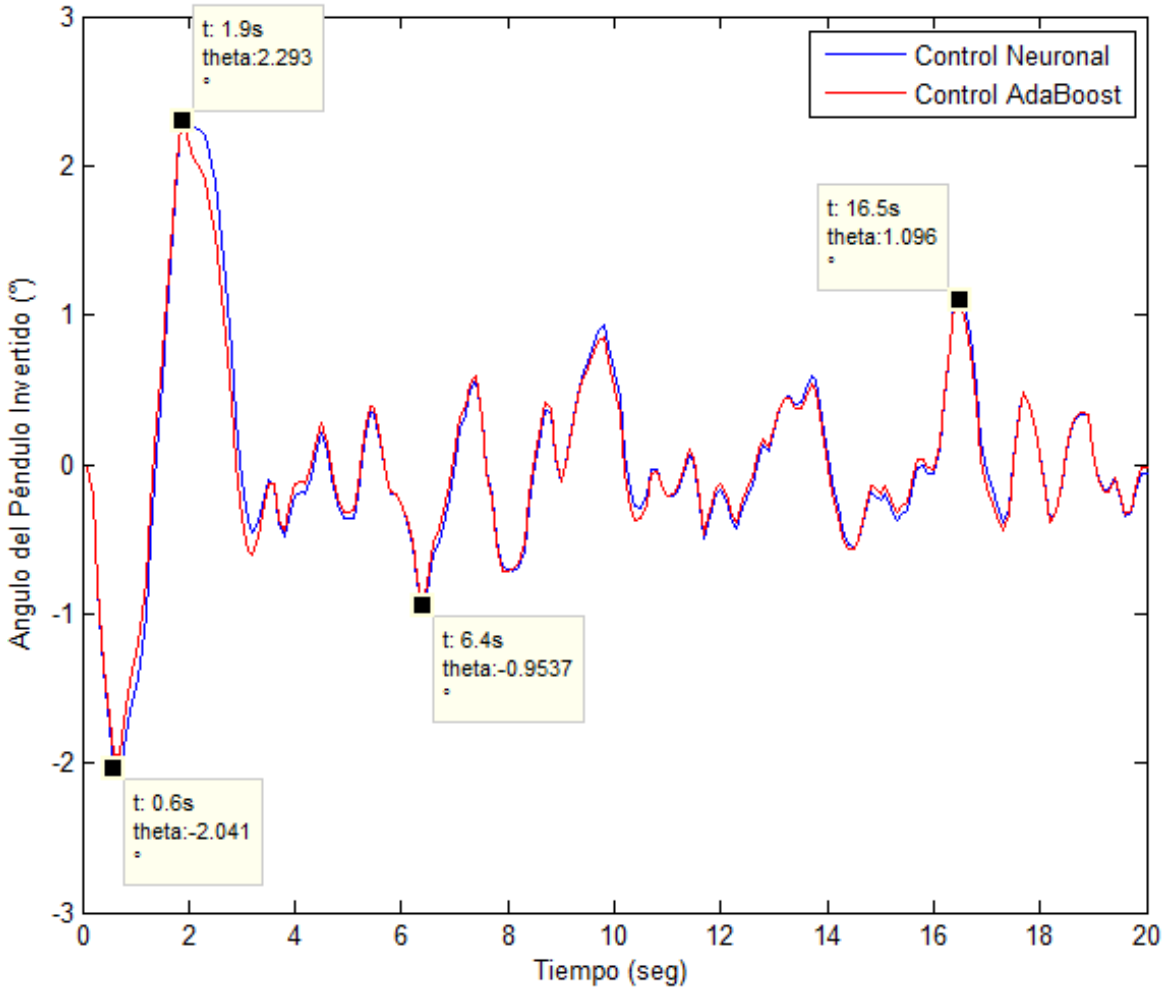


Figura 5.5 Comparación controlador neuronal y AdaBoost.RT ruido 2%

En comparación con los resultados obtenidos con la base de datos original contaminada con ruido del 1% presentados en la Tabla 4.5 en el caso de redes neuronales y Tabla 4.17 para AdaBoost se puede observar que aumentó el rango del Overshoot en un 7.149% y 38.132% respectivamente y el rango de estabilización para redes neuronales aumentó en un 36.471% y para AdaBoost en un 60.468% con respecto a los obtenidos sin el aumento de ruido. A pesar del aumento en los rangos el sistema se considera estable el sistema dado que se mantiene en el rango de -5° a 5°, sin embargo se observa que el controlador neuronal para este caso tiene más robustez puesto que presenta un porcentaje menor de variación tanto de Overshoot como rango de estabilización ante la variación del ruido del sistema con respecto al modelo basado en AdaBoost, además el controlador neuronal es un modelo con menor complejidad al modelo basado en AdaBoost.

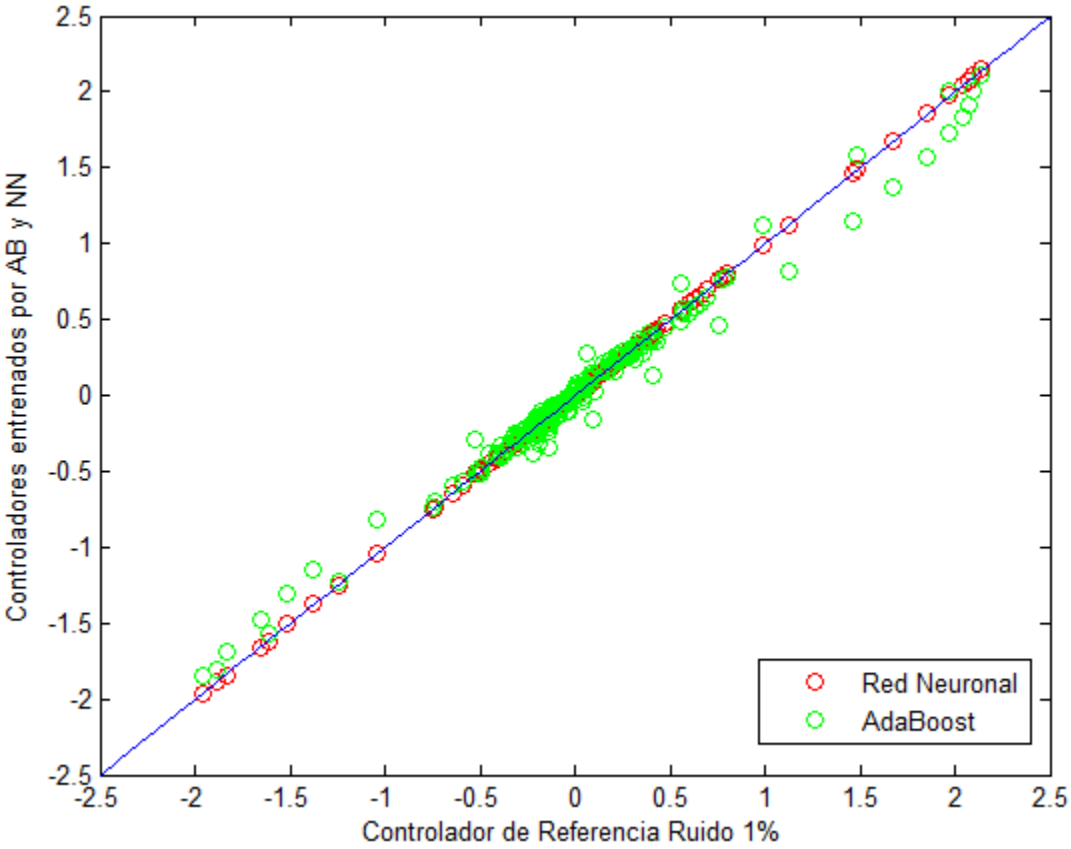


Figura 5.6 Gráfica de dispersión de los controladores entrenados por medio de AdaBoost y Redes Neuronales para ruido del 1%.

Tabla 5.3 Desempeño de los controladores diseñados con ruido 1% con aumento del ruido del 100%

	Controlador Neuronal	Controlador AdaBoost
Overshoot	$-2.041°$ a $2.293°$	$-2.041°$ a $2.293°$
% Variación OV	7.149%	38.132%
Rango Estabilización	$-0.9537°$ a $1.096°$	$-0.9537°$ a $1.096°$
% Variación RE	36.471%	60.468%
Coeficiente de correlación	1	0.993
Complejidad	5 neuronas	34 aprendices débiles

- **Ruido 5%**

Para evaluar el desempeño de los controladores diseñados se aumenta el ruido introducido al sistema en un 100% con respecto al original, es decir, la potencia del ruido es del 5%. En la Figura 5.7 se presenta la comparación y en la Tabla 5.4 se presenta el desempeño de los controladores diseñados con el aumento de la potencia mencionada anteriormente. Se observa que aunque se aumentó la potencia del ruido en un 100%, los controladores diseñados logran estabilizar el sistema

en un rango de $-2.116°$ a $2.695°$. Además, en la *Figura 5.8* se presenta la gráfica de dispersión del controlador de referencia versus los controladores entrenados por medio de AdaBoost y red neuronal, se puede observar que el controlador neuronal no presenta dispersión, lo que se refuerza con el coeficiente de correlación de este controlador que es de 1, en el caso de controlador basado en AdaBoost se observa que se presenta dispersión principalmente en los extremos pero en los puntos cercanos a cero, que son los correspondientes al estado de estabilización se observa que la dispersión es menor.

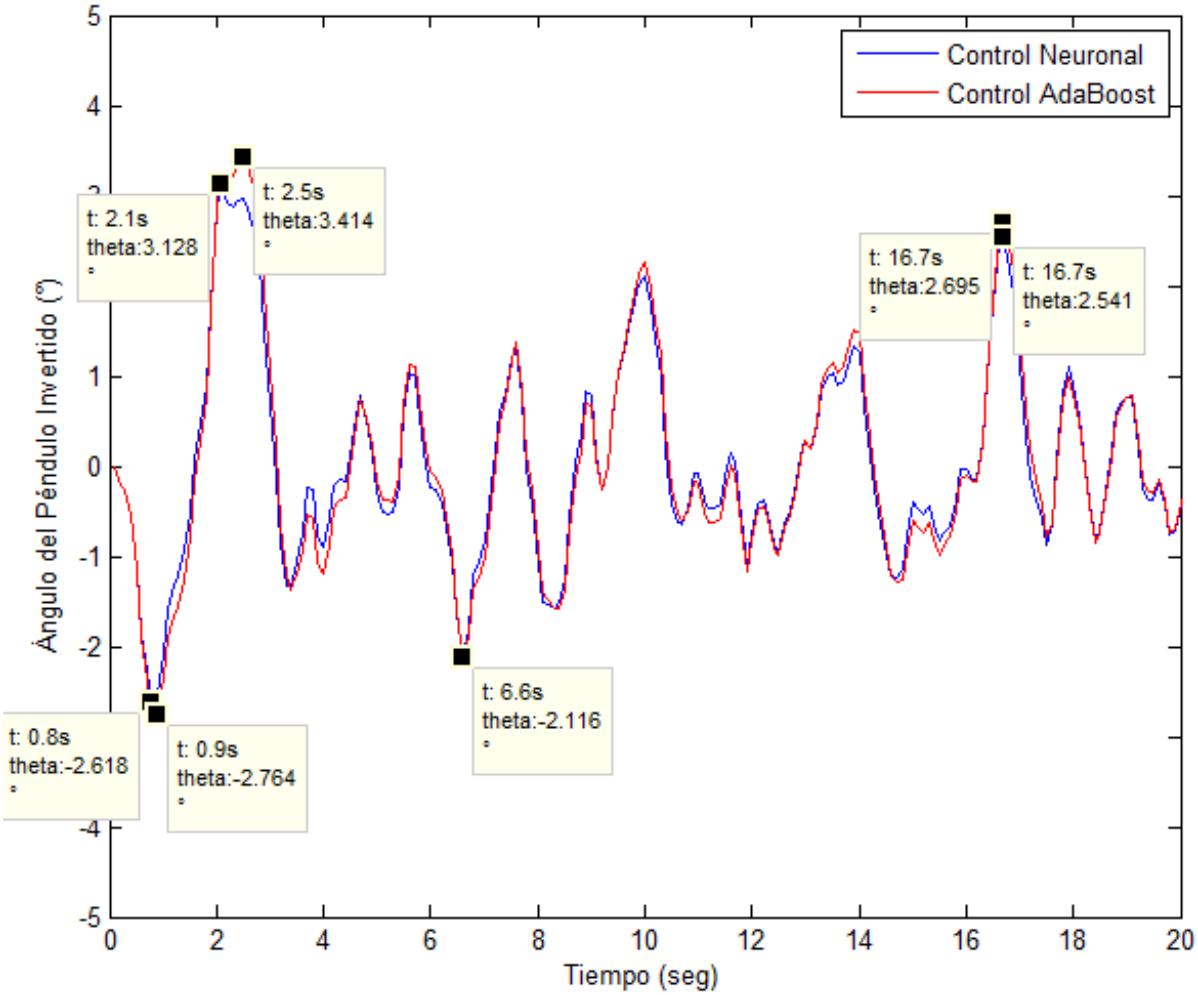


Figura 5.7 Comparación controlador neuronal y AdaBoost.RT ruido 5%

En comparación con los resultados obtenidos con la base de datos original contaminada con ruido del 5% presentado en la Tabla 4.6 en el caso de redes neuronales y Tabla 4.18 para AdaBoost se puede observar que aumentó el rango del Overshoot en un 53.86% y 48.69% respectivamente y el rango de estabilización, pero el sistema realimentado sigue estando dentro de un rango de estabilización para redes neuronales aumentó en un 61.74% y para AdaBoost en un 46.55% con respecto a los obtenidos sin el aumento de ruido. A pesar del aumento en los rangos el sistema se considera estable el sistema dado que se mantiene en el rango de -5° a 5°, sin embargo se observa que el controlador basado en AdaBoost para este caso tiene más robustez puesto que presenta un porcentaje menor de variación tanto de Overshoot como rango de estabilización ante la variación del ruido del sistema con respecto al modelo neuronal. A pesar de que el modelo más robusto sea el

basado en AdaBoost también acarrea mayor complejidad, por tanto, en caso de tener en cuenta la complejidad del modelo para la selección del controlador, se consideraría mejor modelo el controlador neuronal debido a que presenta menor complejidad.

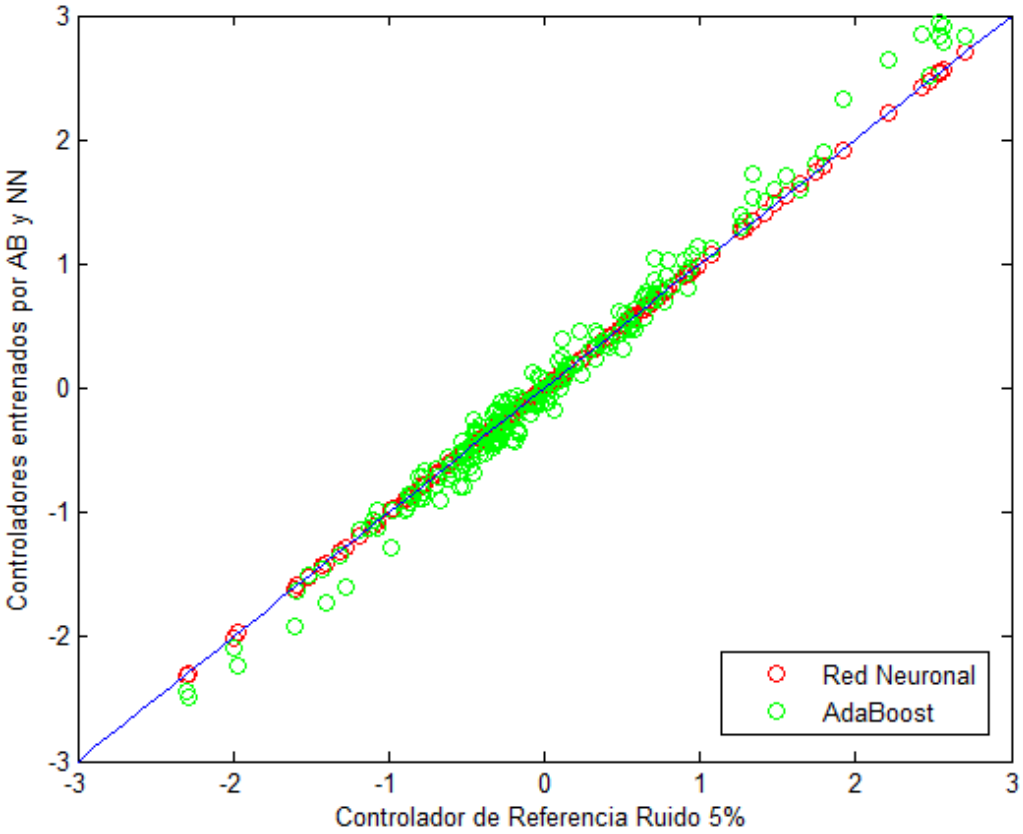


Figura 5.8 Gráfica de dispersión de los controladores entrenados por medio de AdaBoost y Redes Neuronales para ruido del 1%.

Tabla 5.4 Desempeño de los controladores diseñados con ruido 5% con aumento del ruido del 100%

	Controlador Neuronal	Controlador AdaBoost
Overshoot	$-2.618°$ a $3.128°$	$-2.764°$ a $3.414°$
% Variación OV	53.861%	48.69%
Rango Estabilización	$-2.116°$ a $2.541°$	$-2.116°$ a $2.695°$
% Variación RE	61.744%	46.547%
Coeficiente de Correlación	1	0.994
Complejidad	5 neuronas	34 aprendices débiles

5.1.2.1 Control del péndulo invertido por medio de AdaBoost.RT con φ auto-adaptativo

Los resultados del control realizado por medio de AdaBoost se presentan en la sección 4.3.2 , que se presenta dividida en la variación de los parámetros del algoritmo como lo son $\beta, \varphi_{inicial}$ y r, donde

cada uno de los parámetros se presenta dividido en el porcentaje de ruido introducido a la base de datos y luego de las pruebas se escogió el mejor modelo de control basado en AdaBoost para cada base de datos. En la Tabla 5.5 se presentan los parámetros de los modelos generados.

Tabla 5.5 Parámetros de los controladores basados en AdaBoost

Ruido	# WL	β	$\varphi_{inicial}$	r	Overshoot	Rango Estabilización
0.1%	39	$error^2$	0.1	0.5	$-2.103°$ a $1.604°$	$-0.236°$ a $0.149°$.
1%	34	$error^3$	0.2	0.5	$-1.666°$ a $1.535°$	$-0.683°$ a $0.458°$
5%	34	$error$	0.4	0.5	$-2.073°$ a $2.296°$	$-1.839°$ a $1.015°$

- **Variación del peso de las muestras correctas β**

Para el caso de ruido de 0.1% con variación de la actualización β los resultados se presentan en la Tabla 4.7 donde se puede observar que el mejor método de actualización para esta base de datos se encuentra con $\beta = \varepsilon^2$ (cuadrado), aunque los modelos diseñados con los otros tipos de actualización (lineal y cúbica) tienen un error de entrenamiento menor al 2% que se considera un buen desempeño de entrenamiento. Además el modelo obtenido con actualización cuadrada presenta 32 aprendices débiles, mientras los demás métodos de actualización generaron modelos con más de 40 aprendices débiles, lo que significa mayor complejidad del modelo.

En cuanto a la base de datos con ruido del 1% los resultados son presentados en la Tabla 4.8 se puede observar que el mejor método de actualización para esta base de datos se presenta con actualización cúbica con 34 aprendices débiles y aunque esta actualización presenta los mejores resultados, los modelos generados con los otros métodos de actualización (lineal y cuadrada) presentan un error de entrenamiento menor al 2%. En caso de incluir como criterio para escoger el mejor modelo la complejidad del mismo, se podría considerar como mejor modelo el entrenado con actualización lineal puesto que cuenta con 27 aprendices débiles, reduciendo la complejidad del sistema un 20.58%, y el aumento en el error de entrenamiento sería de 0.6% que dependiendo del desempeño deseado podría no ser relevante.

Finalmente, para la base de datos con ruido del 5% los resultados se presentan en la Tabla 4.9 se observa que los modelos generados sin importar el método de actualización obtuvieron un error menor al 2% lo que se considera un buen desempeño de entrenamiento. El mejor modelo fue obtenido con actualización lineal y posee 15 aprendices débiles. En caso de tener en cuenta la complejidad del sistema al escoger el mejor modelo, se podría considerar el modelo generado con actualización cúbica puesto que conlleva una disminución de 10 aprendices débiles (66.6%) y un aumento en el error de entrenamiento del 0.6%, lo cual dependiendo de la aplicación puede no ser significativo en el desempeño del controlador.

A partir de lo anterior se puede observar que el método de actualización de β depende de la aplicación que se lleve a cabo y aunque se aplique en la misma aplicación, como en el caso de este proyecto, al ser bases de datos diferentes de la misma aplicación varía el método de actualización óptimo. También se puede observar que el método de actualización que presenta el mejor desempeño no conlleva necesariamente al modelo con menor complejidad y que dependiendo del desempeño deseado se puede acarrear un aumento del error con una disminución de la complejidad.

- **Variación del valor inicial de φ**

Para la base de datos con ruido del 0.1% los resultados se presentan en la

Tabla *4.10* se puede observar que los modelos obtenidos variando el valor inicial de φ a excepción de $\varphi_{inicial} = 0.4$ presentan errores de entrenamiento menores al 2%. El mejor modelo se obtuvo con $\varphi_{inicial} = 0.1$ y posee 39 aprendices débiles. Para escoger el mejor modelo no se tuvo en cuenta la complejidad del mismo, pero en caso de tenerse en cuenta el modelo generado con $\varphi_{inicial} = 0.5$ se podría considerar como el mejor puesto que posee 15 aprendices débiles, es decir, conlleva a una reducción de la complejidad del modelo del 35.9% y el error de entrenamiento aumenta en menos del 0.3%, lo cual dependiendo de la aplicación puede no ser significativo en el desempeño del controlador.

En cuanto a la base de datos con ruido del 1% los resultados se presentan en la Tabla 4.11 donde se puede observar que a excepción del generado con $\varphi_{inicial} = 0.4$ los modelos presentan un error de entrenamiento menor al 2%. El mejor modelo fue generado con $\varphi_{inicial} = 0.2$ y posee 34 aprendices débiles. En este caso, no se presenta un modelo que reduzca la complejidad sin acarrear un aumento significativo del error, por este motivo, así se incluya como criterio de selección la complejidad, el mejor modelo se generó con $\varphi_{inicial} = 0.2$.

Finalmente, para la base de datos con ruido del 5% los resultados se presentan en la Tabla 4.12 a partir de la cual se puede observar que los modelos generados poseen un error menor al 2%. El mejor modelo se presenta con $\varphi_{inicial} = 0.4$ y 34 aprendices débiles. Si se considera la complejidad al seleccionar el mejor modelo se podría considerar el generado con $\varphi_{inicial} = 0.2$ puesto que el aumento del error del entrenamiento es menor al 0.2% pero la reducción de la complejidad del modelo es del 55.88%.

A partir del análisis presentado anteriormente se puede observar que el valor de φ inicial depende del problema a resolver puesto que aunque se desea resolver el mismo problema, incluir ruido de diferente potencia conlleva a valores óptimos de $\varphi_{inicial}$ dependientes del caso.

- **Variación de r**

En cuanto a la base de datos con ruido del 0.1% se observa que los modelos generados para la variación de r presentados en la Tabla 4.13 poseen error de entrenamiento menor al 2% lo que conlleva un buen desempeño del controlador. El mejor modelo se generó con $r = 0.5$ y posee 39 aprendices débiles. Si se tiene en cuenta la complejidad al seleccionar el mejor modelo, se podría considerar el modelo entrenado con $r = 0.3$ como el mejor puesto que conlleva a un aumento en el error menor al 0.2% y una disminución de la complejidad del 30.7%.

Para la base de datos con ruido del 1% se presentan los resultados en la Tabla 4.14, en donde los modelos generados presentan un error de entrenamiento menor al 2% y el mejor de ellos fue entrenado con $r = 0.5$ y posee 34 aprendices débiles. Si se tiene en cuenta la complejidad al seleccionar el mejor modelo, se podría considerar el modelo entrenado con $r = 0.7$ puesto que conlleva una disminución de la complejidad del 38.46% con un aumento del error menor al 0.8%.

Finalmente, para la base de datos con ruido del 5% se presentan los resultados en la Tabla 4.15, en donde los modelos generados presentan un error menor al 25, el mejor se calculó con $r = 0.5$ y tiene 34 aprendices débiles. En este caso, el modelo menos complejo también es el que presenta el menor error, por tanto, si se considerara la complejidad para la selección del mejor modelo, no cambiaría el modelo seleccionado.

A partir del análisis presentado anteriormente se puede determinar que el valor óptimo para resolver este problema, sim importar el porcentaje de la potencia del ruido introducido, es $r = 0.5$ puesto que en los tres casos fue el que presentó el menor error, pero no necesariamente la menor

complejidad. Así que en caso de que la complejidad no sea tenida en cuenta para la selección de los modelos, se considera como valor óptimo para los casos presentados $r = 0.5$.

5.1.2.2 Controlador neuronal óptimo del péndulo invertido

Se realiza la comparación de los mejores controladores diseñados para cada base de datos generada por medio de redes neuronales para observar el comportamiento de los modelos en un sistema sin ruido y determinar el efecto de introducir ruido en el proceso de validación. El proceso de validación de cada uno de ellos se lleva a cabo sin introducir ruido al sistema. En la Figura 5.9 se puede observar los resultados de la comparación, en donde los controladores presentan desempeños prácticamente igual puesto que los tres tienen un rango de Overshoot de $-1.688°$ a 1.722 y tiempo de estabilización de 8.4s.

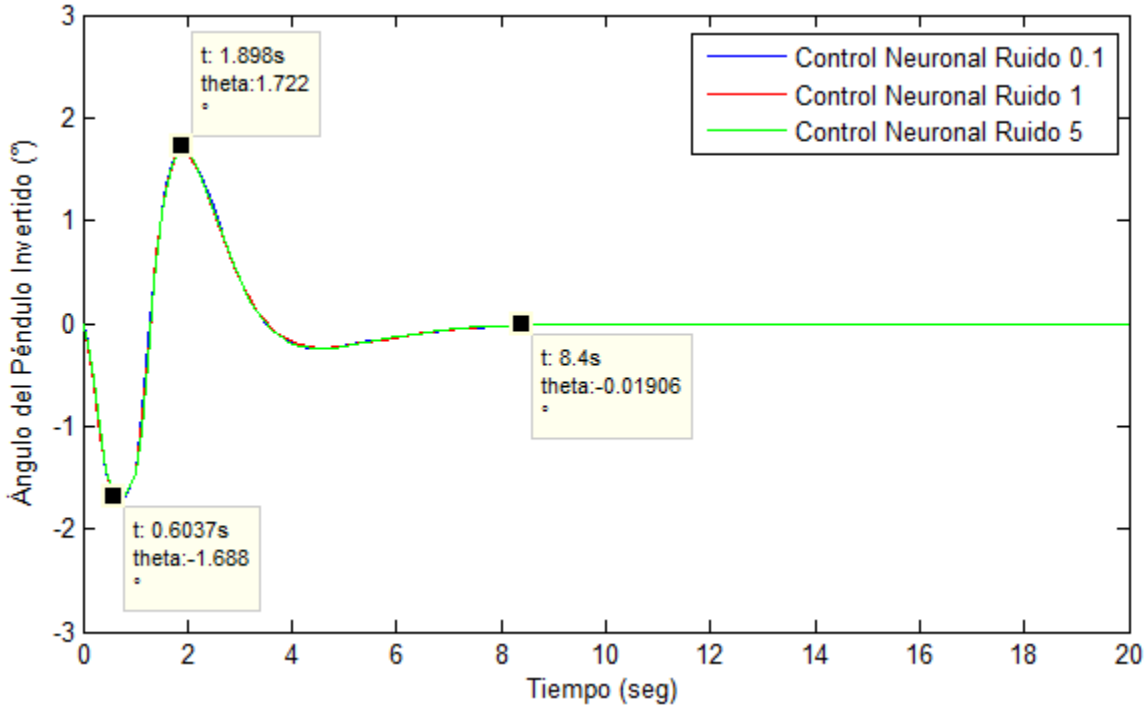


Figura 5.9 Comparación controladores neuronales sin ruido

A partir de la figura, se puede determinar que al entrenar un controlador neuronal con una base de datos contaminada por ruido, además de agregarle robustez al sistema, como se analizó al principio de esta sección, se obtiene un buen desempeño con el modelo de la planta ideal en el que no se presenta ruido dado que al contrario de los modelos en los que se introduce ruido el controlador logra estabilizar al sistema en el valor deseado que es cero, por tanto para la selección del controlador neuronal óptimo se tendría en cuenta la complejidad del modelo. A partir de la información presente en la Tabla 5.6 en donde se presentan los parámetros de desempeño de los controladores neuronales sin ruido se puede seleccionar como controlador neuronal óptimo los entrenados con las bases de datos con ruido del 1% y 5%, pero debido a la robustez de los modelos presentadas en las Tablas Tabla 5.3 y Tabla 5.4, se selecciona como controlador neuronal óptimo el generado con base de datos con ruido del 1%, puesto que presenta menor porcentaje de variación tanto de Overshoot como rango de estabilización.

Tabla 5.6 Parámetros de desempeño de los controladores neuronales sin ruido

Ruido	Overshoot	Tiempo estabilización	Neuronas de Capa Oculta
0.1	−1.688° a 1.722	8.4s	11
1	−1.688° a 1.722	8.4s	5
5	−1.688° a 1.722	8.4s	5

5.1.2.3 Controlador basado en AdaBoost óptimo del péndulo invertido

Se realiza la comparación de los mejores controladores diseñados para cada base de datos generada por medio de AdaBost. El proceso de validación de cada uno de ellos se lleva a cabo sin introducir ruido al sistema. En la Figura 5.10 se puede observar los resultados de la comparación, en donde los controladores presentan desempeños parecidos con valores de Overshoot y tiempo de estabilización diferentes, estos valores se observan más claramente en la Tabla 5.7.

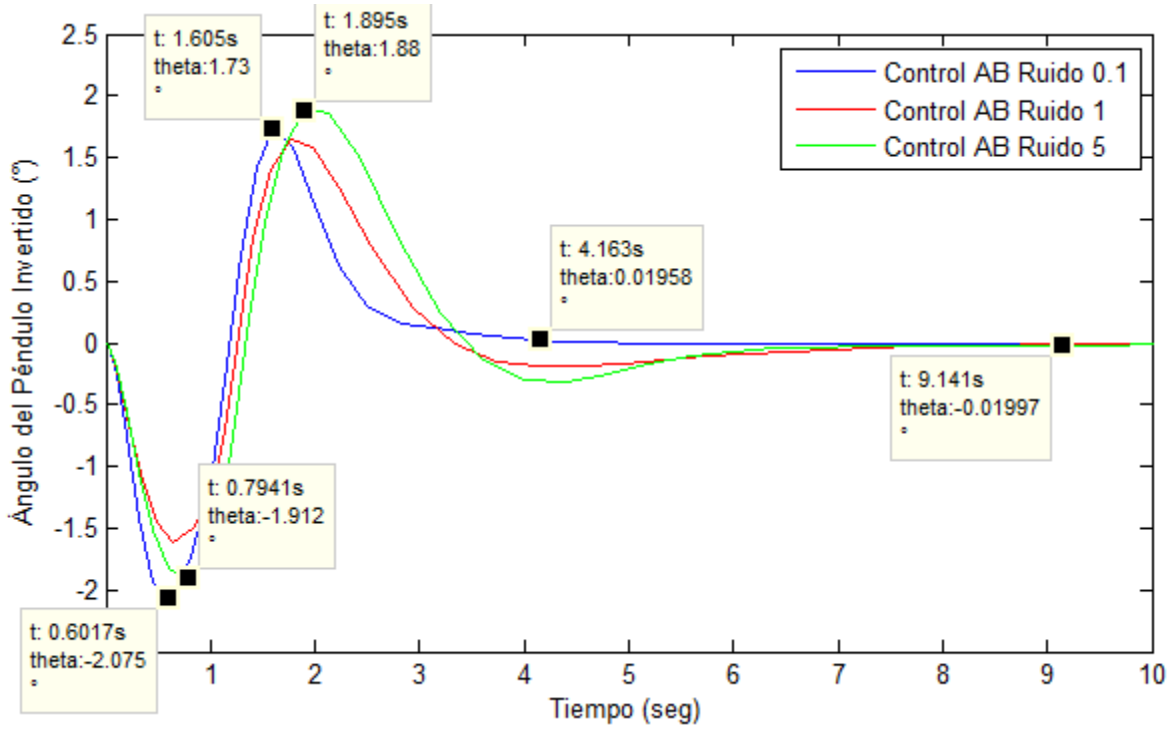


Figura 5.10 Comparación controladores AdaBoost sin ruido

A partir de la figura, se puede determinar que al entrenar un controlador AdaBoost con una base de datos contaminada por ruido, además de agregarle robustez al sistema, como se analizó al principio de esta sección, se obtiene un buen desempeño con el modelo de la planta ideal en el que no se presenta ruido dado que al contrario de los modelos en los que se introduce ruido, el controlador logra estabilizar al sistema en el valor deseado que es cero, se observa que cada controlador presenta rango de Overshoot y tiempo de estabilización diferentes, los cuales se presentan en la Tabla 5.7. A partir de los resultados, se selecciona el controlador generado con la base de datos con ruido del 0.1% como el controlador AdaBoost óptimo debido al bajo tiempo de estabilización que presenta a pesar de ser el modelo de mayor complejidad.

Tabla 5.7 Parámetros de desempeño de los controladores AdaBoost sin ruido

Ruido	Overshoot	Tiempo estabilización	Neuronas de Capa Oculta
0.1	−2.08° a 1.73°	4.16s	39
1	−1.61° a 1.65°	9.13s	34
5	−1.92° a 1.88°	8.34s	34

5.1.2.4 Modelo óptimo de control del péndulo invertido

Finalmente, se presentan controladores óptimos basados en redes neuronales y AdaBoost, se lleva a cabo una comparación de los modelos la cual se presenta en la Figura 5.11. Debido a los antecedentes presentados en la Tabla 2.2 se observa que no hay un criterio único para medir el desempeño del control del péndulo invertido, por lo cual, se continúa con el criterio usado a lo largo del proyecto que es el rango de Overshoot del controlador, el tiempo de estabilización y se procede a calcular el coeficiente de efectividad descrito en [76] según (21) para tener un criterio más de comparación. Los criterios de evaluación se presentan en la Tabla 5.8

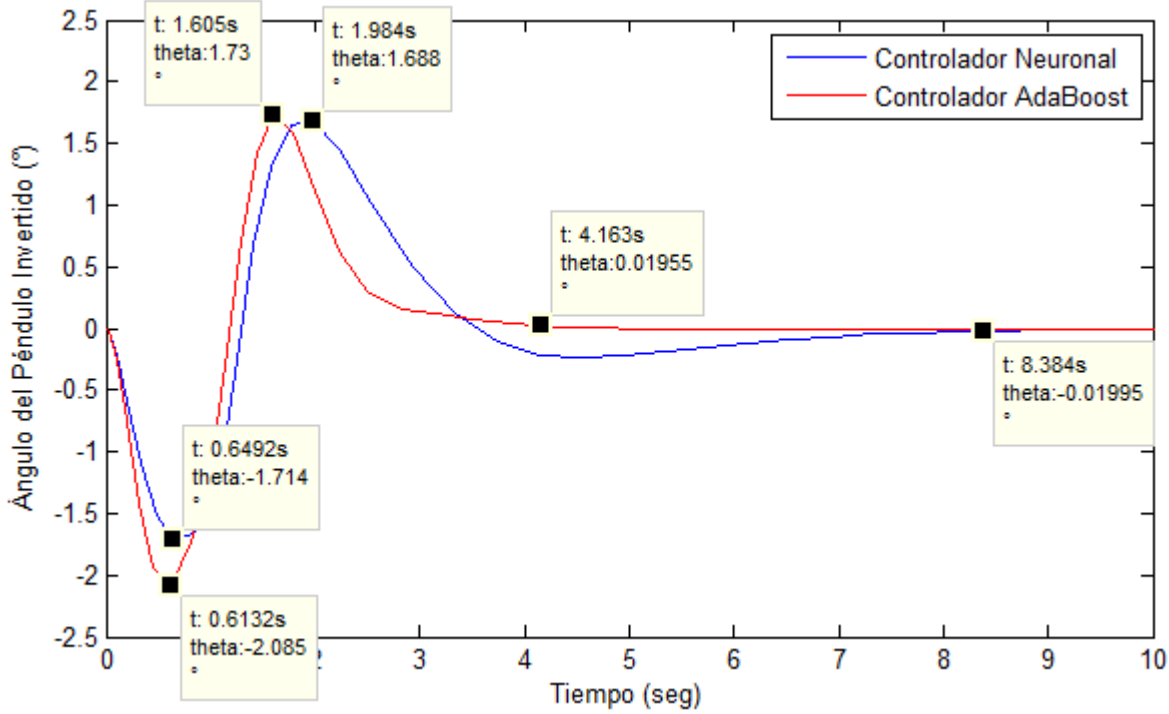


Figura 5.11 Comparación de los mejores modelos basados en Inteligencia Artificial

Tabla 5.8 Criterios de evaluación de los controladores óptimos basados en Inteligencia Artificial

	Rango de Overshoot	*Tiempo estabilización*	*Coef. efectividad*
AdaBoost	−2.08° a 1.73°	4.16 s	0.994
Redes Neuronales	−1.688° *a* 1.722	8.4 *s*	0.995

5.2 DISCUSIÓN

5.2.1 PREDICCIÓN ÍNDICE COLCAP

Como se analizó anteriormente en la sección 5.1.1 los modelos de redes neuronales y AdaBoost para la predicción del índice COLCAP considerados los modelos óptimos presentan un error del 1.206% con 9 neuronas y 1.247% con 48 aprendices débiles respectivamente, pero se hace necesario analizar los resultados en comparación con otros resultados obtenidos en la literatura.

En [67] se llevó a cabo la predicción del índice de la bolsa de valores de Corea por medio de redes neuronales combinadas con métodos computacionales evolutivos como GA y SA, los resultados obtenidos se presentan en la Tabla 2.1. Se puede observar que los modelos propuestos en este trabajo presentan un mejor desempeño que los desarrollados en [67], es necesario tener en cuenta que la predicción se realiza en índices económicos diferentes y esto afecta los resultados puesto que no presentan el mismo comportamiento.

El desempeño que presentaron los modelos presentados en este trabajo, puede deberse al análisis de entradas que se realizó previo al entrenamiento del modelo, dado que se buscaron las acciones que mejor describían el índice COLCAP mientras en la referencia se hace uso de 5 indicadores técnicos para llevar a cabo la predicción, los cuales son Oscilador estocástico (K), Momentum, Tasa de cambio (ROC), Ìndice de comodidad del canal (CCI) y el oscilador seguidor de precio (RSI). .

En [69] se llevó a cabo la predicción del precio de cierre de la acción DELL que cotiza en la bolsa de valores de New York por medio de un método estadístico, ARIMA y Redes Neuronales y sus resultados se presentan en la Tabla 2.1 a partir de los cuales se determina que los modelos propuestos en este trabajo para la predicción del índice COLCAP presentan mejor desempeño que los presentados en el artículo. Dado que el mejor resultado de la referencia se presenta con redes neuronales, y consta de una sola capa oculta el modelo se hace comparable con el usado en este trabajo. A pesar de que el modelo del artículo presenta la misma topología que el de este trabajo, el desempeño tanto del modelo neuronal como del modelo conjunto de AdaBoost presentan mejor desempeño al del artículo. Es necesario resaltar que el comportamiento de cada serie económica es diferente aunque posean características similares.

En la Tabla 5.9 se presenta el resumen de la discusión de la predicción de índices económicos, en donde se encuentra el tipo de error usado para evaluar el desempeño del modelo de predicción, el método de predicción usado, el índice a predecir y el desempeño del sistema predictor.

De acuerdo a lo observado en el desarrollo del proyecto y la comparación realizada previamente, se puede determinar que el análisis previo de la serie y la selección de las señales que harán de entrada del sistema conllevan a obtener modelos que presentan un mayor desempeño comparado con aquellos en que no se realiza un análisis de la serie.

Tabla 5.9 Resumen discusión predicción de índices económicos

Referencia	Índice	Método de Predicción	Tipo de Error	Desempeño
Assessment and prediction of tropospheric ozone concentration levels	DELL	ARIMA	MAPE (%)	11.134

using artificial neural networks [69]		ANN	MAPE (%)	15.041
Predicting Stock Index Using Neural Network Combined with Evolutionary Computation Methods [67]	Corea Price	NN con GA	MAPE (%)	30.173
		NN con SA	MAPE (%)	22.603
		NN con GA+SA	MAPE (%)	40.544
Este Trabajo	COLCAP	Redes Neuronales	MAPE(%)	1.206
		AdaBoost.RT	MAPE(%)	1.247

5.2.2 Control del sistema del péndulo invertido

En los analices llevados a cabo anteriormente del péndulo invertido se observó que los modelos de control cumplieron su objetivo principal puesto que mantuvieron el sistema dentro del rango de estabilización de $-5°$ a $5°$, sin embargo, se hace necesario comparar los resultados obtenidos con resultados de la literatura para validar los modelos generados en este proyecto.

En [75] se llevó a cabo el control del péndulo invertido con las mismas características del sistema por medio de un controlador PID y un LQR y fueron usados como criterio de evaluación el tiempo de establecimiento y el rango del Overshoot del sistema controlado. Los resultados del trabajo se presentan en la Tabla 5.10 junto a los resultados de los controladores óptimos presentados en la Tabla 5.8, en donde se puede observar que el método LQR propuesto en la referencia presenta el menor tiempo de establecimiento, sin embargo, también presenta el rango de Overshoot más grande de los modelos presentados, por lo cual, si se busca que el sistema se estabilice en el menor tiempo posible, el modelo LQR sería el mejor modelo para controlar el péndulo.

Si la aplicación específica requiere mantener en todo momento el sistema en el menor rango de variación del valor deseado, se considera el modelo neuronal generado en este trabajo como el modelo óptimo puesto que es el que presenta el menor rango de Overshoot de todos los controladores.

Finalmente, si se requiere cumplir con la condición de tiempo de establecimiento y de variación del valor deseado al mismo tiempo, el controlador AdaBoost propuesto en este trabajo es el que mejor cumple los requisitos.

Tabla 5.10 Comparación resultados de control de [75] y los obtenidos en este proyecto

Criterio de evaluación	**Performance comparison between LQR and PID controllers for an inverted pendulum system** [75]**.**		**Este trabajo**	
Tiempo de establecimiento	PID	4.48 s	Neuronal	8.4s
	LQR	3.34 s	AdaBoost	4.16s
Rango del Overshoot	PID	$0.50° \; a \; -4.39°$	Neuronal	$-1.688° \; a \; 1.722$
	LQR	$19.6° a \; -46.5°$	AdaBoost	$-2.08°$ a $1.73°$

En [76], se llevó a cabo el control del péndulo invertido por medio de cinco controladores diferentes y se hace uso del coeficiente de efectividad como criterio de evaluación de los controladores, el cual, entre más cercano a uno sea el coeficiente se considera mejor el controlador, los resultados de este artículo se presenta en la Tabla 5.11, en donde se puede observar que los controladores propuestos en este trabajo presentan un valor aproximadamente 1 puesto que no presentan rango de estabilización y además de la variación del Overshoot el sistema se mantiene en el valor deseado.

Tabla 5.11. Comparación resultados de control de [76] y los obtenidos en este proyecto

Criterio de evaluación	**Comparative Study of Motion Control Methods for a Nonlinear System** [76]		**Este trabajo**	
Coeficiente de efectividad	PD	0.324	Neuronal	0.995
	LQR	0.524		
	Red Neuronal	0.785	AdaBoost	0.994
	No lineal	0.248		
	Lógica difusa	0.349		

Al comparar los resultados se observa que los controladores generados en este proyecto poseen un desempeño mejor con los encontrados en la literatura, puesto que presentan menor Overshoot y mayor coeficiente de efectividad, pero es necesario tener en cuenta que en ambos casos se hizo uso de un controlador de referencia, el que se busca imitar por medio de AdaBoost y redes neuronales, cuyo desempeño ya había sido validado en [54], lo que aseguraba un buen desempeño de los controladores generados a partir de este.

6. CONCLUSIONES

En este capítulo se presentan las conclusiones del proyecto realizado y los trabajos futuros que surgieron a medida del desarrollo del mismo.

6.1 RESUMEN

En este proyecto se generó un modelo de predicción del índice COLCAP y un controlador con características no lineales basado en el algoritmo AdaBoost.RT con φ auto-adaptativo. Para el caso de la predicción se lleva a cabo a partir de acciones que hacen parte de la bolsa de valores colombiana y a partir de los valores de las acciones del día actual se realiza la predicción del índice COLCAP del día siguiente. En cuanto al caso de control se estabilizó el sistema del péndulo invertido por medio de un controlador diseñado a partir del algoritmo AdaBoost. La motivación de este proyecto se basa en que en la revisión bibliográfica no se encontró aplicación del algoritmo en predicción de series económicas ni en problemas de control.

Para la predicción del índice COLCAP se realizó primero un análisis de la serie para identificar si presentaba estacionalidad, luego se hizo la selección y el análisis de los parámetros de entrada del sistema clasificador, en seguida se determinaron los modelos de predicción por redes neuronales y por medio del algoritmo AdaBoost.RT con φ auto-adaptativo, en donde para cada caso, se realizaron experimentos para encontrar la arquitectura óptima del modelo y en el caso específico de AdaBoost se realizaron experimentos adicionales para sintonizar los parámetros propios del algoritmo, determinando los valores óptimos de la actualización de pesos de las muestras correctas ($\beta = error$), el valor inicial del delimitador de muestras correctas ($\varphi_{inicial} = 0.4$) y el valor relativo a la tasa de cambio del error ($r = 0.5$) para la adaptación de φ, con un error del 1.247% con 48 aprendices débiles.

A partir de los experimentos realizados, se determinó que el valor de cada uno de ellos influye en el error medio absoluto de predicción y que el valor óptimo de cada uno de los parámetros mencionados anteriormente es necesario sintonizarlo de acuerdo al problema que se desea resolver, como se presenta en la sección 3.5.2

Para el control del sistema del péndulo invertido se llevó a cabo la generación de los modelos de Inteligencia Artificial haciendo uso de un controlador de referencia, esto debido a que como se mencionó anteriormente, no se había implementado el algoritmo AdaBoost en problemas de control y antes de empezar directamente con la creación de un modelo de control adaptivo, se comprobara el comportamiento no lineal del controlador diseñado. Por tanto, en un principio se generó el modelo del péndulo invertido a partir de las variables de estado Luego se generaron tres bases de datos contaminadas con diferentes potencias de ruido y a partir de estas se llevó a cabo el entrenamiento de los controladores basados en inteligencia Artificial. Se realizaron pruebas para encontrar la arquitectura óptima de los modelos y en el caso específico de AdaBoost se realizaron experimentos adicionales para sintonizar los parámetros propios del algoritmo.

A partir de las pruebas realizadas para AdaBoost, se determinaron los valores óptimos presentes en la Tabla 5.5 de cada uno de los parámetros para cada base de datos, a partir de lo cual se determinó que aunque se busca resolver el mismo proceso (estabilización del péndulo invertido), el introducir

ruido de diferentes potencia hace cada caso un problema específico, en donde los valores óptimos para el sistema con una potencia de ruido determinada difieren de los de el mismo sistema con otra potencia de ruido, como se observa en la Tabla 5.5, a partir de lo cual se puede concluir que el desempeño del controlador tiene dependencia de los valores de los parámetros del algoritmo, por lo cual es necesario sintonizarlos para obtener un buen desempeño, en este caso, los rangos de Overshoot del sistema controlado se mantienen entre -5° a 5° y se obtiene un error de estado estacionario de 0 cuando se evalúa cada sistema realimentado sin error.

También es necesario aclarar que el desempeño de los controladores generados está ligado directamente con el controlador de referencia usado, puesto que los controladores generados buscaban aprender el comportamiento del controlador de referencia, que tiene un Overshoot de 1.72° y tiempo de establecimiento de 5.4s, de tal manera que al hacer uso de este tipo de control es de vital importancia escoger un controlador de referencia con un muy buen desempeño dado que tiene una repercusión en el comportamiento y los resultados del controlador generado, que en el caso de redes neuronales presenta un Overshoot de 1.72° y tiempo de establecimiento de 8.4s y para AdaBoost un Overshoot de 2.08° y tiempo de estabilización de 4.16s. Al incluir un controlador de referencia es necesario llevar a cabo todo el proceso de diseño de control que se espera evitar haciendo uso de controladores basados en Inteligencia Artificial.

El desempeño del algoritmo AdaBoost tiene una alta relación con los valores de inicialización del aprendiz débil, dado que el espacio de búsqueda no es simple y fue necesario llevar a cabo experimentos para poder obtener un resultado con un error menor al 2% en el caso de predicción y estabilizar el sistema del péndulo invertido para el caso de control.

Finalmente se observó que el modelo de predicción de AdaBoost presentó un error menor al 2% lo que hace al algoritmo AdaBoost.RT con φ auto-adaptativo un modelo de Inteligencia Artificial capaz de resolver satisfactoriamente problemas de predicción de series de tiempo y para el caso del control se logró estabilizar el sistema con un error de estabilización de 4.16s y Overshoot de 4.39° con error de estado estacionario cero, lo que lo indica que es un algoritmo capaz de generar modelos de control con comportamiento idóneo para llevar a cabo el control de sistemas altamente no lineales.

6.2 TRABAJO FUTURO

En el desarrollo del proyecto se plantearon propuestas para continuar con el mejoramiento del algoritmo AdaBoost.RT con φ auto-adaptativo y su uso en problemas de predicción y control:

- Diseño de un método autónomo de selección de los parámetros $\beta, \varphi_{inicial}$ y r, de tal manera que no sea necesario realizar experimentos para encontrar a prueba y error el valor óptimo para resolver el problema, sino de acuerdo a las características propias del problema encontrar el valor óptimo de los parámetros.
- Diseño de un método auto-adaptativo de φ diferente al usado en este proyecto, que tenga en cuenta además del error de la iteración, el gradiente del mismo, de tal manera que se tenga en cuenta el error de iteraciones seguidas.
- Implementar esquemas de control basados en Inteligencia Artificial, en el cual no se haga necesario el diseño previo del controlador de referencia, sino que el ajuste del controlador del controlador se haga de manera on-line por medio del algoritmo de AdaBoost
- Diseño de un controlador adaptativo que realiza el ajuste de los parámetros por medio de AdaBoost.RT con φ auto-adaptativo.

BIBLIOGRAFÍA

[1] V. Ortega, P. Ojeda, F. Sutil, and C. Sierra, "Culpabilidad sexual en adolescentes : Estudio de algunos factores relacionados Introducción," *An. Psicol.*, vol. 21, pp. 268–275, 2005.

[2] A. Alderete, "Fundamentos del Análisis de Regresión Logística en la Investigación Psicológica," *Evaluar*, vol. 6, pp. 52–67, 2006.

[3] D. Fernando, C. Madariaga, J. Leonardo, and G. Rodríguez, "Aplicación de la regresión lineal en un problema de pobreza Application of linear regression on the problem of poverty."

[4] A. F. R. L. de Hierro, J. Martínez-Moreno, C. Aguilar Peña, and C. Roldán, "A fuzzy regression approach using Bernstein polynomials for the spreads: Computational aspects and applications to economic models," *Math. Comput. Simul.*, vol. 128, pp. 13–25, 2016.

[5] S. Ahmad, N. M. Ramli, and H. Midi, "Outlier detection in logistic regression and its application in medical data analysis," in *Humanities, Science and Engineering (CHUSER), 2012 IEEE Colloquium on*, 2012, no. Chuser, pp. 503–507.

[6] H. Y. Lin, W. Wang, Y. H. Liu, S. J. Soong, T. P. York, L. Myers, and J. J. Hu, "Comparison of multivariate adaptive regression splines and logistic regression in detecting SNP-SNP interactions and their application in prostate cancer," *J. Hum. Genet.*, vol. 53, no. 9, pp. 802–811, 2008.

[7] M. Hou, Y. Wang, and B. Chaib-draa, "Online Local Gaussian Process For Tensor-Variate Regression: Application to Fast Reconstruction of Limb Movements From Brain Signal," vol. 167, no. 1998, p. 11104, 2000.

[8] D. Berberidis, V. Kekatos, and G. B. Giannakis, "Online Censoring for Large-Scale Regressions with Application to Streaming Big Data," vol. 64, no. July, pp. 1–12, 2015.

[9] J. Frecon, R. Fontugne, G. Didier, N. Pustelnik, K. Fukuda, and P. Abry, "Non-linear regression for bivariate self-similarity identification-application to anomaly detection in Internet traffic based on a joint scaling analysis of packet and byte counts," *2016 IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 4184–4188, 2016.

[10] A. Eslami, E. M. Qannari, A. Kohler, and S. Bougeard, "Multi-group PLS Regression: Application to Epidemiology," *Springer Proc. Math. Stat.*, vol. 56, pp. 243–255, 2013.

[11] J. L. Devore, *Probabilidad y Estadística para Ingeniería y Ciencias*, 7th ed. San Luis Obispo: CENGAGE Learning.

[12] N. Liu, Z. Lin, J. Cao, Z. Koh, T. Zhang, G. Bin Huang, W. Ser, and M. E. H. Ong, "An intelligent scoring system and its application to cardiac arrest prediction," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 6, pp. 1324–1331, 2012.

[13] G. V. Weinberg, "Estimation of Pareto clutter parameters using order statistics and linear regression," *Electron. Lett.*, vol. 49, no. 13, pp. 845–846, 2013.

[14] F. Liang, "Cycle time estimation approach based on mathematical programming and statistical regression," *2010 IEEE 17Th Int. Conf. Ind. Eng. Eng. Manag.*, pp. 742–745, 2010.

[15] H. Jiang and Z. Wang, "GMRVVm-SVR model for financial time series forecasting," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 7813–7818, 2010.

[16] P. Senthil Kumar, A. Saravanan, R. Yashwanth, K. Anish Kumar, and S. Visvesh, "Removal of toxic zinc from water/wastewater using eucalyptus seeds activated carbon: non-linear regression analysis," *IET Nanobiotechnology*, vol. 10, no. 4, pp. 244–253, 2016.

[17] J. G. De Gooijer and R. J. Hyndman, "25 Years of Time Series Forecasting," *Int. J. Forecast.*, vol. 22, no. 3, pp. 443–473, 2006.

[18] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *J. Clin. Epidemiol.*, vol. 49, no. 11, pp. 1225–1231, 1996.

[19] W. C. Wang, K. W. Chau, C. T. Cheng, and L. Qiu, "A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series," *J. Hydrol.*, vol. 374, no. 3–4, pp. 294–306, 2009.

[20] F. Eyben, F. Weninger, E. Marchi, and B. Schuller, "Likability of human voices: A feature analysis and a neural network regression approach to automatic likability estimation," *Int. Work. Image Anal. Multimed. Interact. Serv.*, pp. 8–11, 2013.

[21] J. N. K. Liu and B. O. Feng, "a Neural Network Regression Model for Tropical Cyclone Forecast," *Mach. Learn.*, no. August, pp. 18–21, 2005.

[22] J. Arshad, A. Zameer, and A. Khan, "Wind Power Prediction Using Genetic Programming Based Ensemble of Artificial Neural Networks (GPeANN)," *2014 12th Int. Conf. Front. Inf. Technol.*, pp. 257–262, 2014.

[23] D. Xu, "Research on the prediction of gasoline engine air intake flow based on the BP neural network," *Proc. - 2015 Int. Conf. Intell. Transp. Big Data Smart City, ICITBS 2015*, pp. 673–678, 2016.

[24] P. Ge, F. Gao, and G. Chen, "Predictive models for prostate cancer based on logistic regression and artificial neural network," *2015 IEEE Int. Conf. Mechatronics Autom.*, pp. 1472–1477, 2015.

[25] H. Patel, M. Pandya, and M. Aware, "Short Term Load Forecasting of Indian System Using Linear Regression and Artificial Neural Network," *2015 5th Nirma Univ. Int. Conf. Eng.*, no. 1, pp. 1–5, 2015.

[26] R. ~L. Watrous, "Learning Algorithms for Connectionist Networks : Applied Gradient Methods of Non-Linear Optimization," *Tech. Reports*, no. MS-CIS-87-51, p. 597, 1987.

[27] R. E. Schapire, "The Strength of Weak Learnability (Extended Abstract)," *Mach. Learn.*, vol. 227, pp. 28–33, 1989.

[28] Y. Freund and R. E. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting.," *Comput. Syst. Sci.*, vol. 57, pp. 119–139, 1997.

[29] R. E. Schapire, "Explaining adaboost," *Empir. Inference Festschrift Honor Vladimir N. Vapnik*, pp. 37–52, 2013.

[30] E. Alfaro, N. García, M. Gámez, and D. Elizondo, "Bankruptcy forecasting: An empirical comparison of AdaBoost and neural networks," *Decis. Support Syst.*, vol. 45, no. 1, pp. 110–122, 2008.

[31] Y. Wang, P. Han, X. Lu, R. Wu, and J. Huang, "The performance comparison of Adaboost and SVM applied to SAR ATR," *CIE Int. Conf. Radar Proc.*, vol. 0, pp. 1–4, 2007.

[32] J. Wen, X. Zhang, Y. Xu, Z. Li, and L. Liu, "Comparison of AdaBoost and logistic regression for detecting colorectal cancer patients with synchronous liver metastasis," *2nd Int. Conf. Biomed. Pharm. Eng. ICBPE 2009 - Conf. Proc.*, 2009.

[33] Y. Freund and R. E. Schapire, "A Desicion-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Comput. Learn. theory*, vol. 904, no. Chapter 2, pp. 23–37, 1995.

[34] D. P. Solomatine and D. L. Shrestha, "AdaBoost.RT: a boosting algorithm for regression problems," *2004 IEEE Int. Jt. Conf. Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, pp. 1163–1168, 2004.

[35] G.-S. Hu, F.-F. Zhu, Y.-C. Zhang, and J.-L. Yu, "Study of Integrating AdaBoost and Weight Support Vector Regression Model," *2009 Int. Conf. Artif. Intell. Comput. Intell.*, pp. 258–262, 2009.

[36] T. Lasota, B. Londzin, Z. Telec, and B. Trawiński, "Comparison of ensemble approaches: Mixture of experts and Ada Boost for a regression problem," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8398 LNAI, no. PART 2, pp. 100–109, 2014.

[37] H. Liu, H. Tian, Y. Li, and L. Zhang, "Comparison of four Adaboost algorithm based artificial neural networks in wind speed predictions," *Energy Convers. Manag.*, vol. 92, pp. 67–81, 2015.

[38] H. X. Tian and Z. Z. Mao, "An ensemble ELM based on modified AdaBoost.RT algorithm for predicting the temperature of molten steel in ladle furnace," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 73–80, 2010.

[39] L. Zhou and K. Keung, "AdaBoost Models for Corporate Bankruptcy Prediction with Missing Data," *Comput. Econ.*, 2016.

[40] J. Sun, M. Y. Jia, and H. Li, "AdaBoost ensemble for financial distress prediction: An empirical comparison with data from Chinese listed companies," *Expert Syst. Appl.*, vol. 38, no. 8, pp. 9305–9312, 2011.

[41] S. Liu, J. Xu, J. Zhao, X. Xie, and W. Zhang, "Efficiency enhancement of a process-based rainfall-runoff model using a new modified AdaBoost.RT technique," *Appl. Soft Comput. J.*, vol. 23, pp. 521–529, 2014.

[42] B. de V. de Colombia, "Metodología para el cálculo del íncide COLCAP," Bogotá, 2016.

[43] G. A. Vásquez Perdomo and J. Muñoz Sepúlveda, "Diseño y Evaluación de un Modelo de Pronóstico para el Índice COLCAP mediante Filtros de señal y Redes Neuronales Artificiales," in *Encuentro Internacional de Investigadores en Administración*, 2011, no. c, pp. 625–643.

[44] O. A. Espinosa Acuña and P. A. Vaca González, "Fitting the Classical and Bayesian GARCH models with Student-t innovations to COLCAP Index," *Simp. Int. Estadística 2015*, no. c, 2015.

[45] D. Upadhyay, N. Tarun, and T. Nayak, "ANN based intelligent controller for inverted pendulum system," *2013 Students Conf. Eng. Syst. SCES 2013*, 2013.

[46] J. S. Noh, G. H. Lee, and S. Jung, "Position control of a mobile inverted pendulum system using radial basis function network," *Int. J. Control. Autom. Syst.*, vol. 8, no. 1, pp. 157–162, 2010.

[47] A. K. Singh and P. Gaur, “Adaptive control for non-linear systems using artificial neural network and its application applied on inverted pendulum,” *India Int. Conf. Power Electron. 2010*, pp. 1–8, 2011.

[48] . Bolsa de Valores de Colombia, “Índices bursátiles en línea.” [Online]. Available: https://www.bvc.com.co/pps/tibco/portalbvc/Home/Mercados/enlinea/indicesbursatiles?com.tibco.ps.pagesvc.renderParams.sub45d083c1_14321f5c9c5_-78350a0a600b=action=detallar&org.springframework.web.portlet.mvc.ImplicitModel=true&.

[49] D. F. Specht, “A general regression neural network,” *Neural Networks, IEEE Trans.*, vol. 2, no. 6, pp. 568–576, 1991.

[50] S. Haykin, *Neural networks: a comprehensive foundation*, vol. 13, no. 4. Pearson, 1999.

[51] G. P. Zhang, E. B. Patuwo, and H. Michael Y., “Forecasting with artificial neural networks: The state of the art,” *Int. J. Forecast.*, vol. 14, no. 1, pp. 35–62, 1998.

[52] Z. Benmabrouk, A. Abid, B. E. N. H. M, and L. Sbita, “Speed Control of DC Machine Using adaptive neural IMC controller based on recurrent neural network,” pp. 198–203, 2016.

[53] W. Shen and M. Xing, “Stock index forecast with back propagation neural network optimized by genetic algorithm,” *2009 2nd Int. Conf. Inf. Comput. Sci. ICIC 2009*, vol. 2, pp. 376–379, 2009.

[54] J. S. Guez, Allon, “Neurocontroller Design Via Supervised and Unsupervised Learning,” *J. Intell. Robot. Syst.*, vol. 2, no. 1983, pp. 307–335, 1989.

[55] C. W. Anderson, “Learning to control an inverted pendulum using neural networks,” *Control Syst. Mag. IEEE*, vol. 9, no. April, pp. 31–37, 1989.

[56] H.-Y. Lin, Y.-F. Juan, and A.-P. Chen, “Hybrid Intelligent Trading Approach XCS Neural Network Model for Taiwan Stock Index Trend Forecasting,” *2007 Int. Conf. Converg. Inf. Technol. (ICCIT 2007)*, pp. 1408–1416, 2007.

[57] D. P. Solomatine and D. L. Shrestha, “AdaBoost. RT: a boosting algorithm for regression problems,” *IEEE Int. Jt. Conf. Neural Networks*, vol. 2, pp. 1163–1168, 2004.

[58] G. Leshem and Y. Ritov, “Traffic flow prediction using adaboost algorithm with random forests as a weak learner,” *Proc. Int. Conf. …*, vol. 1, no. 1, pp. 193–198, 2007.

[59] Y. Shen, Y. Jiang, W. Liu, and Y. Liu, “R - Multi-class AdaBoost ELM.pdf,” *Proceedings in Adaptation, Learning and Optimization*. pp. 179–188, 2015.

[60] P. Zhang and Z. Yang, “Ensemble Extreme Learning Machine Based on a New Self-adaptive AdaBoost.RT,” *Proceedings of ELM-2014 Volume 1 SE - 21*, vol. 3. pp. 237–244, 2015.

[61] J. D. Cryer and K.-S. Chan, “Introduction,” in *Time Series Analysis with Applications in R*, 2008, pp. 1–18.

[62] B. E. Hansen, *Econometrics*. 2016.

[63] Autorregulador del Mercado de Valores de Colombia, “Todo lo que un Inversionista debe saber sobre los nuevos Índices de la Bolsa de Valores de Colombia.” .

[64] S. S. Sastry and A. Isidori, “Adaptive Control of Linearizable Systems,” *IEEE Trans. Automat. Contr.*, vol. 34, no. 11, pp. 1123–1131, 1989.

[65] Y. Xin, B. Xu, H. Xin, J. Xu, and L. Hu, “The computer simulation and real-time control for the inverted pendulum system based on PID,” *Lect. Notes Electr. Eng.*, vol. 100 LNEE, no. VOL. 4, pp. 729–736, 2011.

[66] H. H. Çevik, “A Comparative Study of Artificial Neural Network and ANFIS for Short Term Load Forecasting,” pp. 0–5, 2014.

[67] A. H. Kamal, H. A. Abdelbary, and A. R. Abas, “Predicting Stock Index Using Neural Network Combined with Evolutionary Computation Methods,” in *2010 IEEE 17Th International Conference on Informatics and Systems*, 2010.

[68] S. Mahfoud, G. Mani, and S. Reigel, “Nonlinear Versus Linear Techniques for Selecting Individual Stocks,” in *Decision Technologies for Financial Engineering*, 1997, pp. 65–75.

[69] A. A. Adebiyi, A. O. Adewumi, C. K. Ayo, S. . a Abdul-Wahab, and S. . M. Al-Alawi, “Assessment and prediction of tropospheric ozone concentration levels using artificial neural networks,” *Environ. Model. Softw.*, vol. 17, pp. 219–228, 2002.

[70] S. Chi, H. Chen, and C. Cheng, “A Forecasting Approach for Stock Index Future Using Grey Theory and Neural Networks,” pp. 3850–3855, 1998.

[71] P.-F. Pai and C.-S. Lin, “A hybrid ARIMA and support vector machines model in stock price forecasting,” *Omega*, vol. 33, pp. 497–505, 2005.

[72] L. B. Prasad, B. Tyagi, and H. O. Gupta, “Optimal control of nonlinear inverted pendulum system using PID controller and LQR: Performance analysis without and with disturbance input,” *Int. J. Autom. Comput.*, vol. 11, no. 6, pp. 661–670, 2014.

[73] M. Bugeja, “Non-linear swing-up and stabilizing control of an inverted pendulum system,” *IEEE Reg. 8 EUROCON 2003 Comput. as a Tool - Proc.*, vol. B, pp. 437–441, 2003.

[74] Z. Li and C. Xu, “Adaptive fuzzy logic control of dynamic balance and motion for wheeled inverted pendulums,” *Fuzzy Sets Syst.*, vol. 160, no. 12, pp. 1787–1803, 2009.

[75] A. N. K. Nasir, M. A. Ahmad, and M. F. Rahmat, “Performance comparison between LQR and PID controllers for an inverted pendulum system,” *AIP Conf. Proc.*, vol. 1052, pp. 124–128, 2008.

[76] D. J. Pack and M. Meng, “Comparative Study of Motion Control Methods for a Nonlinear System,” 1992.

[77] V. Sirisha and A. S. Junghare, “A Comparative study of controllers for stabilizing a Rotary Inverted Pendulum,” *Int. J. Chaos, Control. Model. Simul.*, vol. 3, no. 1/2, pp. 1–13, 2014.

[78] K. E. Xu and X. Duan, “ComparativeStudy of Control Methods of Single-Rotational Inverted Pendulum,” no. November, pp. 776–778, 2002.

[79] V. Mladenov, “Application of Neural Networks for Control of Inverted Pendulum,” *WSEAS Trans. Circuits Syst.*, vol. 10, no. 2, pp. 49–58, 2011.

[80] S. Sharma, V. Kumar, and R. Kumar, “Supervised Online Adaptive Control of Inverted Pendulum System Using ADALINE Artificial Neural Network with Varying System Parameters and External Disturbance,” *Int. J. Intell. Syst. Appl.*, vol. 4, no. 8, pp. 53–61, 2012.

[81] P. J. Brockwell and R. a Davis, *Introduction to Time Series and Forecasting , Second Edition*. 2002.

[82] R. Sallehuddin, S. M. H. Shamsuddin, and S. Z. M. Hashim, “Application of Grey Relational Analysis for Multivariate Time Series,” *2008 Eighth Int. Conf. Intell. Syst. Des. Appl.*, vol. 2, pp. 432–437, 2008.

[83] A. C. Rencher, *Methods of Multivariate Analysis*, Second Edi. 2002.

ANEXO 1 – Análisis de Entradas de Series de Tiempo

Análisis Relacional Gris

Antes de entrar en detalles en el Análisis de relaciones gris (GRA, por sus siglas en inglés), es necesario hablar sobre la teoría gris, el cuál es un sistema en el que una parte de la información es conocida y la otra parte desconocida, la cual se ha ido desarrollando en un grupo de teorías y técnicas de las que hace parte matemática, análisis de relaciones, modelamiento, agrupamiento, predicción, toma de decisiones, programación y control gris que han sido utilizadas satisfactoriamente en múltiples campos de la ingeniería. El mayor avance de la teoría gris es que puede lidiar tanto con información incompleta como con problemas poco claros de una manera bastante precisa [82].

GRA es un método de análisis basado en matemáticas geométricas que cumple con los principios de normalidad, simetría, totalidad y proximidad es adecuado para resolver complicadas relaciones entre múltiples factores y variables y ha sido aplicado satisfactoriamente en análisis de clúster, planeamiento de trayectoria de robot, análisis de predicción y criterio de decisiones múltiples [82].

GRA consta de tres pasos principales. El primero de ellos es pre-procesamiento de los datos, el cuál es requerido cuando el rango de uno de los parámetros es diferentes de los demás, por tanto, los datos deben ser normalizados, escalados y polarizados en series que sean comparables antes de proceder con los demás pasos. Hay dos procesos involucrados en la etapa de pre-procesamiento, representación y normalización de los datos. Inicialmente la serie de datos original (COLCAP) es representada como la referencia (x_0) y las series de las acciones usadas para la predicción como series comparativas (x_i). Luego, se normaliza los datos y para esto hay múltiples maneras de llevar a cabo el pre-procesamiento como los mostrados en (24) y (25) dependiendo de los resultados que se deseen, por ejemplo:

Si se espera “mayor mejor”, el pre-procesamiento se llevará a cabo como se muestra en
)

$$x_i^* = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{24}$$

Si se espera “menor mejor”, el pre-procesamiento se llevará a cabo como se muestra en (25)

$$x_i^* = \frac{\max(x_i) - x_i}{\max(x_i) - \min(x_i)} \tag{25}$$

Donde, para ambos casos

$i = 1, \dots , m$
m es el número de series comparativas que se desea normalizar, para nuestro caso $m = 6$

x_i es la i-ésima secuencia original de la serie i
x_i^* es la i-ésima secuencia luego de la normalización
$\min(x_i)$ y $\max(x_i)$ es el mínimo y el máximo valor de la i-ésima serie x_i

El segundo paso es hallar el coeficiente relacional gris usando (26)

$$\xi_i = \frac{\Delta\, min + \zeta\, \Delta\, max}{\Delta_i + \zeta\, \Delta\, max} \tag{26}$$

Donde,
x_0^* es la serie de referencia COLCAP normalizada
x_i^* es la i-ésima serie comparativa normalizada
$i = 1, \ldots, m$
m es el número de series comparativas que se desea normalizar, para nuestro caso $m = 6$
$\Delta_i = \|x_0 - x_i\|$ son las secuencias de desviación entre la series de referencia y la i-ésima serie comparativa.
$\Delta\, min = \min(\Delta_i)$
$\Delta \max = \max(\Delta_i)$
ζ es conocido como un coeficiente de identificación.

El coeficiente $\zeta \in [0,1]$, el cual puede ser ajustado para tener una mejor distinción entre la serie de referencia normalizada y las series comparativas normalizadas. Normalmente $\zeta = 0.5$ dado que ofrece un efecto distinguidor moderado y estabilidad.

Luego de calcular el coeficiente relacional gris, se calcula el grado relacional gris (GRG, por sus siglas en inglés) el cual está definido como una medida numérica de la relevancia entre 2 secuencias como una secuencia de referencia y una secuencia comparativa. El grado relacional gris existente entre dos series siempre está distribuido entre 0 y 1 y se calcula usando (27)

$$\gamma_i = \frac{1}{m}\sum_{i=1}^{m} \xi_i \tag{27}$$

Donde γ_i representa el grado relacional gris. Generalmente $\gamma_i > 0.9$ indica una influencia marcada, $\gamma_i > 0.8$ una influencia relativamente marcada, $\gamma_i > 0.7$ una influencia notable y $\gamma_i < 0.5$ una influencia despreciable [82],

ANEXO 2 – Función AdaBoost.RT con φ auto-adaptativo

```
function [est_total, modelo,MARE] = adaboost_reg(inputs,y_des,itt)
%Adaboost
%Estimado Total (est_total) es el resultado del "StrongLearner"
%En modelo se encuentran las características del modelo "StrongLearner".
%Inputs son las características de entrada del problema
%y_des es la predicción esperada
%itt es el número máximo de "WeakLearner" que se desea encontrar
phi(1) = 0.2; % %Define el valor inicial de phi
n = length(y_des); %Cantidad de muestras
r = 0.5;
%D es el peso de las muestras de entrenamiento. En la primera iteración todas las muestras son igual de
importantes, es decir, tienen el mismo peso
D(:,1) = ones(n,1)/n;
est_total = 0;

%Se realiza el número de iteraciones de entrenamiento del modelo
for t=1:itt
    [w,err,y_est] = weak_learner(inputs,y_des,D(:,t),phi(t)); %Se llama la función weak_classifier a la cual
se le dan las variables de entrada(inputs), de salida (y_des) y la distribución de pesos de las muestras (D)
%El "WeakLearner" proporciona el modelo que mejor predice la serie, el error y la salida estimada del
aprendiz débil
    beta(t,1) = err^2; %Calcula el peso del WeakLearner de la iteración t
    for j = 1:n%Calcula la distribución de peso de todas las muestras para la iteración t+1
        ARE(j) = abs(y_est(j)-y_des(j))/y_des(j); %Cálculo del ARE para la muestra j
        if ARE(j)<=phi
            D(j,t+1) = beta(t);%Si se cumple, la muestra se considera correcta, el nuevo peso es beta, cuyo
valor siempre será menor a 1
        else
            D(j,t+1) = 1;%Si no se cumple, la muestra se considera errónea, se le da un peso de 1 que
siempre será mayor que las "correctas"
        end
    end
    Z(t) = sum(D(:,t+1)); %Factor de normalización de tal manera que D es una distribución, es decir, la
sumatoria de D es 1
```

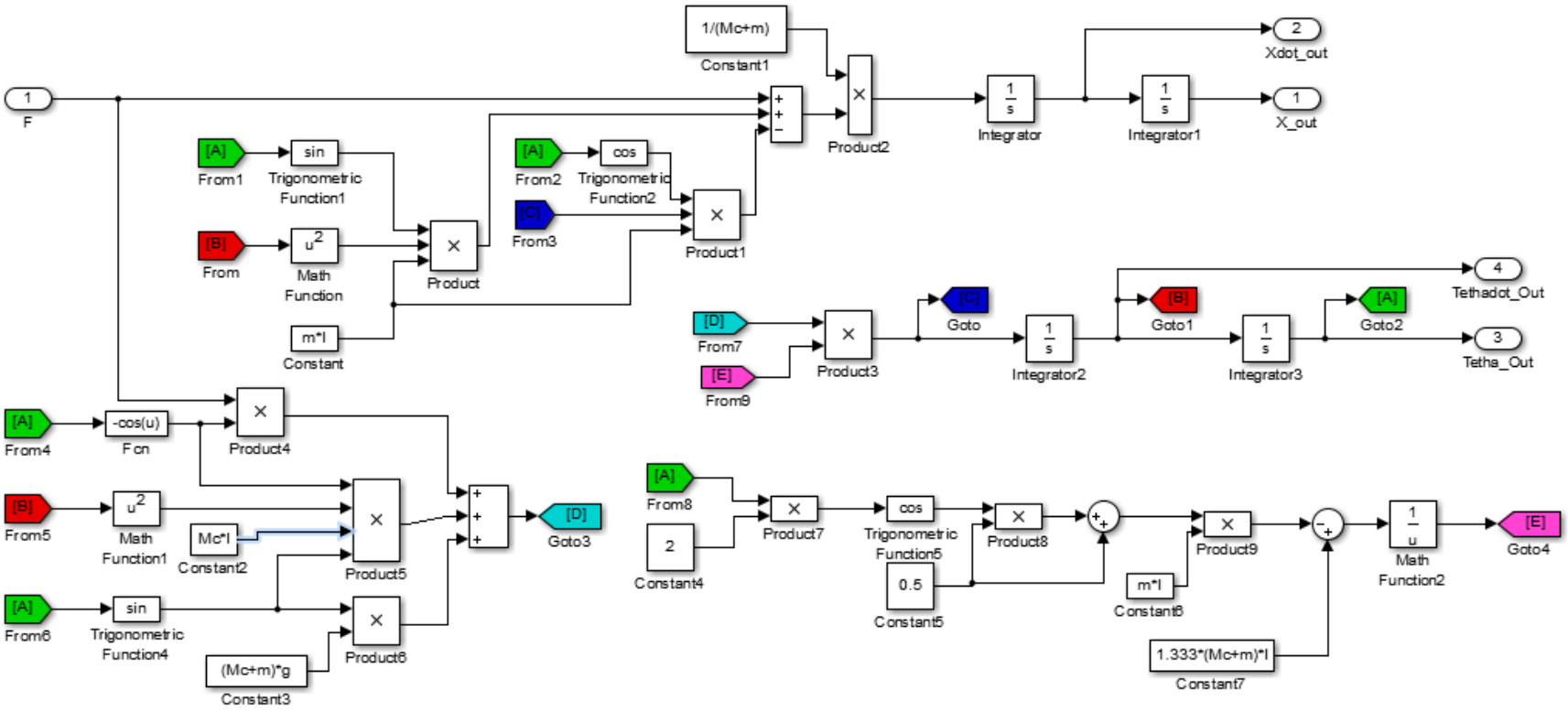
```
    D(:,t+1) = D(:,t+1)/Z(t);
    %Actualización phi
    e(t) = sqrt((1/n)*sum(((y_est-y_des).^2)));%cálculo del  RMSE
    if t>1
        lambda = r*abs((e(t)-e(t-1))/e(t)); %Calculo de lambda para la actualización de phi
        if e(t)>e(t-1)
            phi(t+1) = phi(t)*(1+lambda);
        else
            phi(t+1) = phi(t)*(1-lambda);
        end
    end
    est(:,t) = y_est;
    %Se guardan los parámetros del clasificador débil en la iteración t
    modelo(t).w = w;
    modelo(t).beta = beta(t);
    fac = log(1./beta);
    est_total = (est*fac)./sum(fac); %Cálculo del estimado total del algoritmo hasta la iteración actual
    modelo(t).RMSE = sqrt((1/n)*sum((y_des-est_total).^2));
    modelo(t).MARE = (100/n)*sum(abs((est_total-y_des)./(y_des)));
end
end
```

ANEXO 3 – Función Weak Learner Perceptrón

```
function [best_learner, min_error, y_est]  = weak_learner(inputs,out_des,Di,phi)
%WEAK LEARNER
T = size(inputs);
x(1:T(1),1)= 1;                 %Se agrega el bias
x(:,2:T(2)+1)= inputs;          %Entradas a clasificar
y_des = out_des;                %Salida deseada
n = length(y_des);
%Inicialización aleatoria del vector de pesos w
for i=1:T(2)+1
    an = rand();
    w(i,1) = an;
end
No_iter = 0;
while No_iter<100%Define el número máximo de iteraciones
    No_iter = No_iter+1;
    y_calc = x*w; %Calcula la salida con los pesos actuales
    error =0;
    %Hallar el error del Weak Learner
    for i=1:n
        ARE(i) = abs(y_calc(i)-y_des(i))/y_des(i);%Cálculo de ARE
        if ARE(i)> phi %Si el error es mayor al umbral phi, se suma el error a la tasa de error
            error = error+Di(i);%se hace la suma de los pesos
        end
    end
       lista_error(No_iter,1) = error; %Guarda el error de cada clasificador
       lista_w(No_iter,1:T(2)+1) = w;
    %Seleccionar una muestra mal clasificada con el máximo peso
    k = find(error>=phi);
    [~,idx_max_malclas] = (max(Di(k)));
    malclas = k(idx_max_malclas);
    dato_malclas = x(malclas,:);
    y = y_des(malclas);
    w = w+(y*dato_malclas)';
```

```
end
[min_error, idx_min_error] = min(lista_error); %Guarda el error mínimo y el índice de este error
w = lista_w(idx_min_error,:); %Guarda el modelo del mejor Weak Learner
wt = w';
best_learner = w;
y_est = x*wt; %Calcula la salida con los pesos del mejor Weak Learner
end
```

ANEXO 4. MODELO EN SIMULINK® DEL PÉNDULO INVERTIDO



ANEXO 5. MODELO EN SIMULINK® DEL CONTROLADOR DE REFERENCIA