UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE COMPUTACIÓN

USO DE MÉTODOS EVOLUTIVOS PARA LA PREDICCIÓN DE SERIES DE TIEMPO

Por:
Luis Paúl Serrano Rodríguez

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero de Computación

Sartenejas, Septiembre de 2009

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE COMPUTACIÓN

USO DE MÉTODOS EVOLUTIVOS PARA LA PREDICCIÓN DE SERIES DE TIEMPO

Por:
Luis Paúl Serrano Rodríguez

Realizado con la asesoría de:
Ivette Carolina Martínez

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero de Computación

Sartenejas, Septiembre de 2009

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE COMPUTACIÓN

ACTA FINAL PROYECTO DE GRADO

USO DE MÉTODOS EVOLUTIVOS PARA LA PREDICCIÓN DE SERIES DE TIEMPO

Presentado por:

LUIS PAÚL SERRANO RODRÍGUEZ

Este Proyecto de Grado ha sido aprobado por el siguiente jurado examinador:

Prof. Hugo Montesinos

Prof. Isabel Llatas

Prof. Ivette Carolina Martínez

SARTENEJAS, 15/10/2009

USO DE MÉTODOS EVOLUTIVOS
PARA LA PREDICCIÓN DE SERIES DE TIEMPO

POR
LUIS SERRANO

RESUMEN

Las series de tiempo son secuencias de observaciones sucesivas de un fenómeno. Se encuentran en las más diversas áreas. La predicción de futuros valores de estas series, basadas en observaciones anteriores, es un problema que ha generado interés y que ha sido aproximado tanto con métodos estadísticos como métodos computacionales. Dentro de los métodos computacionales se han obtenido buenos resultados con arquitecturas conexionistas. En esta investigación se plantean dos arquitecturas para enfrentarse a este problema: SERENA y CLARE, que plantean la integración de elementos evolutivos y conexionistas.

SERENA es una arquitectura que utiliza una red neuronal para realizar la predicción, pero no utiliza el tradicional algoritmo de retropropagación, sino que se utiliza una estrategia evolutiva para adaptar los pesos de la misma y aprender la serie; esta arquitectura es una modificación del sistema GRASPES planteado por Lima y Rodrigues. CLARE es un sistema de clasificadores genéticos extendido para realizar aproximación de funciones, con condiciones hiper-elisoipdales, donde la función de predicción de recompensa es sustituida por redes neuronales. Esta es la primera vez que se utiliza este mecanismo para la predicción de series de tiempo.

Para determinar la efectividad de las arquitecturas planteadas, se utilizaron dos series de tiempo. Después de un proceso de entonación de parámetros y de decidir no utilizar el algoritmo de compactación en CLARE, se realizaron experimentos sobre ambas series, donde se observa que tanto SERENA como CLARE logran predecirlas. Es CLARE la arquitectura que mejor predice estas series de tiempo ya que obtiene errores más bajos y de forma consistente.

Palabras Claves: series de tiempo, predicción, redes neuronales, clasificadores genéticos, estrategias evolutivas, XCS, XCSF.

AGRADECIMIENTOS

En cierto modo, esta es la sección más difícil de escribir, pues pienso que un simple agradecimiento "A Dios, a mis familiares y amigos" no es suficiente para expresar mi gratitud a todos aquellos que me dieron su apoyo no sólo durante el año que tomó realizar este trabajo, sino durante toda mi estadía en la Universidad.

En primer lugar, quiero agradecer a Ivette Carolina Martínez, que de tutora pasó a convertirse en una gran amiga que me brindó su apoyo tanto en las buenas como las malas. Quiero agradecer también a todos los profesores que me dieron aliento y me inspiraron a mejorar: Carolina Chang, Jesús Ravelo, Gabi Escuela, Ángela Di Serio, Dionisio Acosta, Edna Ruckhaus, Marlene Goncálvez, Blai Bonet, Oscar Meza y Emely Arraiz.

Gracias también a todos mis compañeros computistas por hacer el trabajo menos pesado y las horas compartidas menos tediosas. Gracias a la gente de BubbleSoft, Beatriz, Juano, Carlos, Moisés, David, Mati, Alex, Ana, Jesireth, y muchos otros que compartieron conmigo algún proyecto de laboratorio.

A todos los de la Comisión de Carrera de Computación les digo que mi vida cambió después de conocerlos y de pasar tantas horas juntos, siempre con algún evento encima, y manejando mil problemas a la vez. Confieso que en un principio no pensé que llegáramos a formar tan buen equipo, pero la realidad derrumbó mi teoría, así que a ustedes, gracias.

A Cristian Caroli y José Dunia, gracias por preocuparse tanto por mi trabajo como por mi, y gracias también por ser personas a quienes admiro y respeto, no sólo por su creatividad y su inteligencia, sino también por su espíritu.

Edith Benítez, Gianfranco Stanzione, Jesús Palacios y Esther Baptista, gracias por estar siempre a mi lado durante esta larga travesía, por sus consejos y su compañía incondicional. Gracias también a todos que tuvieron la paciencia para soportarme en mis momentos de presión, particularmente a Rosa de Serrano y mis tíos Jorge Luis Serrano, Alberto Serrano, William Serrano y José Serrano.

Gracias a los miembros de SAC, por estar siempre conmigo y darme ánimos para seguir adelante. Francis, MaryG, Alej, Steph, Paty, Dada, gracias por las risas y por los pequeños trozos de felicidad que me regalaron.

A Hebert, gracias por la confianza y por esas interminables conversaciones filosóficas a medianoche, donde discutíamos lo bueno y lo malo de la amistad, la sociedad, los disfraces y la vida.

A ti, Gustavo, gracias por ser la prueba viviente de que los hermanos todavía existen, y aunque no compatimos sangre, gracias por el apoyo, el interés y por estar ahí para levantarme cuando pensaba que no tenía fuerzas para continuar.

Gracias a todos por las experiencias compartidas.

ÍNDICE GENERAL

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

LISTA DE ABREVIATURAS

RAM	*Random Access Memory*. Memoria de Acceso Aleatorio.
MB	*Megabyte*.
GB	*Gigabyte*.
MSE	*Mean Squared Error*. Error Cudrático Medio.
DJIA	*Dow Jones Industrial Average*. Índice Dow Jones.
ES	*Evolutionary Strategies*. Estrategias Evolutivas.
XCS	*accuracy-based learning classifier system*. Sistema de clasificadores genéticos basado en precisión.
XCSR	*accuracy-based learning classifier system with real entries*. Sistema de clasificadores genéticos basado en precisión con entradas reales.
XCSF	*accuracy-based learning classifier system for function approximation*. Sistema de clasificadores genéticos basado en precisión para aproximación de funciones.

INTRODUCCIÓN

Una serie de tiempo es una secuencia de observaciones de un fenómeno en intervalos de tiempo sucesivos y generalmente equidistantes entre sí. Las series de tiempo pueden estar en los más diversos lugares, desde los cambios en la presión arterial de un paciente hasta la variación en la intensidad de la luz de una estrella ubicada a millones de años luz de distancia.

Al encontrarse en tantos lugares, el estudio de las series de tiempo se ha vuelto importante, y se encuentran referencias a investigaciones sobre series climáticas, bursátiles, de utilidad eléctrica, aplicaciones de negocios, entre otras [1, 2].

En particular, son de interés el modelaje, caracterización y predicción de futuros valores de la serie. Éste último campo ha sido objeto de diversas aproximaciones, pues obtener un modelo que sea capaz de predecir los siguientes valores de una serie tiene numerosos usos:

- Entender el comportamiento de algún fenómeno complejo y las relaciones que tiene con otros factores, por ejemplo, la variación del ritmo cardíaco dependiendo de la hora del día, concentración de oxígeno en la sangre, estado del paciente, entre otros.
- Prevenir y reducir el impacto de eventos indeseados, determinando, por ejemplo, la cantidad de lluvia que caerá en una zona para evitar inundaciones.
- Obtener ganancias de las predicciones, por ejemplo, decidiendo cuándo comprar y vender de acuerdo al comportamiento esperado del mercado bursátil.

Pero realizar predicciones precisas es un problema complejo, pues es necesario encontrar los patrones y relaciones presentes en la serie, si existen, con una cantidad de datos limitada y que pueden presentar ruido y errores, que además pueden depender de condiciones externas que no han sido consideradas y que no se pueden modelar, como la propagación de un rumor sobre el estado de una compañía, que puede aumentar o disminuir significativamente el precio de sus acciones.

Debido a todas estas dificultades, se han utilizado distintas aproximaciones, tanto matemáticas como computacionales para intentar obtener mejores predicciones. En el campo es-

tadístico destacan los modelos lineales de media variable y autoregresivos, como ARMA y ARIMA, que han sido ampliamente estudiados y sencillos de implementar, pero que tienen dificultades en modelar series muy complejas [3].

En computación, es popular el uso de redes neuronales para el problema de predicción, pues son estructuras capaces de detectar y modelar características de las series de tiempo y relaciones entre los puntos. En la competencia de predicción de series de tiempo del Instituto Santa Fe, los mejores resultados fueron obtenidos por modelos conexionistas [4].

El objetivo general de este trabajo es utilizar una aproximación desde el punto de la computación evolutiva al problema de la predicción de series de tiempo. Existen además, varios objetivos específicos de esta investigación: analizar otros métodos evolutivos utilizados para resolver el problema de predicción, implementar una arquitectura que sea capaz de resolver el problema y estudiar sus resultados sobre series de tiempo de orígenes diferentes.

De acuerdo a estos objetivos y el estudio de diversos métodos evolutivos que podrían ser utilizados, se replantearon las metas del proyecto y se decidió construir dos arquitecturas: una primera que utiliza estrategias evolutivas [5], y una segunda que usa una extensión del sistema de clasificadores genéticos XCS[1], planteado por Wilson [6].

Otra meta es establecer comparaciones del comportamiento de ambas arquitecturas sobre las mismas series de tiempo, y determinar si obtienen mejores resultados que una red neuronal simple con retropropagación, cuyo desempeño servirá como un valor de referencia.

SERENA, una modificación de la arquitectura GRASPES [7], utiliza una red neuronal multicapas con el propósito de realizar las predicciones, tomando como entrada una secuencia de valores sucesivos de la serie para realizar la predicción del siguiente valor. Sin embargo, no se utiliza el método de retropropagación para el entrenamiento de la red neuronal, sino que se usa una estrategia evolutiva para ajustar el vector de pesos.

En esta estrategia, se aplica en cada iteración una mutación que consiste en sumar a cada peso un valor obtenido de una distribución normal dependiente de un valor auto-adaptativo y se evalúa la nueva red, que sustituye a la anterior si realiza mejores predicciones.

[1]XCS: *accuracy-based learning classifier system*. Mecanismo de aprendizaje de clasificadores basado en precisión.

La segunda arquitectura, CLARE, usa una modificación de los clasificadores genéticos denominada XCSF[2]diseñada por Wilson [8] con el propósito específico de realizar aproximaciones a funciones, usando las condiciones de los clasificadores para dividir el espacio y las predicciones de los mismos para la aproximación.

Mientras que en XCSF se utilizan funciones lineales para aproximar las funciones, en CLARE, inspirada en trabajos anteriores de Loiacono y Lanzi[9], se sustituyen estas por redes neuronales, debido a la compleja naturaleza de algunas series de tiempo, que evitarían que rectas lograran formar buenas predicciones.

En esta arquitectura se toma una secuencia de valores y son procesadas por todas las redes neuronales correspondientes a los clasificadores en los que corresponda este estado, para obtener la predicción del sistema como el promedio de los valores arrojados por estas redes, y se realiza retropropagación sobre ellas utilizando el valor del próximo punto de la serie.

Para evaluar el desempeño de las arquitecturas, se escogieron dos series de tiempo distintas, la primera corresponde al precio de cierre del DJIA[3]en un período de tiempo [10] y la segunda a la temperatura máxima diaria detectada en una estación meteorológica de Australia [11]. Estas series fueron seleccionadas por tener diferentes orígenes, la primera depende del comportamiento del mercado bursátil, mientras que la segunda es de origen natural.

De acuerdo a estos análisis, se logró concluir que tanto las dos arquitecturas planteadas como la red neuronal son capaces de realizar predicciones certeras sobre ambas series de tiempo, siempre que sean escogidos parámetros adecuados para las mismas. También se determinó que bajo estas condiciones, el desempeño de CLARE sobrepasa a los otros dos sistemas. Adicionalmente, se mostró que CLARE obtiene buenas predicciones constantemente, mientras que SERENA y la red neuronal pueden presentar, en el curso de varias ejecuciones, errores atípicamente altos.

Este informe consta de cinco capítulos. En el Capítulo 1 se presentan conceptos básicos relacionados con el estudio, como el de redes neuronales, el algoritmo de retropropagación, las bases de las estrategias evolutivas y una introducción a XCSF. El Capítulo 2 describe el diseño de SERENA y de las decisiones tomadas en el mismo. El Capítulo 3 presenta a CLARE, su

[2]XCSF: *accuracy-based learning classifier system for function approximation.* Mecanismo de aprendizaje de clasificadores basado en precisión que se utiliza para realizar aproximaciones de funciones

[3]DJIA: *Dow Jones Industrial Average.* Promedio Industrial Dow Jones, es un índice del mercado bursátil

estructura y funcionamiento, se describe también su implementación y las modificaciones hechas a la librería de XCSF. El Capítulo 4 presenta las series de tiempo utilizadas para evaluar el desempeño de las arquitecturas, los experimentos hechos y los resultados de los mismos. Finalmente, el Capítulo 5 presenta las conclusiones obtenidas y direcciones para trabajos futuros.

CAPÍTULO 1

MARCO TEÓRICO

En este capítulo se presenta el problema de predicción de series de tiempo, sobre el cual se desarrolló este trabajo. También se presentan conceptos básicos de redes neuronales y el algoritmo de retro-propagación.

Además se dan las bases de las estrategias evolutivas, que son utilizadas para la construcción de SERENA, una arquitectura que utiliza una red neuronal evolucionada con este método para realizar la predicción.

Se da también una breve introducción al sistema de clasificadores XCS[6] y su extensión, XCSF[8], que constituye la base de CLARE, una arquitectura que utiliza este sistema, mezclado con redes neuronales para predecir las series.

1.1 Series de Tiempo

Una serie de tiempo es una secuencia de observaciones de un fenómeno en intervalos de tiempo sucesivos y generalmente equidistantes entre sí, por lo que se pueden estar en los más diversos áreas: medicina, astronomía, física, economía, entre otras.

Al encontrarse en tantos lugares, el análisis de las series de tiempo se ha vuelto importante, enfocándose en tres aspectos: caracterización, que consiste en determinar propiedades de una serie de tiempo teniendo poca información sobre la misma; modelaje, en donde se intenta conseguir una descripción del sistema que permita captar cualidades sobre su comportamiento; y predicción, que consiste en obtener, con la mayor precisión posible, los siguientes valores que tomará la serie. [4]

Técnicas básicas para la caracterización de una serie consiste en realizar gráficas de la misma, para observar su comportamiento y la presencia de patrones o muestras de periodicidad.

El modelaje y la predicción están muy relacionadas entre sí, pues obtener un buen modelo de una serie permitirá realizar predicciones precisas. Estas tareas son aproximadas comúnmente por medios estadísticos, con modelos autoregresivos [3]; y por medios computaciones, generalmente con métodos de inteligencia artificial y aprendizaje de máquinas, por ejemplo, redes neuronales[4] y máquinas de soporte vectorial [2].

Con el propósito de hacer más ordenado el análisis de una serie de tiempo, se han establecido diversos atributos a las mismas, para que sea más fácil clasificarlas y compararlas con series similares. De estos atributos, son de interés particular para esta investigación dos de ellos:

- Origen: se refiere a la procedencia de la serie. Puede ser tan específica como el área de donde proviene, pero básicamente se puede decir que una serie es natural, si se obtiene de algún fenómeno natural; o sintética, si viene de alguna ecuación o es generada de forma artificial para modelar alguna situación.
- Cantidad de Variables: se refiere a la cantidad de datos que se obtiene en una observación. Se dice que una serie es univariada si solamente se tiene un dato por cada observación, y multivariada si se toman varios valores en una observación.

En esta investigación se trabaja sobre series de tiempo univariadas, que generalmente son representadas como:

$$X_t = \{x_t \in R | t = 1, 2, ..., N\} \tag{1.1}$$

Donde x_t es el valor observado, t es el índice temporal y N es el número de observaciones

hechas.

1.2 Redes Neuronales

Una red neuronal artificial es un modelo diseñado para simular la forma en que el cerebro realiza una tarea o función particular, imitando a las redes neuronales biológicas. Este modelo adaptativo es utilizado para almacenar conocimiento mediante un proceso de aprendizaje, que cambia su estructura y los pesos de las conexiones entre las neuronas para adquirir el conocimiento[12].

Las redes neuronales son utilizadas para tareas muy diversas, que incluyen: clasificación, reconocimiento de patrones, compresión de datos, aproximación de funciones y manipuladores en robótica, entre otros. Han sido ampliamente utilizadas porque presentan varias ventajas, como su flexibilidad, capacidad de aproximar funciones no lineales y potencial de paralelización; sin embargo, presentan una clara desventaja, que la información guardada es muy difícil de interpretar para un ser humano.

Una red neuronal está formada, básicamente, por unidades neuronales y un conjunto de conexiones entre ellas, cada una caracterizada por un peso propio. Las neuronas realizan una combinación lineal de las distintas señales de entrada, sumando sus valores multiplicados por la conexión de la que vienen, y le suman luego el valor del umbral, pasando posteriormente este valor por una función de activación, cuyo propósito es limitar el rango de la señal de salida de la neurona. Esta estructura se puede apreciar en la Figura 1.1.

Luego, matemáticamente, la salida de una neurona k se puede representar matemáticamente con la fórmula:

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj}x_j + b_k\right) \tag{1.2}$$

Donde y_k es la salida de la neurona k, w_{kj} es el peso que conecta la neurona k con la neurona j, x_j es la señal recibida de la neurona j y b_k es el umbral de la neurona k.

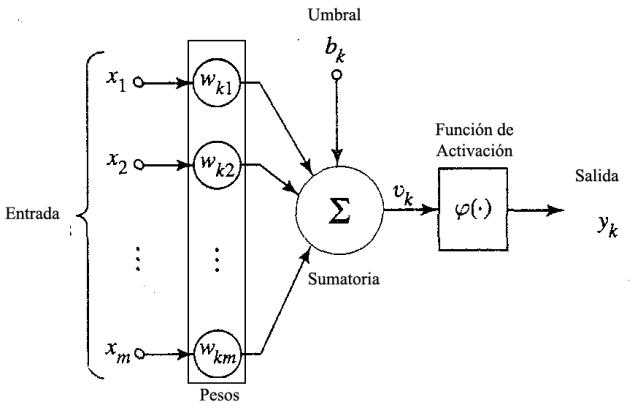

Figura 1.1: Funcionamiento de una neurona no lineal, como es presentada por Haykin[12]

Existen varios tipos de funciones de activación, pero la más común es la función sigmoidal, que presenta un comportamiento estrictamente creciente y que limita los valores al intervalo $[0; 1]$. La fórmula de la sigmoidal es:

$$f(x) = \frac{1}{1 + \exp\{-x\}} \tag{1.3}$$

Una neurona sigmoidal tiene la característica de que cuando la activación es fuertemente positiva, su salida son valores cercanos a 1. En caso de ser fuertemente negativa, la neurona responde con valores cercanos a 0.

Es fácil visualizar gran cantidad de estas neuronas interconectadas entre sí, y es de interés de este trabajo describir las redes neuronales multicapas, que no son más que redes organizadas de tal forma que poseen conjuntos de neuronas organizadas en capas, y que poseen una capa de entrada, una capa de salida y una o más capas ocultas, que son donde se realiza la mayor parte del aprendizaje. Un ejemplo de una red neuronal de 3 capas se muestra en la Figura 1.2.

En estas redes, la señal recibida pasa por la capa de entrada, donde son distribuidos a las neuronas ocultas, que los procesan para luego enviar los resultados a las neuronas de salida. Estas redes tienen capacidad de aproximar funciones y superficies más complejas que una sola neurona, pero determinar la cantidad de capas ocultas y el número de neuronas en cada una de ellas es un proceso complicado, más un arte que ciencia, que requiere experiencia y experimentación.

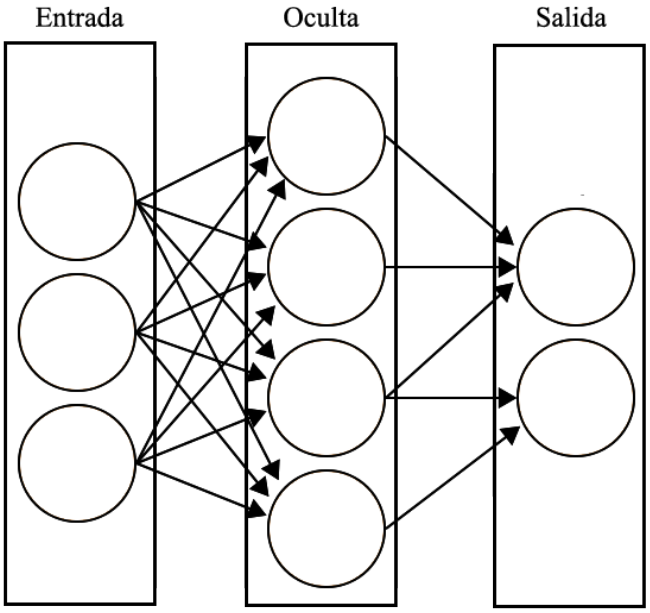

Figura 1.2: Red Neuronal de 3 Capas

1.2.1 Algoritmo de Retro-propagación

Para el aprendizaje en redes neuronales, es comúnmente utilizado el algoritmo de retro-propagación. La idea básica, como lo plantea Michalewicz [13] consiste en tomar el resultado de la red neuronal y ajustar los pesos de la misma de tal forma que se reduzca el error entre este resultado obtenido y el esperado.

El algoritmo de retro-propagación pretende reducir el error cuadrático total del modelo alterando el vector de pesos en proporción al error entre el valor obtenido y el valor esperado. Para este ajuste, se suma a cada peso w_{ji} una variación Δw_{ij}, que se calcula de acuerdo a la fórmula:

$$\Delta w_{ij} = \eta \delta_j(p) y_i(p) \tag{1.4}$$

Donde j es el nodo de salida, i recorre el número de nodos en la capa anterior $l-1$, η representa la tasa de aprendizaje, $y_i(p)$ es la entrada que viene de la neurona i y $\delta_j(p)$ es el gradiente local del nodo j para el ejemplo p.

El gradiente para las neuronas de salida, dada una función de activación sigmoidal, es:

$$\delta_j(p) = e_j y_j(p)(1 - y_j(p)) \tag{1.5}$$

donde e representa a la diferencia entre el valor esperado y el obtenido por la neurona de salida y $y_j(p)$ es la entrada que viene de la neurona i .

El problema es ajustar entonces los pesos que conectan las capas ocultas, pues no se tiene información sobre el error relacionado con cada neurona oculta, sin embargo, se puede derivar que el valor $\delta_j(p)$ de estas neuronas se corresponde a:

$$\delta_j(p) = y_j(p)(1 - y_j(p)) \sum_k \delta_k(p) w_{kj}(p) \tag{1.6}$$

donde k va de 1 al número de nodos de salida, o en el caso de varias capas ocultas, al número de neuronas en la capa siguiente. La sumatoria es esencialmente el gradiente pesado asociado con cada nodo. Una vez se obtienen todos los valores delta, se procede a ajustar los pesos, desde los que llegan a la capa de salida hacia atrás, sumando a cada uno el valor obtenido usando la Ecuación 1.4.

Este proceso se repite para cada uno de los patrones de entrada del conjunto de entrenamiento. Este algoritmo siempre converge, pero no existe garantía de que llegue al mínimo global, sino que el vector de errores puede quedarse estancado en un mínimo local, una limitación del algoritmo de retro-propagación

1.2.2 Construcción de Ejemplos

Para la construccíon de los ejemplos que serán alimentados a la red neuronal, en el caso de las series de tiempo se fija el tamaño de la ventana de predicción, que es la cantidad de valores previos necesarios para obtener el próximo valor de la serie.

De esta forma, en el caso de tener una serie de tiempo $< t_1, t_2, \ldots, t_{n-3}, t_{n-2}, t_{n-1}, t_n >$, si fijamos una ventana de predicción de tamaño 1, entonces tendríamos un ejemplo de la forma $\{t_k | t_{k-1}\}$, donde tendríamos que obtener t_k dado el valor de t_{k-1}. Si fijamos la ventana de predicción de tamaño 2, entonces tendríamos un ejemplo de la forma $\{t_k | t_{k-1}, t_{k-2}\}$, donde

tendríamos que obtener t_k dado los valores de t_{k-1} y t_{k-2}.

El tamaño máximo de la ventana de predicción viene dado por la cantidad de puntos conocidos previamente. Nótese que la cantidad de nodos en capa de entrada de la red neuronal debe ser igual al tamaño de la ventana de predicción previamente fijado.

1.3 Estrategias Evolutivas

Los algoritmos genéticos son un método de aprendizaje cuya idea esencial está basada en una analogía a la evolución biológica. Se mantiene una población de soluciones candidatas que son recombinadas y mutadas para generar hijos, escogiendo los mejores para sustituir una parte de la población original.

De acuerdo a Tom Mitchell[14], existen varios factores que motivan la popularidad de los algoritmos genéticos:

- La evolución ha probado ser un sistema exitoso en biología, que ha permitido la adaptación y mejora de las especies.
- Los algoritmos genéticos son capaces de explorar espacios de soluciones complejas, con interacciones difíciles de modelar.
- Los algoritmos genéticos son fácilmente paralelizables.

El mecanismo general de funcionamiento de los algoritmos genéticos consiste en actualizar iterativamente una población de soluciones. En cada iteración se determina la utilidad de cada individuo de acuerdo a una función de evaluación[1]que se determina de acuerdo al problema. Posteriormente se genera una nueva población seleccionando probabilísticamente los mejores individuos, algunos de los cuales pasan a la siguiente generación idénticos y otros son usados para generar nuevas soluciones mediante los operadores de cruce y mutación.

El operador de cruce toma dos individuos, y los recombina con el propósito de generar mejores soluciones. Por otro lado, el operador de mutación toma un individuo y realiza pequeños cambios en él para así descubrir nuevas soluciones.

[1]Comúnmente llamada función de *fitness*, es una medida de calidad del cromosoma, que determina lo que significa una mejora

Las estrategias evolutivas pertenecen a este tipo de algoritmos, por lo que poseen las características básicas de este grupo: una población de cromosomas, operador de cruce, operador de mutación, mecanismos para selección de padres, mecanismos de reemplazo y función de evaluación.

Las también llamadas ES[2], introducen los parámetros auto-adaptativos, que son parámetros que modifican el funcionamiento del algoritmo y evolucionan junto a los cromosomas durante el proceso normal de recombinación y mutación.

Las estrategias evolutivas se caracterizan por el fuerte énfasis que hacen en el operador de mutación, pues es el operador principal utilizado para obtener los descendientes y los parámetros de la misma son cambiados durante la ejecución.

La estructura general de la ES simple es descrita por Back, Michalewicz y Fogel[15], donde dado un individuo X, un vector de reales de longitud n escogidos aleatoriamente, éste es mutado sumando un vector Z de longitud n, en el que cada componente es un valor extraído de una distribución Gaussiana de media 0 y desviación estándar σ. Este valor de σ es conocido como el tamaño de la mutación y es el parámetro auto-adaptativo que siempre se utiliza en las estrategias evolutivas, por lo que la amplitud de la mutación coevoluciona junto con el vector X.

El vector obtenido después de la mutación es aceptado si es mejor que el anterior, tomando esta decisión basados en una comparación de la función de evaluación f sobre ambos individuos. Los individuos aceptados son colocados en un conjunto del que posteriormente se seleccionan los padres para la recombinación, que puede ser intermediaria o discreta; en la primera cada componente del vector hijo toma el promedio del componente correspondiente de los vectores padres, y en la segunda, cada componente es escogido aleatoriamente de uno de los padres. Posteriormente, se toman de este conjunto una cierta cantidad de sobrevivientes que pasarán a la siguiente generación y se toma el mejor de ellos para ser mutado y recomenzar el proceso.

Las estrategias evolutivas trabajar bajo la suposición que diferentes tamaños de mutación se comportarán diferente de acuerdo al individuo, el momento de la evolución, el espacio de posibles soluciones y otras circunstancias, por lo que un σ que cambie de acuerdo al individuo

[2]ES :*Evolution Strategies* o Estrategias Evolutivas en español

abre la posibilidad de aprender una mutación adecuada al problema en particular.

1.4 Sistema de Clasificadores Genéticos

Propuestos por Holland[16], los sistemas de clasificadores genéticos son una aproximación evolutiva al aprendizaje de máquinas, que aprende mediante interacción con un ambiente, del que recibe respuesta en forma de recompensa.

De acuerdo a Holmes, Lanzi, Stolzmann y Wilson [17], un sistema de clasificadores tiene cuatro componentes principales, que se observan en la Figura 1.3:

- Una población finita de reglas condicionales de forma "Si se cumple (condición) entonces se realiza (acción)", que almacenan el conocimiento adquirido. Estas reglas son también llamadas clasificadores.
- Un componente de desempeño, que determina los clasificadores a utilizar y se encarga de la interacción con el ambiente.
- Un componente de reforzamiento, que se encarga de distribuir la recompensa obtenida del ambiente por el sistema entre los clasificadores responsables de adquirirla.
- Un componente de descubrimiento, que utiliza un algoritmo genético para explorar el espacio de soluciones, buscando descubrir buenas reglas y mejorar las existentes.

Los sistemas de clasificadores son utilizados en aplicaciones donde el objetivo principal es evolucionar un sistema que responda a un estado del ambiente sugiriendo una acción que obtenga la recompensa máxima; luego, en el caso ideal, se tendría un conjunto de reglas que cubran todos los estados posibles y que indique la acción más apropiada para cada uno de ellos.

Sin embargo, la propuesta original de clasificadores presentaba problemas para evolucionar generalizaciones precisas, obtenía algunas reglas con poco significado y no tenía motivación para construir reglas que cubrieran todos los posibles estados. En respuesta a estos inconvenientes, surgieron modificaciones del algoritmo, una de ellas denominada XCS[3], propuesta por

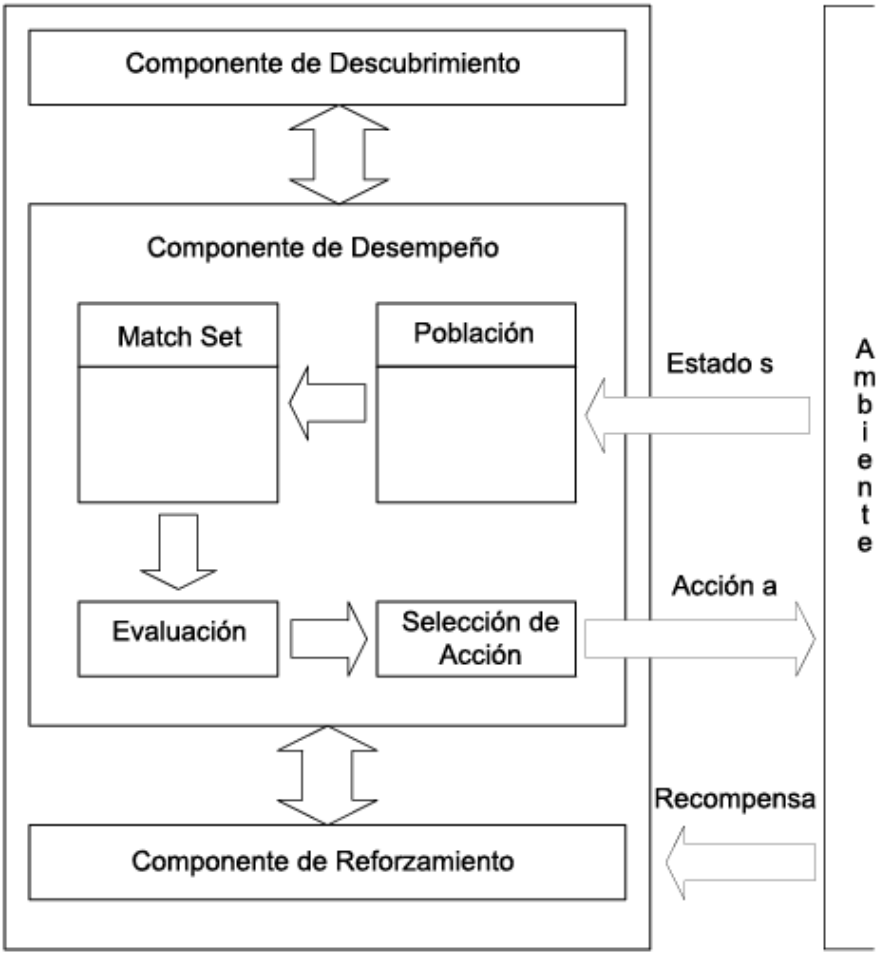

Figura 1.3: Componentes de un Sistema de Clasificadores Genético. Tomado de [17]

Wilson[6], que es capaz de resolver problemas más complejos y producir modelos compactos, con reglas generales y precisas.

XCS introduce dos cambios fundamentales en la estructura de los clasificadores genéticos:

- Se establece que la función de utilidad de un clasificador no depende directamente de su predicción, sino que se basa en la exactitud de las mismas. De esta forma, la recompensa esperada se utiliza al momento de considerar la acción, pero un clasificador se considera bueno si contiene información precisa sobre el problema. Este cambio favorece la cobertura de todos los posibles estados, pues mantiene las reglas para los

[3]XCS: *accuracy-based learning classifier system*. Mecanismo de aprendizaje de clasificadores basado en precisión.

malos escenarios, aunque la recompensa sea muy baja.

- El algoritmo genético usado en el componente de descubrimiento no actúa sobre toda la población de clasificadores, sino sobre subconjuntos que aplican en el mismo estado, para obtener buenas reglas sobre situaciones en particular.

1.4.1 Conceptos Básicos de XCS

Antes de describir el funcionamiento de XCS, es importante hacer una reseña de conceptos y mecanismos importantes para entender este algoritmo. En primer lugar, hay que considerar tres conjuntos que son definidos como:

- Población: conjunto de clasificadores en el sistema
- *Match Set*: conjunto de clasificadores presentes en el sistema que se ajustan al estado actual proporcionado por el ambiente.
- *Action Set*: conjunto de clasificadores pertenecientes al *match set* que proponen la acción escogida.

Es también importante describir los clasificadores, que son tuplas que contienen cinco atributos básicos:

- Condición: indica cuáles son los estados cubiertos por el clasificador.
- Acción: es la acción que la regla propone utilizar.
- Predicción: estimado de la recompensa que se obtendrá al aplicar la acción propuesta.
- Estimación del error de la predición
- Utilidad del clasificador

Además, cada clasificador contiene otros atributos adicionales: la experiencia, que corresponde a la cantidad de veces que un clasificador ha estado en el *action set*; una marca de tiempo que indica la última vez que se ejecutó el algoritmo genético sobre el clasificador; el

tamaño promedio los *action sets* y la numerosidad[4]del clasificador. Se debe tener en cuenta que el término "clasificador" usado en XCS se refiere realmente a macroclasificadores, que representan a varios clasificadores idénticos llamados microclasificadores.

1.4.2 Funcionamiento de XCS

En esta sección se presenta el algoritmo XCS, como es descrito por Butz y Wilson[18]. El primer paso consiste en la inicialización del sistema, donde se asignan valores a las variables y se crea la población inicial; que puede ser vacía o llenarse con clasificadores aleatorios. Se inicializan los componentes y los parámetros del ambiente. Una vez se haya terminado esta fase, se entra en un ciclo principal, que consta de los siguientes pasos:

1. El sistema percibe el estado actual del ambiente.
2. Se crea el *match set*, revisando las condiciones de todos los clasificadores. Si la condición es satisfecha por el estado actual, entonces se agrega al conjunto. Si la cardinalidad de este conjunto es menor que cierto umbral, se realiza la creación de clasificadores que cubran este estado.
3. Sea A' el conjunto de acciones presentes en el *match set*. Se crea un arreglo de predicciones, donde se calcula la predicción de la recompensa que se obtendrá del ambiente al ejecutar cada acción de A', que se calcula, para cada $a \in A'$ y R_a el subconjunto de clasificadores en el *match set* que sugieren a, con f_r la utilidad del clasificador r, y se define:

$$p_a = \sum_{r \in R_a} w_r p_r \quad \text{con} \quad w_r = \frac{f_r}{\sum_{r \in R_a} f_r} \tag{1.7}$$

4. Se escoge la acción a ejecutar
5. Se genera el *action set*, con los clasificadores del *match set* que sugieran la acción escogida en el paso anterior.
6. Se ejecuta la acción escogida y se recibe la recompensa del ambiente.
7. Se actualizan los parámetros de los clasificadores en el *action set* de la iteración anterior. En primer lugar se determina la remuneración P, con R la recompensa recibida en la

[4]Representa la cantidad de clasificadores que cubren la misma condición, que no son almacenados directamente para mejorar la eficiencia del sistema.

iteración anterior y γ un factor de descuento, que es fijado al crear el modelo y que toma valor 1 para problemas en los que la recompensa se obtenga inmediatamente después de ejecutar la acción.

$$P = \gamma \, \text{máx}\{p_a \, | a \in A'\} \tag{1.8}$$

Posteriormente, utilizando este valor P se actualizan la experiencia, error, predicción y utilidad de cada uno de los clasificadores en el conjunto].

8. Se ejecuta el algoritmo genético sobre el *action set* anterior si la diferencia entre el número de iteración actual y el número de iteración promedio de la última aplicación sobre estos clasificadores supera un umbral determinado.

1.4.3 Consideraciones importantes sobre XCS

Una consideración importante se refiere a la selección de la acción a realizar. En el algoritmo de XCS no se describe el criterio para escoger la acción a ejecutar entre todas las posibles sugeridas por los clasificadores del *match set*. Dos criterios comunes para esta selección es explotación, que consiste en escoger la de mayor predicción, y exploración, donde se escoge una aleatoriamente con distribución uniforme.

Un mecanismo importante a considerar es la subsunción de clasificadores. Se dice que un clasificador subsume a otro si la condición del primero contiene a la condición del segundo, lo que indica que es también más general que este; se puede entonces eliminar el clasificador más específico y aumentar la numerosidad del general. Esto permite disminuir el tamaño de la población y tener un conjunto más conciso de reglas.

La subsunción es utilizada durante el algoritmo genético, donde se verifica si los padres subsumen a sus hijos, y durante la creación del *action set*, donde se verifican sus miembros para determinar si existe alguno que sea más general, y de acuerdo a umbrales de precisión y eficiencia definidas en el modelo, éste subsume al más específico.

1.4.4 Extensiones y Modificaciones de XCS

El sistema de clasificadores XCS tiene la limitación de que sus condiciones de entrada son cadenas de caracteres que representan booleanos, lo que evita que pueda ser utilizado en problemas que no admitan esta representación. Esta restricción del sistema original fue superada realizando una extensión que permite a XCS manejar valores reales [19].

Otra extensión fue hecha a XCS para permitir que en vez de seleccionar una acción, permitiera aproximar funciones. Como la función de utilidad de XCS se basa en la precisión, es natural que en este caso evolucionara clasificadores que obtuviesen aproximaciones precisas de la función objetivo.

Adicionalmente, se planteó la hipótesis de que las condiciones hiper-rectangulares originalmente planteadas podrían no ser adecuadas para particionar ciertos espacios de problema, por lo que se decide extenderlo utilizando condiciones hiper-elisoipdales que tienen la capacidad de expandirse y rotar en el espacio de búsqueda para lograr una mejor aproximación.

XCSR

Originalmente, el sistema de clasificadores XCS recibe como entrada una cadena de caracteres consistentes en unos, ceros y un caracter especial utilizado como *don't care*[5], lo que limita las variables de estado a valores booleanos; sin embargo, en muchas aplicaciones, el estado es mejor descrito por valores reales.

Para solucionar esto, Wilson[19] recomienda que cada variable de la condición consista en una tupla $< centro, radio >$, que represente un rango de valores $centro \pm radio$. Luego, un estado cumple con una condición si toda variable de mismo está en el rango de valores correspondiente. Esta extensión es conocida como XCSR[6].

Aunque esta representación permite tener variables continuas, Bull y Stone [20] prueban que introduce un sesgo que empeora el desempeño del sistema, y que mejores resultados se pueden obtener con condiciones hiper-rectangulares, que son pares $< l_i, u_i >$ donde l_i y u_i representan la cota inferior y superior del intervalo, respectivamente.

El funcionamiento general de XCSR se observa en la Figura 1.4:

[5]don't care: caracter que no es tomado en cuenta al momento de determinar si un estado cumple con la condición del clasificador.

[6]XCSR: *accuracy-based learning classifier system with real entries*. Sistema de clasificadores genéticos basado en precisión que permite como entrada estados reales

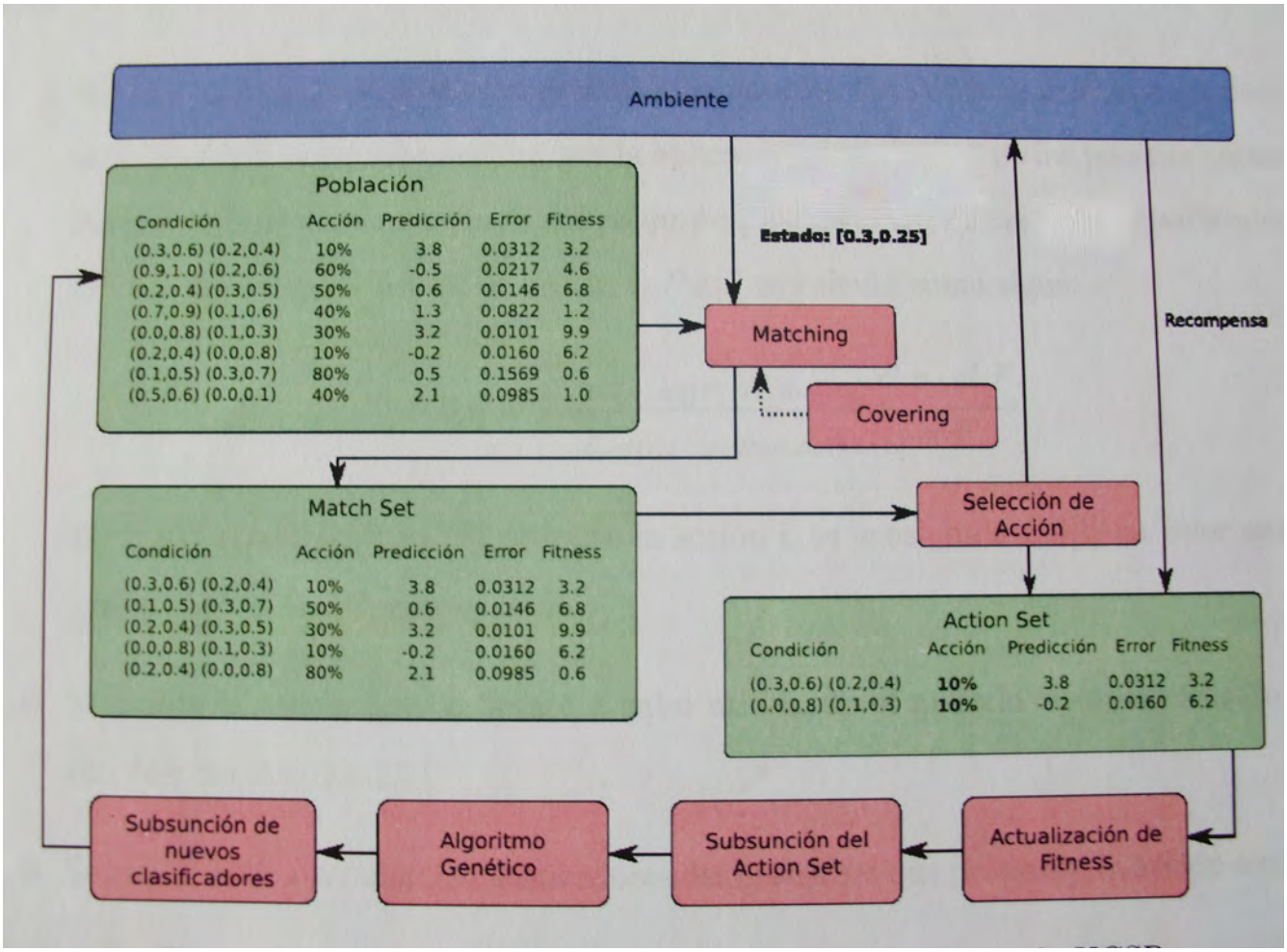

Figura 1.4: Esquema general de la ejecución del algoritmo XCSR. Tomado de [21]

XCSF

En varios problemas sobre los que se usa XCS, se utiliza la función de recompensa esperada para permitir al sistema escoger, dado un estado, la acción que genere más compensación por parte del ambiente. Sin embargo, bajo algunos contextos, lo que se desea obtener del sistema no es una acción discreta sino un valor continuo.

Bajo esta premisa, Wilson[8] propone adaptar la capacidad de XCS de aproximar la función de recompensa para que el sistema arroje valores reales. Así surge XCSF[7], una modificación de XCS que permite la aproximación de funciones.

[7]XCSF: *accuracy-based learning classifier system for function approximation*. Mecanismo de aprendizaje de clasificadores basado en precisión para aproximación de funciones.

Para obtener XCSF, es necesario realizar algunas modificaciones sobre XCS:

- Se adapta el programa para recibir entradas con valores reales, usando XCSR.
- Se restringe el conjunto de acciones a una sola acción ficticia.
- Se hace que el resultado del sistema no sea una acción sino el valor de la predicción de la recompensa.

En XCS y sus variaciones, la medida de valor de un clasificador está dada por su precisión, por lo que XCSF debería converger a una población de clasificadores que hagan buenas estimaciones de la recompensa en sus respectivos rangos de entrada.

La forma más sencilla de predecir la función $y = f(x)$ es hacer x el estado que entra al sistema, con y como la recompensa. Con este esquema, XCSF realiza lo que se conoce como predicción constante por trozos, pues para cada partición del dominio, tendrá como resultado un valor constante dado por la función de recompensa esperada.

En la Figura 1.5 se puede apreciar la aproximación obtenida con este sistema sobre la función de la parábola $y = x^2$, siendo claro que no es muy precisa por estar formada por trozos constantes.

Sin embargo, se desea que la predicción no sea constante, sino que varíe en el dominio del clasificador para adaptarse a la pendiente de la función en el trozo correspondiente. Luego, se propone que la predicción escalar sea sustituída por una función $h(x)$ no constante en cada uno de los clasificadores.

La forma más simple de $h(x)$ sería entonces una función lineal sobre los valores de entrada; es decir, para una aproximar una función $f(x)$, $h(x) = w \cdot x'$, donde $w = (w_0, w_1, ..., w_n)$ es el vector de pesos y $x' = (x_0, x_1, ..., x_n)$ es el vector de entrada aumentado con una constante x_0.

Con este cambio, dado un estado x, cada clasificador en el *match set* calcularía su predicción utilizando la función $h(x)$ propia y cada uno de ellos tendrían vectores de pesos diferentes para lograr que h aproxime mejor a la función original f en el dominio correspondiente.

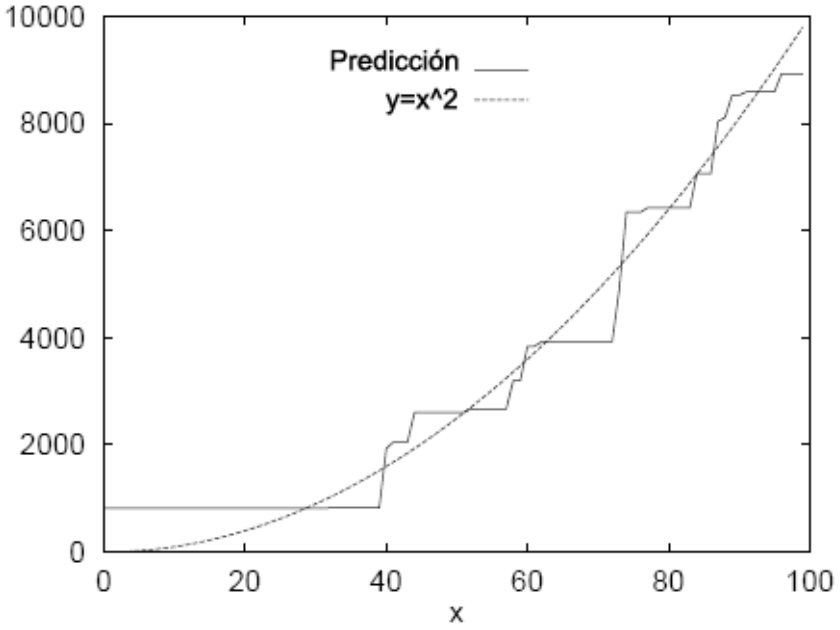

Figura 1.5: Predicción constante por trozos sobre la función $y = x^2$. Tomado de [8]

En la Figura 1.6 se observa que esta sustitución permite que XCSF logre una mejor aproximación sobre la función de la parábola $y = x^2$.

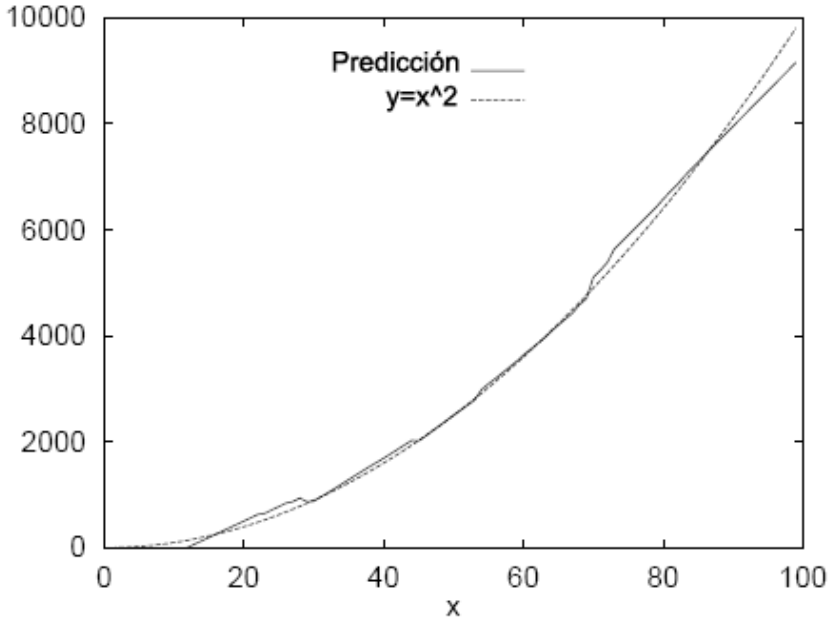

Figura 1.6: Predicción lineal por trozos sobre la función $y = x^2$. Tomado de [8]

Para lograr buenas predicciones es necesario adaptar el vector de pesos, y en XCSF se utiliza una modificación de la regla delta"[8][22] para corregir el valor de la predicción:

$$\Delta w_i = (\eta/|x'|^2)(t-o)x_i \tag{1.9}$$

donde w_i y x_i son los i-ésimos componentes de w y x', respectivamente; t es el valor esperado, en este caso, el valor correcto y de acuerdo a la función $y = f(x)$, o es la predicción obtenida y η es la tasa de aprendizaje.

Trabajos posteriores se han enfocado en utilizar funciones diferentes para generar la predicción. Lanzi, Loiacono, Wilson y Goldberg [23] plantean sustituir la aproximación lineal por funciones polinomiales cuadráticas y cúbicas. Loiacono y Lanzi [9] proponen posteriormente utilizar redes neuronales para realizar la predicción, y luego sugieren el uso de máquinas de soporte vectorial con el mismo propósito [24].

Condiciones Hiper-elisoipdales

Investigando si las condiciones hiper-rectangulares son adecuadas para particionar el espacio de búsqueda, Butz[25] plantea utilizar condiciones hiper-elisoipdales para para lograr una mejor aproximación, obteniendo como resultado que es igual o más efectivo en las funciones examinadas.

La condición general hiper-elisoipdal es representada por su punto central y una matriz de transformación que determina el tamaño y la rotación de la misma. Siendo $\vec{m}$ el punto central y $\Sigma = [\sigma_{i,j}]$ una matriz de valores, se tiene que la condición es:

$$C \;=\; (\vec{m}, \Sigma) \;= ((m_1, m_2, ..., m_n)^T, \sigma_{1,1}, \sigma_{1,2}, ..., \sigma_{n,n-1}, \sigma_{n,n}) \tag{1.10}$$

En un trabajo posterior del mismo Butz[26], se muestra que estas condiciones también permiten obtener buenos resultados en la aproximación de funciones usando XCSF. Además,

[8]Regla Delta: es una regla de aprendizaje basada en descenso del gradiente. Es generalmente utilizada para actualizar los pesos en redes neuronales y es la base del algoritmo de retro-propagación.

concluye que la matriz de transformación contiene información redundante, por lo que se sustituye la misma por los ángulos de Euler, que permiten computar la matriz rápidamente.

La actividad de un clasificador hiper-elisoipdal se determina de acuerdo a la fórmula:

$$cl.ac = \exp\left\{-\frac{(\vec{x}-\vec{m})^T\Sigma^T\Sigma(\vec{x}-\vec{m})}{2}\right\} \tag{1.11}$$

donde $\vec{x}$ representa un vector con las variables obtenidas del estado del sistema. Se dice que un clasificador es activo, es decir, pertenece al *match set*, si $cl.ac$ es mayor que un umbral θ_m definido al realizar el modelo.

Al momento de cubrir un estado, el centro de la hiper-elipsoide se coloca en el valor actual de $\vec{x}$ y los valores diagonales de la matriz se inicializan con el cuadrado del inverso de un número aleatorio distribuido uniformemente entre 0 y 1.

Durante la subsunción, se considera que una hiper-elipsoide A es más general que una hiper-elipsoide B, si A contiene el punto más lejano al centro de A, en el cual la línea que pasa por los centros de A y B intersecta a la superficie de B. En la Figura 1.7 se tiene la elipse blanca A, que puede subsumir a B1 y B2, pero no a C1, C2 o C3.

Compactación

A pesar de la confiabilidad de XCSF y de la precisión en la aproximación de funciones, el tamaño de la población final de XCSF sugiere la idea de que las funciones pueden estar sobre-representadas por varios clasificadores cuyas condiciones tienen un alto porcentaje de solapamiento. Butz, Lanzi y Wilson[27] proponen agregar un mecanismo de compactación a XCSF, que logra reducir el tamaño de la población en un 90 %, con un efecto muy pequeño sobre la precisión.

Un problema para cualquier algoritmo de compactación es que la población reducida puede que no cubra todos los estados. Sin embargo, determinar los subespacios no cubiertos

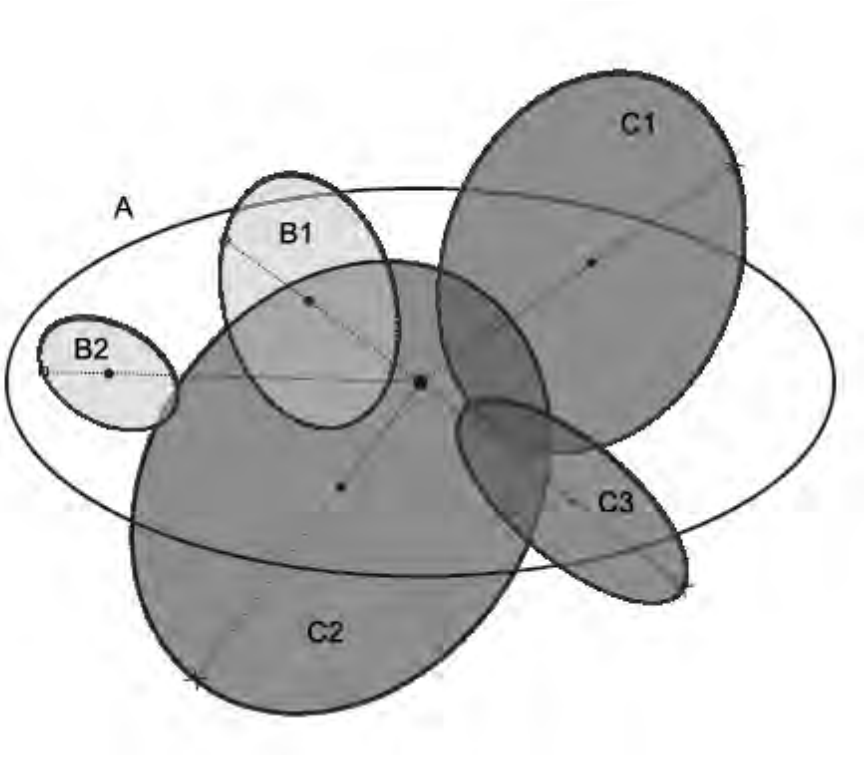

Figura 1.7: Subsunción en hiper-elipsoides

es un problema que requiere alta capacidad de cómputo. El algoritmo propuesto por Butz *et al.* evita este problema, pues al comenzar la aplicación, el sistema XCS entra en "modo de compactación", donde el algoritmo genético ya no aplica los operadores de cruce ni de mutación. Además, se utiliza la escogencia de los clasificadores más cercanos para el *matching*.

Dado un estado cualquiera, la actividad de cada clasificador se determina y luego los Θ_M clasificadores con actividad más alta se agregan al *match set*, con Θ_M un parámetro definido en el modelo. La consecuencia de esto es que la cardinalidad del *match set* es Θ_M, y consiste en un mosaico de clasificadores que se solapan de acuerdo a la distribución de las condiciones en el espacio del problema y el tamaño de los mismos.

El mecanismo de compactación escogido es un algoritmo voraz, que considera iterativamente los clasificadores suficientemente experimentados y con el error más bajo. Posteriormente se crea un conjunto que contiene todos los clasificadores que cubren al centro del clasificador seleccionando anteriormente. Todos los clasificadores en este conjunto son eliminados y su numerosidad se transfiere al clasificador inicial, pues es el de mayor experiencia y menor error en el subespacio.

Este algoritmo garantiza que los subespacios más difíciles del problema son cubiertos por más clasificadores que aquellas regiones que son fáciles de aproximar. Después de aplicarlo, se sigue ejecutando XCSF para ajustar los valores de los clasificadores sobrevivientes.

CAPÍTULO 2

SERENA

SERENA (**S**istema **E**volutivo de **RE**des **Neuronales** **A**leatorias) es una arquitectura cuyo propósito es la predicción de series de tiempo, donde el aprendizaje lo realiza una red neuronal, donde no se utiliza el algoritmo de retropropagación para el entrenamiento de la misma, sino que utiliza un enfoque basado en estrategias evolutivas. Este enfoque es inspirado en la arquitectura GRASPES planteada por Lima[7].

En este capítulo se presenta el diseño de SERENA, y las decisiones de implementación tomadas, explicando también las características que la diferencian de GRASPES.

2.1 Diseño de la Red Neuronal

De acuerdo al propósito de SERENA de predecir series de tiempo con valores continuos, se utiliza una red neuronal *feed-forward*[1]de 3 capas para modelar las relaciones entre los valores de las series. Esta red tiene forma (i, j, k), donde i indica el número de neuronas en la capa de entrada, j representa la cantidad de neuronas en la capa oculta y k denota el número de neuronas en la capa de salida.

Como se desea predecir el siguiente valor de la serie, se fija el valor de $k = 1$. Es importante destacar que basta un horizonte de predicción de un paso, pues es posible obtener los siguientes valores utilizando los resultados previos dados por la red. Se utiliza una función de activación sigmoidal en sus unidades ocultas y también en la neurona de salida, lo que limita los resultados arrojados por la red al intervalo $[0; 1]$.

[1]Red Neuronal *feed-forward*: Una red neuronal en la que las conexiones no forman ciclos, donde la señal circula en un solo sentido.

Cada uno de los pesos de la red es inicializado con un valor aleatorio escogido de acuerdo a una distribución uniforme en el intervalo $[-0,05; 0,05]$. Nótese que para una red neuronal de forma $(i, j, 1)$, la longitud del vector de pesos está dada por la fórmula:

$$n = ij + 2j + 1 \tag{2.1}$$

Esto corresponde a la suma de la cantidad de arcos entre la capa de entrada y oculta (ij), con un peso correspondiente al umbral de cada una de las neuronas en la capa oculta (j), la cantidad de arcos que llegan a la neurona de salida y su umbral $(j + 1)$. Simplificando los términos obtenemos la ecuación presentada arriba.

2.2 Codificación del Individuo

Uno de los puntos más importantes al trabajar con estrategias evolutivas es la codificación del individuo, pues esta debe permitir que puedan realizarse rápida y efectivamente todos los procedimientos necesarios para lograr la evolución. Si se escoge una estructura incorrecta, la eficiencia del sistema en cuanto a memoria y velocidad puede disminuir considerablemente.

En el caso de SERENA, el cromosoma debe representar la red neuronal planteada en la sección anterior, que es la estructura escogida para el modelado de la serie de tiempo, y debe contener también el parámetro auto-adaptativo que se define en las estrategias evolutivas.

En SERENA, se representa un cromosoma como un vector X de la forma:

$$X =< i, j, x_1, x_2, ..., x_n, \sigma > \tag{2.2}$$

donde i, j representan el número de neuronas en la capa de entrada y en la capa oculta, respectivamente; los $x_i \quad (i = 1, 2, ..., n)$ son valores reales que representan el vector de pesos de la red neuronal, con longitud n dada por la ecuación (2.1) y σ es un parámetro auto-

adaptativo que afecta el operador de mutación.

2.3 Estrategia Evolutiva

La estrategia evolutiva de SERENA, que se ejecuta sobre vectores de valores reales, es una estrategia evolutiva simple[15], por lo que no se utiliza el operador de cruce, sino que se coloca el peso de la evolución sobre la mutación.

El operador de mutación consiste en aplicar una perturbación gaussiana al cromosoma. Esto se logra obteniendo, para cada peso un valor aleatorio de una distribución normal de media 0 y desviación definida por el parámetro auto-adaptativo σ que representa el tamaño de la mutación y co-evoluciona con el resto del cromosoma.

Luego, el cromosoma mutado $X' = (i, j, x'_1, x'_2, ..., x'_n, \sigma')$ se obtiene sumando el valor obtenido de la distribución normal a cada uno de los vectores de los pesos, es decir, $x'_i = x_i + N(0, \sigma)$. Posteriormente, se realiza la mutación de σ de acuerdo a la fórmula:

$$\sigma' = \sigma \exp\left\{\frac{1}{\sqrt{n}} N(0,1)\right\} \tag{2.3}$$

Se utiliza una distribución normal para generar los valores aleatorios pues es simétrica alrededor de cero y posee la característica de que la probabilidad de obtener un número de alguna magnitud dada es una función dependiente de la desviación estándar que decrece rápidamente. Esto significa que es más probable obtener mutaciones pequeñas, lo que evita grandes saltos en el espacio de búsqueda.

Adicionalmente, es necesario colocar reglas para controlar el valor de σ, pues es este parámetro el que determina la forma en que se explora el problema.

Si se producen números aleatorios muy cercanos a cero, las mutaciones tendrán un efecto mínimo sobre los cromosomas y podrían quedarse estancados. Por este motivo, se deben evitar valores muy pequeños de σ, lo que se logra estableciendo una cota mínima ε. Luego, $\sigma' < \varepsilon \Rightarrow \sigma' = \varepsilon$. Esta es una mejora planteada en SERENA que no se encuentra presente en GRASPES.

Además, el tamaño de la mutación varía de acuerdo a la regla de 1/5 de Rechenberg[28]. Esta regla establece que la proporción entre las mutaciones exitosas y el total de mutaciones debe ser de 1/5, donde se dice que una mutación es exitosa si la utilidad del hijo es mayor que la utilidad del padre. Por lo tanto, si el radio es mayor a 1/5, el tamaño de la mutación debe ser incrementado para hacer una exploración más amplia del espacio; pero si el radio es menor a 1/5, entonces debe reducirse, para concentrar la búsqueda alrededor de la solución actual.

La regla de 1/5 se ejecuta en intervalos periódicos, es decir, cada k iteraciones, el valor de σ es establecido en:

$$\sigma = \begin{cases} \sigma/c & \text{si } p_s > 1/5, \\ \sigma \cdot c & \text{si } p_s < 1/5, \\ \sigma & \text{si } p_s = 1/5, \end{cases} \tag{2.4}$$

Donde p_s es la frecuencia relativa de las mutaciones exitosas, medida en k iteraciones, y c es un parámetro que debe estar en el intervalo $0,817 \leq c \leq 1$ [28].

Para la escogencia del mejor individuo, se supone que es aquel que tiene el menor error, lo que se refleja en la medida de utilidad, que es inversamente proporcional al mismo:

$$f = \frac{1}{1 + MSE} \tag{2.5}$$

Donde MSE[2]es error el cuadrático medio, que se calcula de acuerdo a la ecuación:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (t_i - o_i)^2 \tag{2.6}$$

siendo n el número de puntos en el conjunto, t_i es el valor real y o_i es el valor arrojado

[2]MSE: *Mean Squared Error*. Error Cuadrático Medio, una medida del cuadrado del error de un estimador, que permite determinar la calidad del mismo

por la red neuronal.

2.4 Funcionamiento de SERENA

Con el propósito de crear una red neuronal que logre buenas predicciones, SERENA genera un individuo inicial y posteriormente procede a aplicar la estrategia evolutiva sobre el mismo. Se aplica el operador de mutación sobre el individuo padre un total de *num_mutaciones* veces para generar los hijos.

Los hijos son evaluados uno a uno de acuerdo a su medida de utilidad sobre el conjunto de entrenamiento y se incluyen en un conjunto M de los mejores cromosomas obtenidos, si y sólo si su utilidad, f, es mejor que la del padre actual, bajo la hipótesis que los cromosomas con alta utilidad producirán hijos aun mejores al ser mutados.

Se dice que un individuo A es mejor que un individuo B si y sólo si la utilidad del primero es mayor que la utilidad del segundo, sumado a una constante u, definida en el modelo, es decir:

$$es\_mejor(A, B) \Leftrightarrow f(A) > f(B) + u \tag{2.7}$$

El parámetro u se utiliza por dos razones: en primer lugar, para contrarrestar la pérdida de precisión en números flotantes al momento de implementar; y en segundo lugar, permite determinar qué tan buena debe ser la mejora.

Una vez se han realizado todas las mutaciones, se escoge el cromosoma en el conjunto M que tenga la mayor utilidad. Si es mejor que el padre, entonces es seleccionado para sustituirlo en la siguiente iteración, de esta forma, SERENA evoluciona siempre los individuos con mayor utilidad de cada iteración.

Si los hijos obtenidos no mejoran las predicciones, entonces se mantiene el mismo padre para la siguiente iteración, lo que podría ocasionar que el algoritmo se quede estancado y termine rápidamente. Sin embargo, en este caso, la proporción de mutaciones exitosas bajará rápidamente, lo que garantiza de acuerdo a la regla de 1/5 que se aumentará el tamaño de la mutación σ para permitir la exploración de nuevas soluciones.

En cualquier caso, se vacía el conjunto M y se repite el proceso. Este vaciado es un cambio de SERENA sobre GRASPES motivado con la idea de generar más individuos y realizar una exploración más amplia del espacio de búsqueda.

La estrategia evolutiva se repite hasta que se cumpla la condición de parada, que en el caso de SERENA consiste en que pasen g generaciones sin que el sistema logre conseguir un mejor individuo, con este parámetro fijado en los experimentos.

Explícitamente, el algoritmo de SERENA consta de los siguientes pasos:

1. Se crea el primer cromosoma *inicial* y se calcula su utilidad.
2. Se establece el padre $P = inicial$
3. Se inicializa un contador $generaciones\_sin\_mejora = 0$
4. Mientras no se cumpla la condición de parada $generaciones\_sin\_mejora > g$ se repite el proceso:
   a) Mientras no se hayan realizado $num\_mutaciones$
      1) Se crea un duplicado del cromosoma padre llamado *clon*
      2) Se aplica el operador de mutación sobre el vector de pesos
      3) Se aplica el operador de mutación sobre σ
      4) Se calcula la utilidad del cromosoma mutado *clon*
      5) Se determina *clon* es mejor que P, de acuerdo a la fórmula (2.7). Si lo es, se agrega *clon* al conjunto M de los mejores clasificadores.

   b) Se escoge $mejor$, el cromosoma con la utilidad más alta del conjunto M. Si M es vacío, entonces se toma $mejor = P$

   c) Si $es\_mejor(mejor, P)$ entonces se sustituye el padre $P = mejor$. En caso contrario, se aumenta el contador $generaciones\_sin\_mejora$

   d) Se vacía el conjunto M

   e) Se aplica la regla de 1/5 de Rechenberg
5. Se devuelve como resultado el individuo P

CAPÍTULO 3

CLARE

CLARE (**CLA**sificadores + **RE**des) es una arquitectura que consiste en un sistema de clasificadores genéticos XCSF modificado de tal forma que la predicción de la recompensa se realice utilizando redes neuronales. Su propósito es aprender las características y relaciones entre los puntos de una serie de tiempo para hacer predicciones sobre la misma.

En este capítulo se describe CLARE y las modificaciones realizadas para adaptar XCSF al objetivo de esta investigación. Adicionalmente, se especifican detalles de implementación del sistema.

3.1 Origen de CLARE

En el campo de la computación, se han planteado diversas perspectivas para enfrentarse al complejo problema de la predicción de series de tiempo. Muchos de los enfoques exitosos utilizan modelos conexionistas, o de redes neuronales, para modelar la relación entre los valores pasados y futuros de una serie de tiempo.

Sin embargo, en series complejas, existe la posibilidad de que en distintos subespacios del problema los valores de la serie tengan relaciones y características diferentes que una red neuronal no pueda modelar de forma eficiente. Bajo esta idea, se plantea utilizar redes neuronales diferentes para cada uno de estos subespacios, con la hipótesis de que cada una de estas redes podrá realizar un mejor aprendizaje sobre el mismo.

La dificultad de esta aproximación viene al momento de determinar la forma más adecuada de particionar el espacio del problema para entrenar cada una de las redes neuronales. El sistema de clasificadores XCSF provee esta capacidad, utilizando las condiciones de los mis-

mos para determinar cuáles son las reglas a aplicar en cierto estado. Además, XCSF permite realizar aproximaciones de funciones, lo que es necesario para la arquitectura.

Luego, de acuerdo a trabajos recientes realizados sobre XCSF donde se sustituye la función de predicción lineal por otras más complejas [23] [24], se decide utilizar el enfoque planteado por Lanzi y Loiacono[9] de utilizar redes neuronales para realizar la predicción de la recompensa de cada clasificador.

3.2 Diseño de CLARE

En CLARE, se mantiene la misma estructura de los clasificadores que en XCSF, pero con dos cambios importantes. En primer lugar, se decide sustituir las condiciones hiper-rectangulares por condiciones hiper-elisoipdales, pues las últimas han probado ser capaces de obtener particiones iguales o mejores que las primeras [25].

En segundo lugar se cambia la función de predicción, sustituyendo la predicción lineal por una red neuronal *feed-forward* de 3 capas para realizar el aprendizaje. Estas redes neuronales tienen la misma estructura $(i, j, 1)$ que las utilizadas por SERENA (ver Sección 2.1), sin embargo, las neuronas de la capa de salida tienen una función de activación lineal.

Luego, cada uno de los clasificadores posee una red neuronal utilizada para generar la predicción del mismo. Esta red es codificada utilizando un vector de reales $w =< w_1, w_2, ..., w_n >$ que representa los pesos sinápticos de la misma. De esta forma, al momento de calcular la predicción de un clasificador, los valores obtenidos del estado del ambiente son procesados por una red neuronal construída con los pesos determinados por el vector w.

El tamaño del vector de pesos está dado por la fórmula (2.1), donde la cantidad de neuronas en la capa de entrada y en la capa oculta son fijados al crear el modelo de acuerdo a la serie a predecir.

En CLARE, la población inicial es vacía, y los clasificadores van siendo creados a medida que son necesarios por el mecanismo de cobertura. Al momento de crear un clasificador para cubrir un estado, se inicializa con una condición centrada en el estado y una red neuronal con un vector de pesos aleatorios con valores escogidos con distribución uniforme en el intervalo $[-0,05; 0,05]$.

Manteniendo las características de XCSF, los clasificadores poseen solamente una acción ficticia, lo que tiene como consecuencia que la predicción del sistema sea un promedio pesado de la predicción de cada clasificador en el *match set*, de acuerdo a su utilidad.

Al recibir la recompensa del ambiente, que en este caso corresponde al valor real de la serie de tiempo, es necesario actualizar la función de predicción de acuerdo al error. En el caso de CLARE, se debe actualizar la red neuronal, para lo que se utiliza el algoritmo de retropropagación (ver Sección 2.1).

El parámetro η, utilizado en XCSF para actualizar la predicción, es utilizado en esta arquitectura como la tasa de aprendizaje del algoritmo de retropropagación. Es importante mencionar que en el caso de la predicción de series de tiempo, como se cuenta con datos limitados, estos valores se alimentan varias veces al sistema, por lo que es preferible utilizar una tasa de aprendizaje bastante baja.

CLARE, en su ejecución, mientras recibe los estados del ambiente, va alternando entre dos modos: exploración y explotación.

En el modo de exploración, se sigue el proceso normal de XCS: obtener el *match set*, realizar la predicción, actualizar el clasificador de acuerdo a la recompensa y posteriormente ejecutar el algoritmo genético sobre el *action set*, que en este caso es equivalente al *match set*. En el modo de explotación, el algoritmo genético no es utilizado, por lo que no se realiza el descubrimiento de nuevas reglas.

Esta alternancia entre exploración y explotación permite que el algoritmo genético busque en el espacio del problema nuevas reglas que podrían permitir mejores predicciones, pero con un refuerzo adicional a los clasificadores que han mostrado una alta precisión.

En cuanto al algoritmo genético, los padres se seleccionan de acuerdo al método del torneo. Se selecciona aleatoriamente una cantidad establecida de clasificadores de la población y se escoge de ellos al que tenga mayor utilidad. Los operadores de cruce y mutación aplican sobre las condiciones, pero no sobre las redes neuronales.

El mecanismo de subsunción de clasificadores es utilizado, pero sólo durante el algoritmo genético. No es utilizado en el *action set*. Esta decisión es tomada pues sólo existe una acción

ficticia, por lo que aplicar este algoritmo al *action set* es equivalente a aplicarla en el *match set*, lo que no es conveniente pues puede reducir mucho la cantidad de reglas en la población.

Como condición de parada de CLARE se fija un número máximo de iteraciones.

La Figura 3.1 muestra un esquema del funcionamiento de CLARE, donde el estado es dos dimensiones, lo que indica que la ventana de predicción es de tamaño 2. Esto se ve reflejado también en las redes neuronales, que tienen 2 neuronas en la capa de entrada.

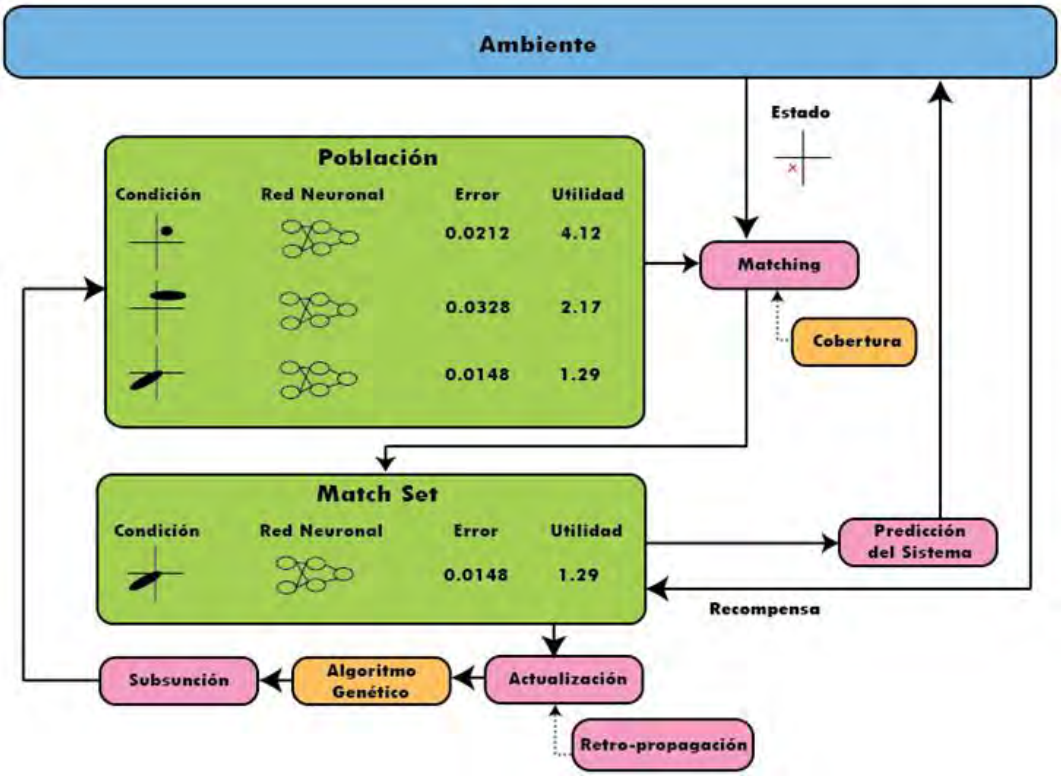

Figura 3.1: Funcionamiento de CLARE

3.3 Detalles de Implementación

Para la implementación de CLARE fue utilizada la librería *XCSFJava 1.1*, escrita por Butz [29]. Este código implementa todas las funciones del sistema de clasificadores XCSF, incluyendo la extensión de las condiciones a números reales, permitiendo también utilizar condiciones hiper-elisoipdales y contiene el algoritmo voraz de compactación, dos características usadas por CLARE.

XCSFJava 1.1 utiliza Java3D 1.5 para permitir la visualización del proceso de aprendizaje,

mostrando en el espacio del problema los clasificadores que son creados y que se encuentran activos, para funciones de dos y tres dimensiones.

En la Figura 3.2 se observan las condiciones hiper-elisoipdales creadas por el CLARE en una iteración temprana del aprendizaje. En azul claro se muestran las condiciones activas con un estado dado y las figuras se vuelven más oscuras a medida que su precisión sube.

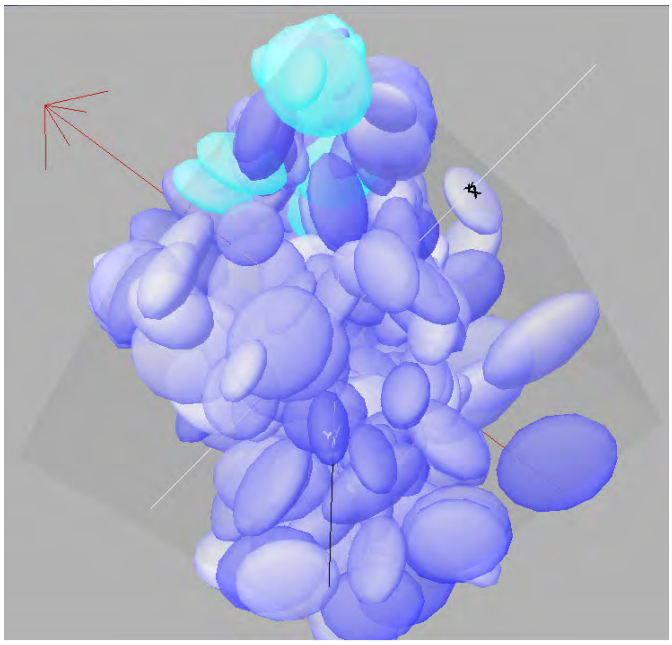

Figura 3.2: Población de condiciones hiper-elisoipdales en un espacio tridimensional

Fue necesario realizar varias modificaciones sobre el código de *XCSFJava 1.1* para adaptarlo a la predicción de series de tiempo.

Fue necesario crear una clase `TimeSeriesFuction` que representa al ambiente, y se encarga de otorgarle al sistema el estado en que se encuentra, un conjunto de valores sucesivos de la serie de tiempo, y posteriormente la recompensa, el próximo valor de la serie.

Adicionalmente, se creó una clase `NeuralNetworkPrediction` que sustituye la función de predicción lineal utilizada por defecto en *XCSFJava 1.1*. Dentro de esta clase se encuentra el vector de pesos que codifica a la red neuronal. Esta clase se encarga de tomar los valores del estado y procesarlos en la red correspondiente, para dar al clasificador la predicción de la recompensa esperada. `NeuralNetworkPrediction` contiene también el método de actualización de la predicción, que en este caso consiste en utilizar el algoritmo de retropropagación sobre la red neuronal.

Además se realizaron modificaciones sobre las clases que representan un clasificador para permitir el uso de este tipo de predicción neuronal.

Finalmente, las clases que se encargan de ejecutar el algoritmo de XCSF fueron también alteradas para utilizar los clasificadores con redes neuronales y para ciclar sobre el conjunto de entrenamiento dado. Se incluyó además un método para obtener el MSE y la numerosidad de los clasificadores al finalizar una iteración sobre el conjunto de entrenamiento completo.

La estructura del código se puede observar en la Figura 3.3, según es descrita por el mismo Butz[30].

`XCSF` es la clase administradora que se encarga de ejecutar el algoritmo, obtener el estado y coordinar la visualización, delegando a `XCSSets` el mantener los conjuntos *action set* y *match set*. Estos conjuntos pertenecen a la clase `ClassifierSet`, que implementa la actualización de los clasificadores, algoritmo genético y subsunción. Los clasificadores son de la clase `Classifier` y pueden ser booleanos o reales. Estos contienen las condiciones y la función de predicción.

Se muestran en letras negras las clases nuevas y dentro de rectángulos las clases modificadas.

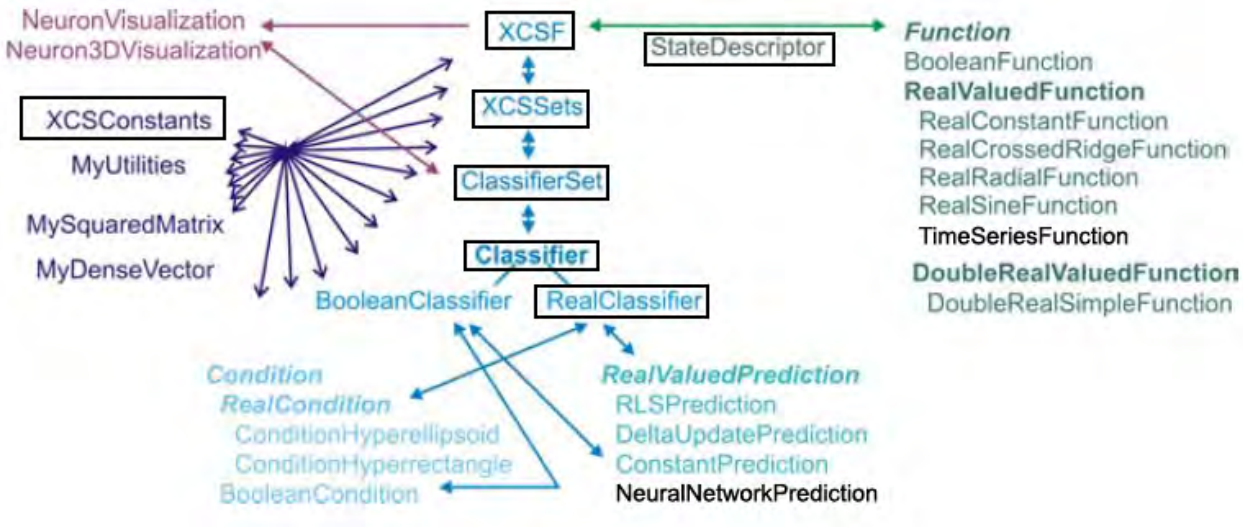

Figura 3.3: Estructura del código de *XCSFJava 1.1* extendido por CLARE. Tomado de [30]

CAPÍTULO 4

EXPERIMENTOS Y RESULTADOS

En este capítulo se presentarán los diversos experimentos realizados y los resultados obtenidos en los mismos, además de las observaciones que se pueden derivar respecto a las series y arquitecturas utilizadas.

Recordemos que el principal objetivo de esta investigación consiste en probar el funcionamiento de SERENA y CLARE, realizando además una comparación entre ambas arquitecturas y una red neuronal entrenada con retropropagación, para determinar si alguna muestra un mejor comportamiento al momento de realizar la predicción sobre un par de series de tiempo previamente escogidas, y si ese es el caso, en cuánto difieren.

Para escoger las condiciones más convenientes para el desempeño de las arquitecturas, se hicieron pruebas empíricas como preparación para realizar los experimentos. Todos los parámetros de importancia fueron fijados después de realizar un proceso experimental para escoger un conjunto adecuado, que diera resultados buenos y estables. Los parámetros pueden consultarse en el Apéndice A.

En particular, dado que ambas utilizan redes neuronales, fue necesario realizar un estudio sobre las mismas para determinar la estructura adecuada para ellas. Detalles de estos experimentos pueden encontrarse en el Apéndice C.

También se realizó un experimento sobre la CLARE para determinar el efecto que tiene el algoritmo de compactación sobre la misma, en cuanto a la numerosidad de los clasificadores y el error de predicción.

Para todos los experimentos se utilizaron dos series de tiempo diferentes, de las cuales se toman los primeros puntos, un total de 70 %, como conjunto de entrenamiento y los siguientes, el 30 % restante, como conjunto de prueba.

Todas las corridas se realizaron en computadores con procesador *Intel(R) Core(TM)2 Duo* 2.0Ghz con 2MB de cache y 2GB de memoria RAM.

El desempeño de ambas arquitecturas se evaluó utilizando el MSE[1]sobre el conjunto de prueba. En base a los resultados del mismo podemos determinar la diferencia de los resultados del sistema contra los valores reales de la serie, lo que a su vez es indicativo de qué tan precisa es la arquitectura.

4.1 Escogencia de las Series de Tiempo

Para realizar los experimentos sobre SERENA y CLARE se decidió escoger dos series de tiempo de orígenes diferentes, una de las proveniente de algún fenómeno natural y otra resultado de la interacción entre seres humanos.

Como la arquitectura SERENA utiliza redes neuronales con una neurona de salida sigmoidal para realizar sus predicciones, se tiene la restricción de que todos los puntos de la serie deben estar en el intervalo $[0; 1]$, sin embargo, esta restricción puede ser superada mediante la normalización de los datos. CLARE no tiene esta limitante, pero se usan las series normalizadas para comparar contra SERENA.

En el caso de que se desee aprender una función cualquiera, basta con aplicar la normalización a las salidas de la misma en el conjunto de entrenamiento, pero en el caso particular de las series de tiempo, al ser dependientes de los valores anteriores, es conveniente realizar la normalización de la serie completa.

Las series utilizadas en la evaluación de las arquitecturas fueron normalizadas para encontrarse en el intervalo $[0; 1]$, aplicando a cada punto la fórmula:

[1]MSE: *Mean Squared Error*. Error Cuadrático Medio, una medida del cuadrado del error de un estimador, que permite determinar la calidad del mismo

$$pnt = \frac{v - vmin}{vmax - vmin} \tag{4.1}$$

donde pnt es el valor normalizado, v es el valor original del punto y $vmin, vmax$ son el valor mínimo y el valor máximo de la serie original, respectivamente. Esta normalización afecta la distancia entre los puntos, pero no afecta la proporción entre los mismos, lo que se traduce en que las arquitecturas no darán el valor exacto de una serie el día siguiente, sino un valor proporcional en el intervalo $[0; 1]$, pero si se desea obtener la predicción en el rango original, la función es fácilmente reversible.

4.1.1 Precio de Cierre del DJIA

Se escogió como primera serie al conjunto de los precios de cierre diarios del DJIA[2]en el período comprendido entre el 1 de enero de 1998 y el 26 de agosto de 2003, obteniendo una serie de 1420 puntos. Estos datos se obtuvieron del website de Yahoo! Finanzas [10]. Influyó en la escogencia de este período el haber sido utilizada previamente por Lima [7].

Esta serie de tiempo normalizada, a la que llamaremos "Serie DJIA", fue escogida por tener su origen en el mercado bursátil.

4.1.2 Temperatura más alta en la estación climática de Emerald, Australia

En la segunda serie de tiempo escogida, recolectada por el Departamento de Meteorología del Gobierno de Australia [11], cada punto corresponde a la temperatura máxima diaria tomada por una estación meteorológica ubicada en Emerald, Australia. Esta serie fue normalizada y nos referiremos de aquí en adelante a ella como "Serie Emerald".

Esta serie fue escogida porque está relacionada con un fenómeno natural que sirve como indicador de los cambios ambientales en una región, además que permite evaluar el compor-

[2]DJIA: *Dow Jones Industrial Average*. Promedio Industrial Dow Jones, es un índice del mercado bursátil

tamiento de las arquitecturas con una serie periódica.

4.2 Experimento #1: Desempeño de SERENA

En este experimento se procede a tomar la estructura de red neuronal que produjo mejores resultados en los experimentos de entonación (ver Apéndice C) y se realizan 10 ejecuciones de SERENA sobre la misma, para determinar si la configuración escogida arroja buenas predicciones constantemente.

4.2.1 Resultados sobre Serie DJIA

Para este experimento fue seleccionada la estructura $(1, 5, 1)$ para la red neuronal de SERENA, y se realizaron 10 corridas de la misma, para posteriormente obtener el MSE de cada una de ellas sobre el conjunto de prueba, de lo que se obtuvo que el MSE promedio es $3,62\times10^{-3}$, con una varianza de $5,4\times10^{-5}$. El MSE mínimo obtenido fue de $9,76\times10^{-4}$ y el máximo fue de $2,56\times10^{-2}$.

Sin embargo, se debe hacer notar que el MSE más alto es un valor atípico y que el resto de los errores son muy similares y de un valor mucho más bajo, lo que indica que la estructura $(1, 5, 1)$ produce generalmente muy buenos resultados.

En la Figura 4.1 y la Figura 4.2 se comparan la mejor y peor predicción de SERENA sobre el conjunto de entrenamiento y el conjunto de prueba de la Serie DJIA, respectivamente. Se puede apreciar que en ambas gráficas, la serie real y la mejor predicción son muy cercanas.

Al comparar la serie con la peor predicción obtenida, se puede apreciar que aunque los valores se encuentran bastante alejados, las formas son parecidas, lo que indica que aun la peor predicción puede dar una idea de las alzas y bajas del mercado.

Al graficar los errores de la predicción en cada uno de los puntos del conjunto de pruebas se obtiene la Figura 4.3, donde se observa que la distribución de los errores en la peor predicción presenta cierto patrón, lo que indica que el sistema no logró captar alguna característica de la serie. Aun mas, el patrón mostrado es similar al de la Figura C.3, lo que es indicativo de que es una cualidad particular difícil de aprender.

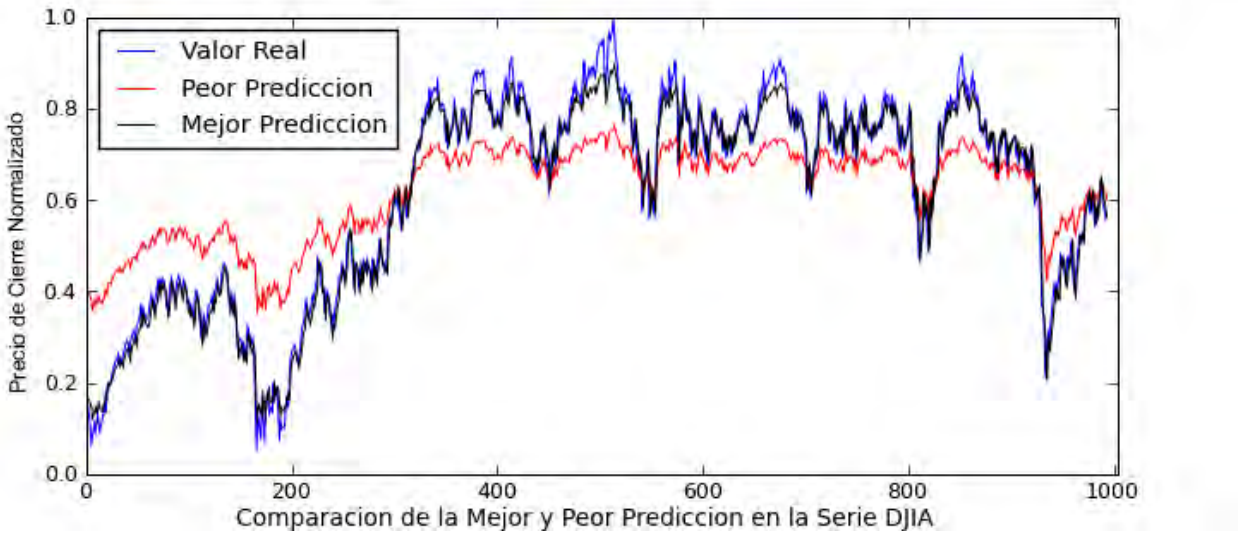

Figura 4.1: Conjunto de entrenamiento de la Serie DJIA, con la mejor y peor predicción dadas por SERENA con red neuronal (1,5,1)

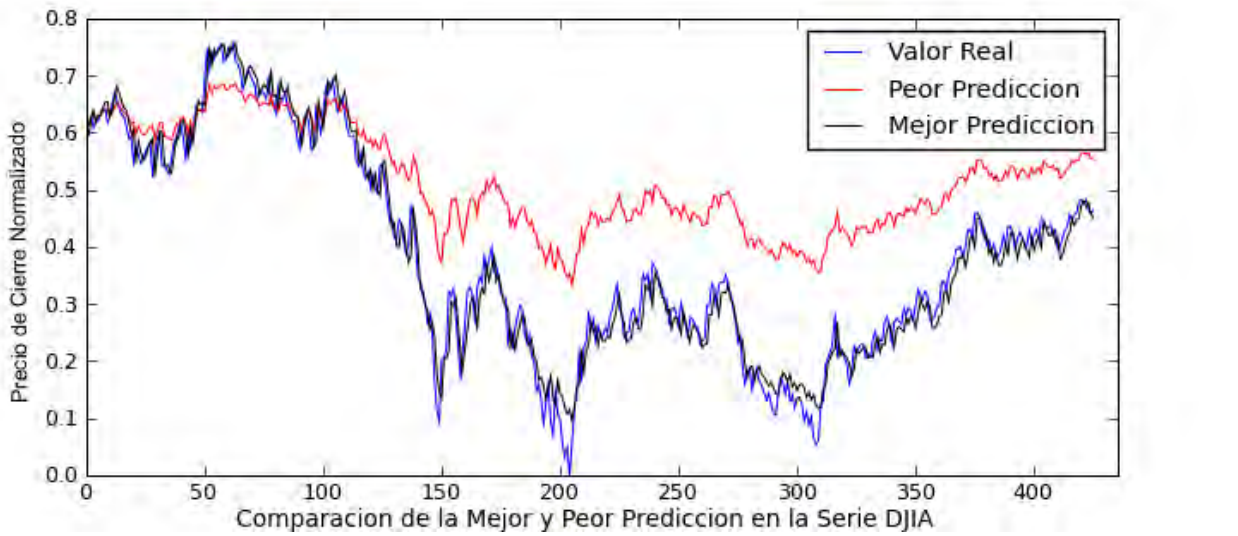

Figura 4.2: Conjunto de prueba de la Serie DJIA, con la mejor y peor predicción dadas por SERENA con red neuronal (1,5,1)

En cambio, la distribución de los errores en la mejor predicción no presenta estructura evidente y los errores son pequeños, lo que permite inferir que la red ha logrado aprender un buen modelo de la red.

Otra evaluación del desempeño del sistema consiste en graficar los valores reales contra los valores predichos por el mismo. En el caso de una predicción perfecta, estos valores serían iguales, por lo que se ajustarían a la recta $f(x) = x$.

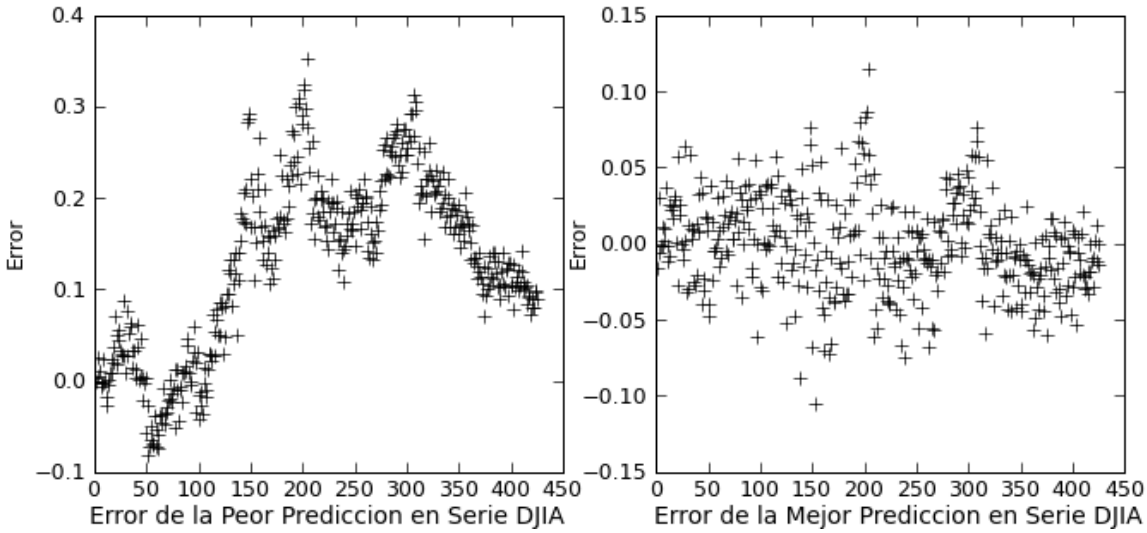

Figura 4.3: Gráfico de error en cada punto de la peor y mejor predicción que obtuvo SERENA con red neuronal (1,5,1) en el conjunto de prueba de la Serie DJIA.

En la Figura 4.4 se presenta la recta ideal, y se puede observar que la peor predicción da resultados acertados en el intervalo $(0,6; 0,7)$, pero se va alejando de los valores reales a medida que estos se hacen más pequeños; mientras que en la gráfica de la red neuronal que obtiene la mejor predicción se ve que los puntos son bastante cercanos a la recta.

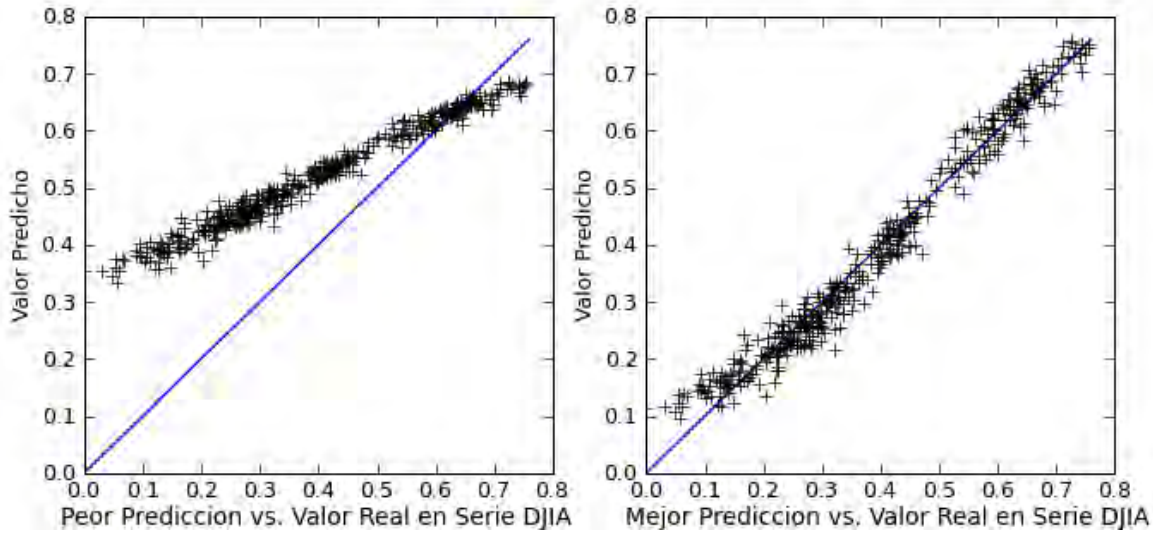

Figura 4.4: Gráfico de valor real contra valor predicho por la peor y mejor predicción que obtuvo SERENA con red neuronal (1,5,1) en el conjunto de prueba de la Serie DJIA.

4.2.2 Resultados sobre Serie Emerald

Se realizaron 10 ejecuciones de SERENA con red neuronal de configuración $(2, 2, 1)$ para ser evolucionada y obtener las predicciones y el MSE sobre el conjunto de prueba de la serie Emerald, obteniendo un MSE promedio de $6,94\times10^{-3}$, con varianza de $2,17\times10^{-4}$. Los valores mínimo y máximo del MSE son, respectivamente $1,53\times10^{-3}$ y $5,11\times10^{-2}$.

Todos los valores del MSE obtenidos, excepto uno, son bajos y parecidos entre sí. El valor del error atípico, nuevamente representa el MSE más alto que se obtuvo en el experimento, lo que significa que SERENA con una red neuronal $(2, 2, 1)$, obtiene resultados consistentes para la Serie Emerald.

La comparación entre la mejor y peor predicción de SERENA sobre el conjunto de entrenamiento y el conjunto de prueba de la Serie Emerald se puede apreciar en la Figura 4.5 y la Figura 4.6, respectivamente.

En las gráficas se aprecia que tanto la mejor como la peor predicción logran captar la periodicidad de la serie. Sin embargo, la peor predicción no tiene la amplitud apropiada. Se observa también que el error de la predicción aumenta en los valores extremos.

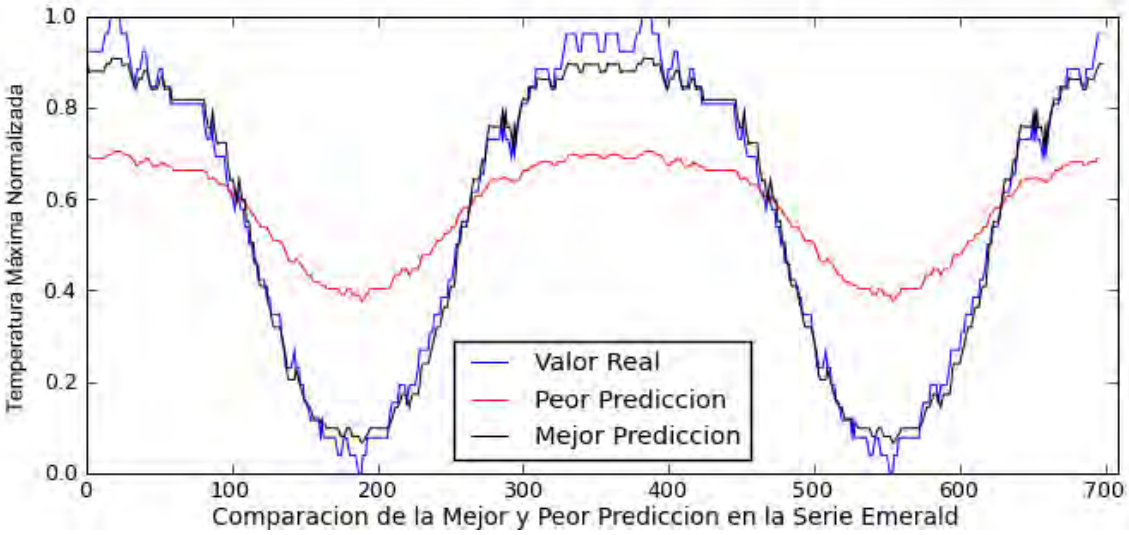

Figura 4.5: Conjunto de entrenamiento de la Serie Emerald, con la mejor y peor predicción dadas por SERENA con red neuronal (2,2,1)

La distribución de los errores en la Figura 4.7 muestra que tanto en el gráfico de la peor

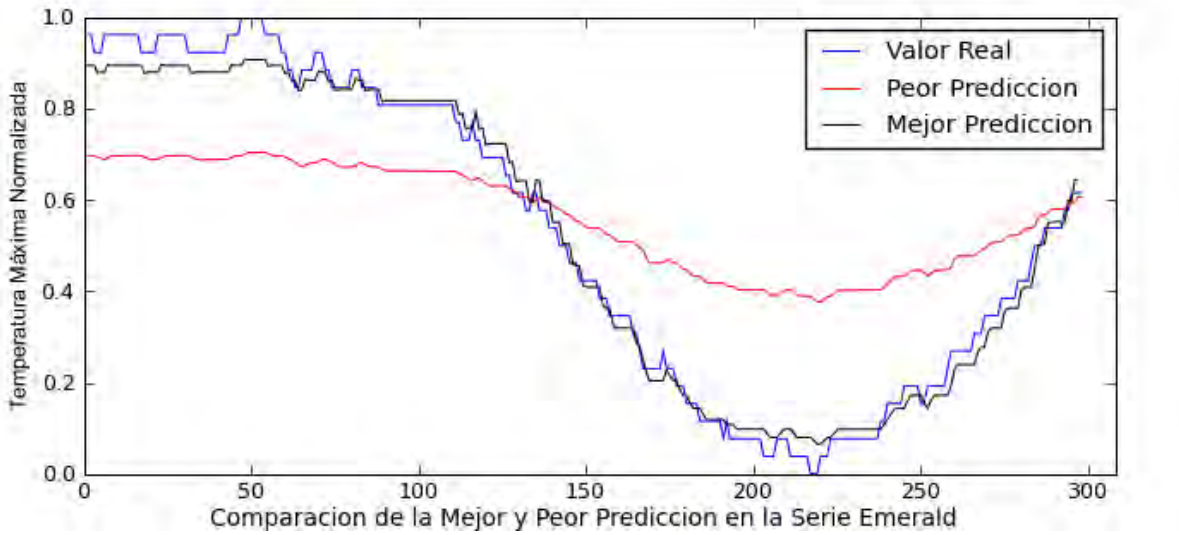

Figura 4.6: Conjunto de prueba de la Serie DJIA, con la mejor y peor predicción dadas por SERENA con red neuronal (2,2,1)

predicción como en la de mejor predicción se distinguen claramente ciertos patrones formados por los errores, pero diferentes entre sí.

En la gráfica se observa que SERENA no logró captar todas las cualidades de la serie. En la peor predicción pierde una característica muy importante, lo que se refleja en un MSE alto; mientras que la mejor predicción tiene MSE bajo, lo que muestra que la característica que no se logró aprender es de menor importancia para el modelo de la serie.

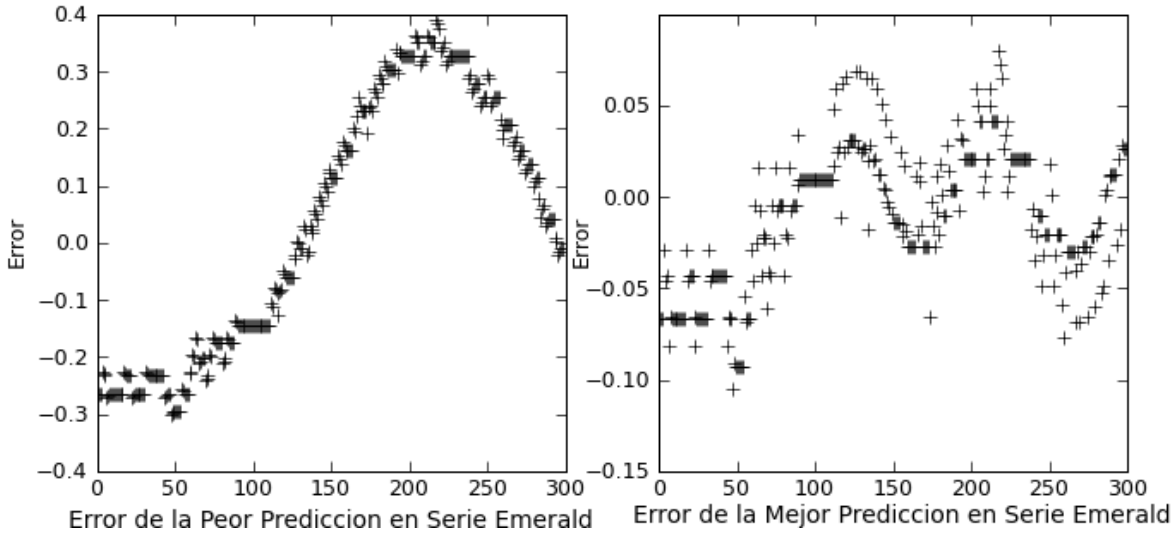

Figura 4.7: Gráfico de error en cada punto de la peor y mejor predicción que obtuvo SERENA en el conjunto de prueba de la Serie Emerald

El gráfico de los valores reales contra los valores predichos en el conjunto de prueba de la Serie Emerald, presentado en la Figura 4.8 , puede dar más datos sobre el comportamiento del sistema, al determinar la relación que tienen con la recta $f(x) = x$, que corresponde a una predicción perfecta.

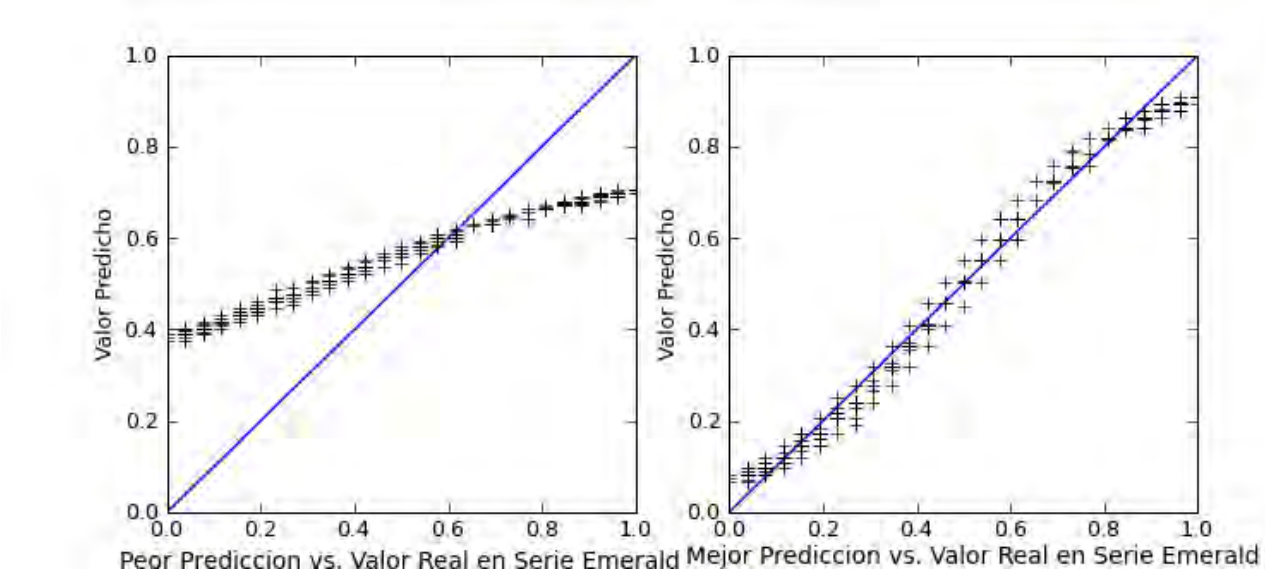

Figura 4.8: Gráfico de valor real contra valor predicho por la peor y mejor predicción que obtuvo SERENA en el conjunto de prueba de la Serie Emerald.

Se observa que en la mejor predicción, los puntos obtenidos forman también un patrón simétrico sobre la línea azul, lo que indica que aunque los resultados son buenos, pues son cercanos a la línea, el sistema no ha logrado hacer un buen ajuste, ya que se puede apreciar los valores reales para los cuales subestima y sobrestima las predicciones.

4.2.3 Observaciones Generales sobre los Resultados de SERENA

Para ambas series de tiempo, se fijó una estructura de red neuronal a ser evolucionada por SERENA, obteniendo en cada ejecución un vector de pesos diferente, lo que se refleja en diferentes predicciones con distintos valores de MSE. Esto es evidencia que no sólo es importante la estructura de la red neuronal al momento de realizar una predicción, ya que la misma configuración puede dar predicciones muy diferentes.

Por lo expuesto anteriormente, queda claro que una sola ejecución de SERENA para una estructura de red no da una idea real del comportamiento de la misma, pues tanto en la Serie DJIA como en la Serie Emerald se observan valores de MSE muy alto que podrían llevar a

pensar que la configuración no es adecuada para la serie, cuando éstos son realmente valores atípicos; por esto es conveniente deben realizar varias corridas usando la misma configuración y utilizar el valor promedio del error y observar la varianza del mismo, esperando valores bajos para ambos.

4.3 Experimento #2: Compactación de Clasificadores en CLARE

El algoritmo de compactación tiene como objetivo reducir la cantidad de clasificadores en la población, tratando de mantener aquellos que permitan obtener las mejores predicciones y evitando que aumente el error total. Este proceso es aplicado para reducir la redundancia y hacer que dado un estado, éste se corresponda con menor cantidad de clasificadores, con lo que se disminuye el tiempo de evaluación, lo que permite obtener más rápido el resultado.

En este experimento se revisan los resultados dados por CLARE con la intención de determinar si es necesario o conveniente utilizar el algoritmo de compactación en las series de tiempo evaluadas. Con este propósito se toman tres configuraciones de redes neuronales: una pequeña, $(1, 2, 1)$, una de tamaño medio, $(5, 5, 1)$ y una grande, $(8, 10, 1)$; evaluando las predicciones de la arquitectura tanto sobre la Serie DJIA como sobre la Serie Emerald, antes y después de aplicar la compactación.

Se aplican 5 ejecuciones de CLARE para cada uno de las redes neuronales seleccionadas, haciendo un total de 15 experimentos por serie. En cada corrida se coloca un máximo de 100000 iteraciones, comenzando la compactación a partir de la iteración 90000. Se utiliza un tamaño máximo de población de 1000 clasificadores, cantidad determinada experimentalmente mediante pruebas sucesivas.

4.3.1 Compactación sobre Serie DJIA

En el Cuadro 4.1 se puede observar que el MSE promedio de la red $(1, 2, 1)$ es mucho más alto que el de las otras estructuras de red, por lo que fue graficado en un cuadro aparte, de escala diferente, para permitir observar mejor las diferencias de la estructura $(5, 5, 1)$ y $(8, 10, 1)$. El error más bajo, por otro lado, es el de la red $(5, 5, 1)$.

Se aprecia que para las tres configuraciones evaluadas, el MSE que se obtiene antes de aplicar el algoritmo de compactación es menor que el obtenido después de compactar. Además, en todos los casos mostrados, la varianza aumenta al compactar, lo que hace a la

arquitectura menos precisa.

Cuadro 4.1: MSE Promedio y Desviación Estándar dados por CLARE sobre el conjunto de prueba de la Serie DJIA, antes y después de la compactación

Red	Promedio A/C	Desviación A/C	Promedio D/C	Desviación D/C
$(1, 2, 1)$	$3,51$	$1,12\times10^{-1}$	$3,53$	$2,09\times10^{-1}$
$(5, 5, 1)$	$2,68\times10^{-3}$	$8,36\times10^{-4}$	$3,16\times10^{-3}$	$8,39\times10^{-4}$
$(8, 10, 1)$	$4,98\times10^{-3}$	$1,01\times10^{-4}$	$5,54\times10^{-3}$	$3,12\times10^{-4}$

Esto es indicativo de que CLARE obtiene mejores predicciones antes de aplicar la compactación, sin embargo, es necesario revisar también las mejoras que hace el algoritmo sobre la cantidad de clasificadores en la población, lo que se muestra en la Figura 4.9, donde se observa que mientras más grande es la red, mayor es el número de macroclasificadores generados.

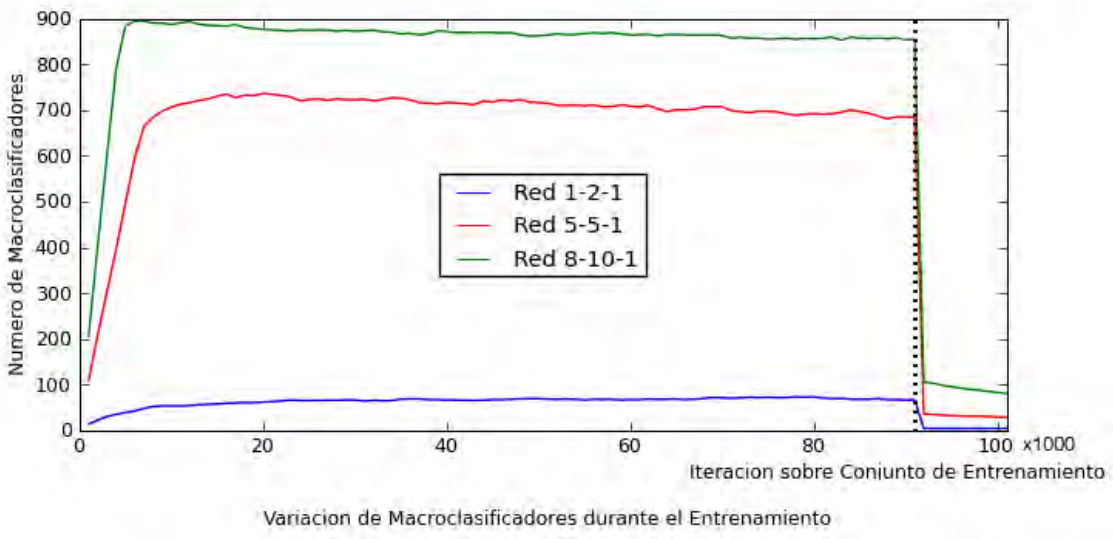

Figura 4.9: Número de Macroclasificadores al terminar cada iteración sobre el conjunto de entrenamiento de la Serie DJIA, representando el comienzo del algoritmo de compactación con una línea vertical punteada.

También se nota que la cantidad de macroclasificadores sube rápidamente en las primeras iteraciones, mientras cubre los posibles estado, y luego se mantiene más o menos estable hasta la compactación, cuando disminuye drásticamente, en particular en las redes $(5, 5, 1)$ y $(8, 10, 1)$.

Esto es evidencia de que el algoritmo de compactación logra reducir el tamaño de la

población de clasificadores, pero al revisar el tamaño promedio del *match set*[3]antes de compactar,se obtienen los valores del Cuadro 4.2.

Cuadro 4.2: Tamaño promedio del *match set* antes de compactar en la Serie DJIA.

Red	Tamaño,
$(1, 2, 1)$	31,21
$(5, 5, 1)$	1,72
$(8, 10, 1)$	2,12

En las últimas dos redes son valores bajos tomando en cuenta la cantidad de macroclasificadores que hay en total, lo que es signo de condiciones son específicas sobre estados de la serie. Al utilizar el algoritmo de compactación, se reducen el número de clasificadores en total, y por lo tanto del *match set*, por lo que las reglas se vuelven entonces más generales, lo que de acuerdo a la serie, puede aumentar el MSE, como en este caso.

4.3.2 Resultados sobre Serie Emerald

CLARE presenta resultados similares sobre la Serie Emerald que sobre la Serie DJIA, presentada en la sección anterior, como se puede observar en el Cuadro 4.3, donde el MSE promedio de la red $(1, 2, 1)$ y su varianza destacan por ser mucho más altos que los de las otras redes, lo que indica que esta estructura es poco apropiada para el aprendizaje de la serie.

Cuadro 4.3: MSE Promedio y Desviación Estándar dados por CLARE sobre el conjunto de prueba de la Serie Emerald, antes y después de la compactación

Red	Promedio A/C	Desviación A/C	Promedio D/C	Desviación D/C
$(1, 2, 1)$	$7662,42$	$10766,56$	$7816,40$	$12813,28$
$(5, 5, 1)$	$2,74\times10^{-3}$	$5,92\times10^{-4}$	$9,01\times10^{-3}$	$6,59\times10^{-3}$
$(8, 10, 1)$	$3,49\times10^{-3}$	$7,20\times10^{-4}$	$6,66\times10^{-3}$	$1,14\times10^{-1}$

El MSE más bajo es presentado por la red $(5, 5, 1)$ antes de compactar, pero después de aplicar este algoritmo, el error y la varianza aumentan significativamente, lo que indica que se pierde gran cantidad de información y de precisión en este proceso.

Por otro lado, el MSE de la red $(8, 10, 1)$ no es mucho más alto que el de la red anterior, pero al compactar, el MSE y la varianza no aumentan tanto, signo de que el algoritmo

[3]*match set*: corresponde al subconjunto de clasificadores existentes en el sistema que se ajustan a un estado dado.

de compactación funciona mejor en esta estructura de red que en las otras presentadas, reduciendo el número de macroclasificadores sin afectar seriamente el desempeño.

Los errores más bajos se obtienen antes de compactar, y en la Figura 4.10 se observa que la compactación disminuye considerablemente la cantidad de macroclasificadores en la población.

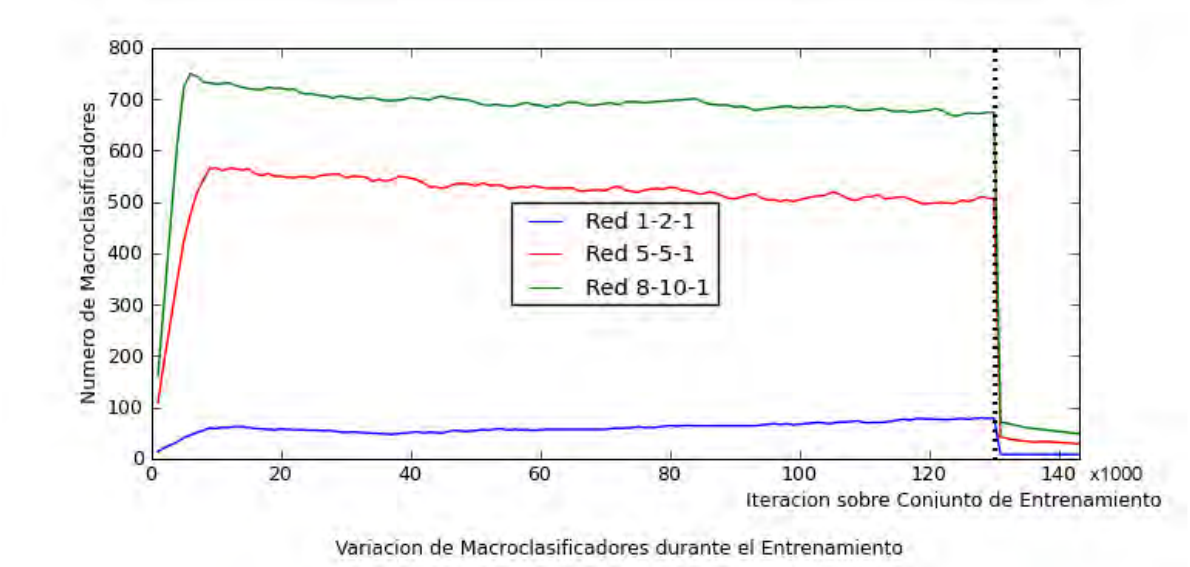

Figura 4.10: Número de Macroclasificadores al terminar cada iteración sobre el conjunto de entrenamiento de la Serie Emerald, representando el comienzo del algoritmo de compactación con una línea vertical punteada.

En cuanto al tamaño promedio del *match set* antes de compactar, se observan en el Cuadro 4.4. La estructura $(1, 2, 1)$, que arroja el peor resultado, posee también el conjunto más grande, lo que sugiere que en este caso CLARE no logra obtener una buena generalización del problema.

Cuadro 4.4: Tamaño promedio del *match set* antes de compactar en la Serie Emerald

Red	Tamaño,
$(1, 2, 1)$	17,29
$(5, 5, 1)$	2,12
$(8, 10, 1)$	3,55

4.3.3 Observaciones Generales sobre la Compactación

De las gráficas sobre ambas series se pueden hacer varias observaciones. En primer lugar, se puede apreciar que la estructura de red $(1, 2, 1)$ no logra aprender ninguna de las series.

Además, se tiene que para las tres estructuras utilizadas, el MSE promedio es más bajo antes de compactar que después de hacerlo, lo que ocurre igualmente con la varianza, lo que indica que este proceso disminuye al precisión de la arquitectura CLARE.

Sin embargo, también se aprecia que la compactación disminuye notable la cantidad de macroclasificadores en la población, con lo que se reduce el tiempo de evaluación de un estado cualquiera. Luego, es necesario plantearse si es conveniente compactar, para mejorar la eficiencia, al costo de disminuir la precisión. Esta decisión depende de las características particulares de cada serie.

Con el tamaño máximo de la población fijo en 1000, y observando que el tamaño promedio del *match set* es bajo, siendo menor a 40 en el peor caso, y notablemente menor en el resto, se determina que no es conveniente aplicar la compactación pues es más importante la precisión al predecir el próximo punto de la serie y la evaluación de los clasificadores no toma mucho tiempo, dado al pequeño tamaño del *match set*.

4.4 Experimento #3: Desempeño de CLARE

En este experimento se prueba el desempeño de CLARE con la estructura de red neuronal que produjo mejores resultados en los experimentos de entonación (ver Apéndice C). Se realizan 10 ejecuciones por cada red neuronal, con el propósito de determinar si las predicciones dadas por el sistema presentan resultados consistentes.

4.4.1 Resultados sobre Serie DJIA

Después de las 10 ejecuciones de CLARE utilizando una red neuronal de estructura $(1, 5, 1)$ se obtuvo que el MSE promedio sobre el conjunto de prueba de la Serie DJIA es de $8,25{\times}10^{-4}$, presentando una varianza de $1,56{\times}10^{-9}$. El MSE máximo que se obtuvo fue de $9,42{\times}10^{-4}$ y el mínimo fue $7,98{\times}10^{-4}$.

La poca diferencia entre el MSE mínimo y el máximo, además del bajo valor de la varianza indican que CLARE logra obtener resultados consistentes, y que es difícil obtener un conjunto de clasificadores que presenten un MSE atípicamente alto sobre la Serie DJIA.

En la Figura 4.11 y la Figura 4.12 se comparan la mejor y peor predicción de CLARE sobre

el conjunto de entrenamiento y el conjunto de prueba de la Serie DJIA, respectivamente. Se puede observar que tanto la peor como la mejor predicción son muy cercanas al valor real.

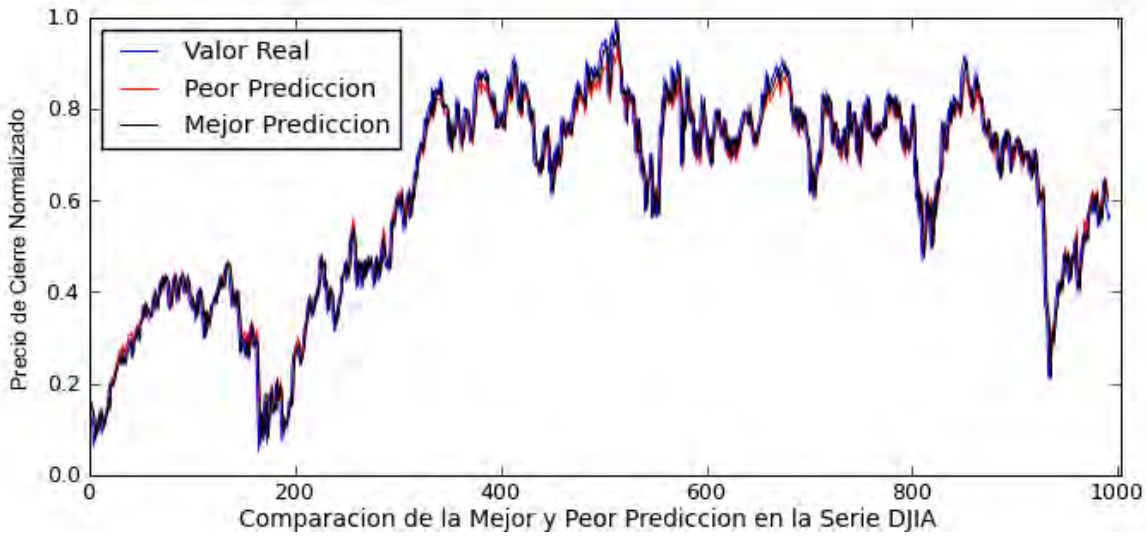

Figura 4.11: Conjunto de entrenamiento de la Serie DJIA, con la mejor y peor predicción dadas por CLARE con red neuronal (1,5,1)

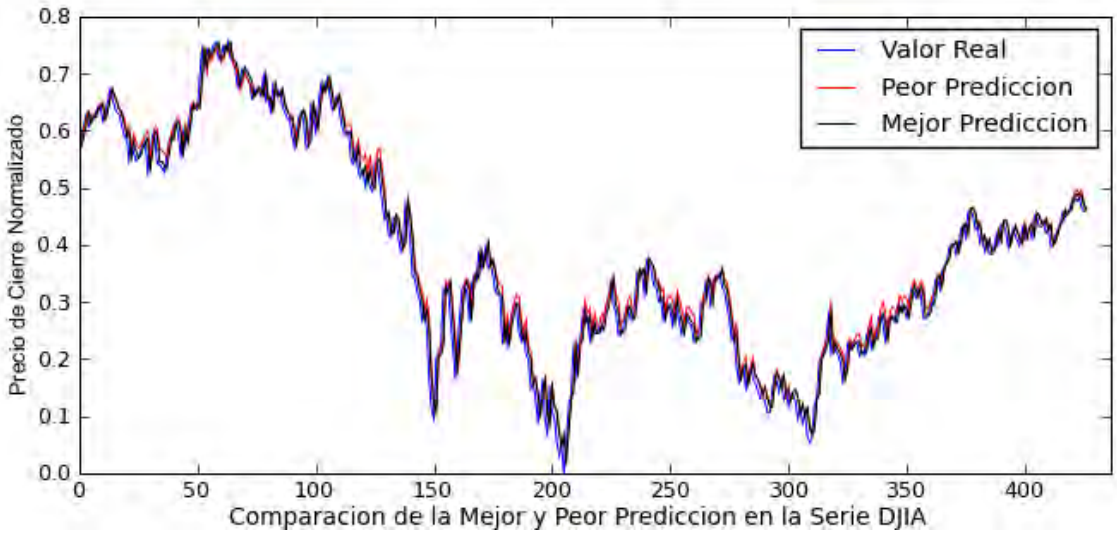

Figura 4.12: Conjunto de prueba de la Serie DJIA, con la mejor y peor predicción dadas por CLARE con red neuronal (1,5,1)

El análisis de los errores, mostrados en la Figura 4.13 y el gráfico de los valores reales contra los valores predichos, en la Figura 4.14 , confirman el hecho de que en ambas ejecuciones las predicciones son bastante acertadas, y que el conjunto de clasificadores aprende bien las características de la serie.

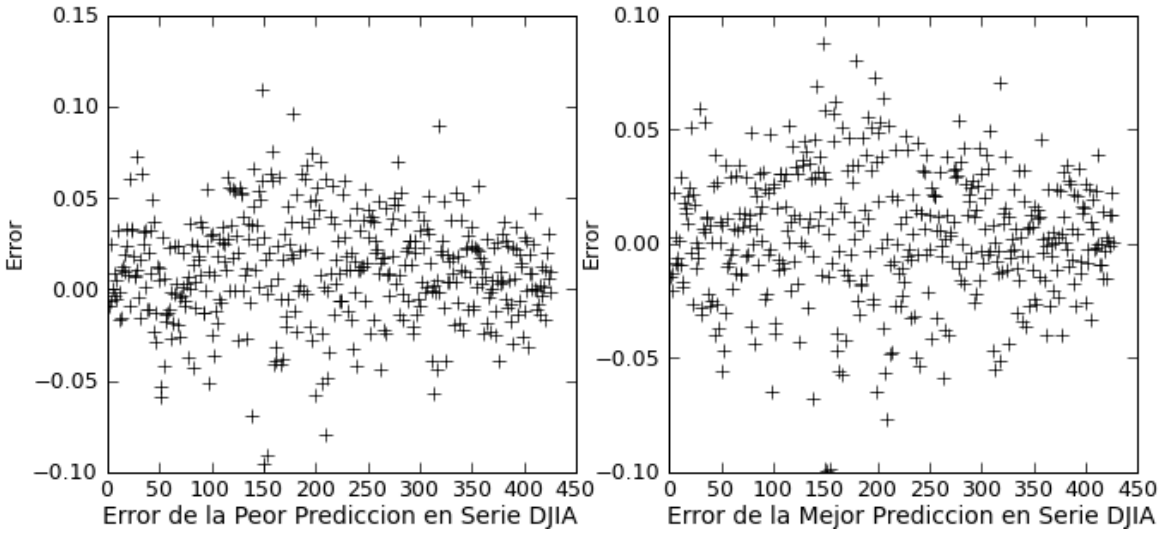

Figura 4.13: Gráfico de error en cada punto de la peor y mejor predicción que obtuvo CLARE en el conjunto de prueba de la Serie DJIA.

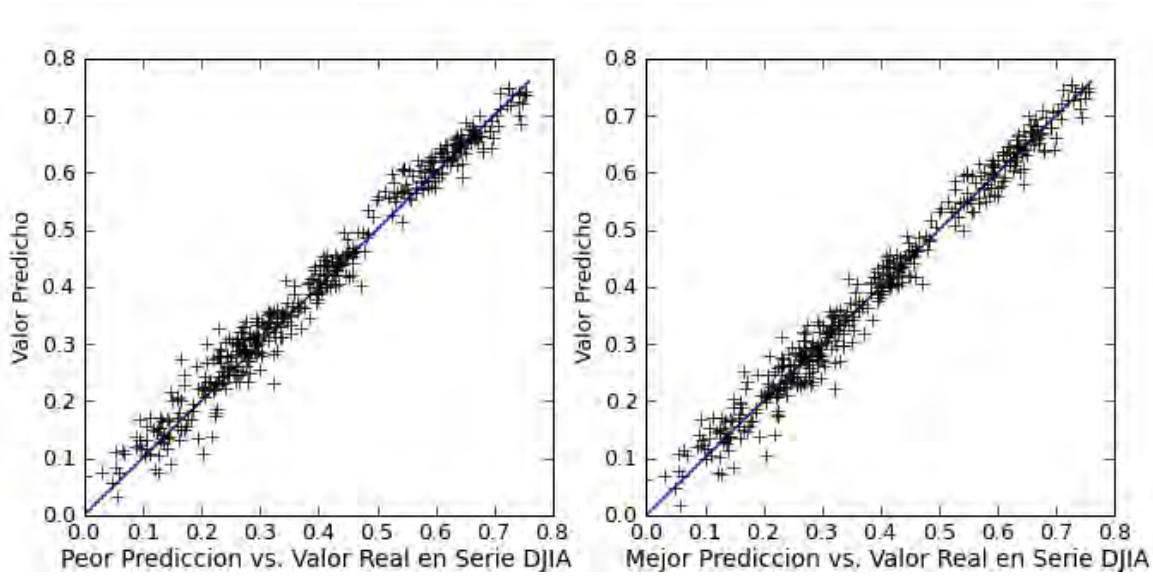

Figura 4.14: Gráfico de valor real contra valor predicho por la peor y mejor predicción que obtuvo CLARE en el conjunto de prueba de la Serie DJIA.

Se observó que el tamaño del conjunto de macroclasificadores que arroja la mejor predicción es de 60, mientras que en la peor predicción es de tamaño 109. Esto quiere decir que en el primer caso, CLARE logró conseguir un conjunto de reglas que generaliza muy bien, lo que no obtuvo en el segundo caso.

4.4.2 Resultados sobre Serie Emerald

Para este experimento fue seleccionada la estructura $(2, 5, 1)$ para la red neuronal de CLARE, y se realizaron 10 ejecuciones de la misma, para posteriormente obtener el MSE de cada una de ellas sobre el conjunto de prueba.

Se obtuvo que el MSE promedio es $4,46\times10^{-4}$, con una varianza de $4,67\times10^{-9}$. El MSE mínimo obtenido fue de $3,86\times10^{-4}$ y el máximo fue de $6,41\times10^{-4}$.

Tanto el MSE mínimo como el máximo se encuentran en el mismo orden de magnitud, y la varianza es muy pequeña, lo que indica que no existen grandes diferencias en el error arrojado en las ejecuciones, y es muestra de que el comportamiento de CLARE sobre la Serie Emerald es consistente.

Las gráficas de las predicciones de CLARE sobre el conjunto de entrenamiento y el conjunto de prueba de la Serie Emerald se pueden apreciar en la Figura 4.15 y la Figura 4.16, respectivamente. Se observa que la mejor y la peor predicción van muy cercanas a la serie real en ambos conjuntos.

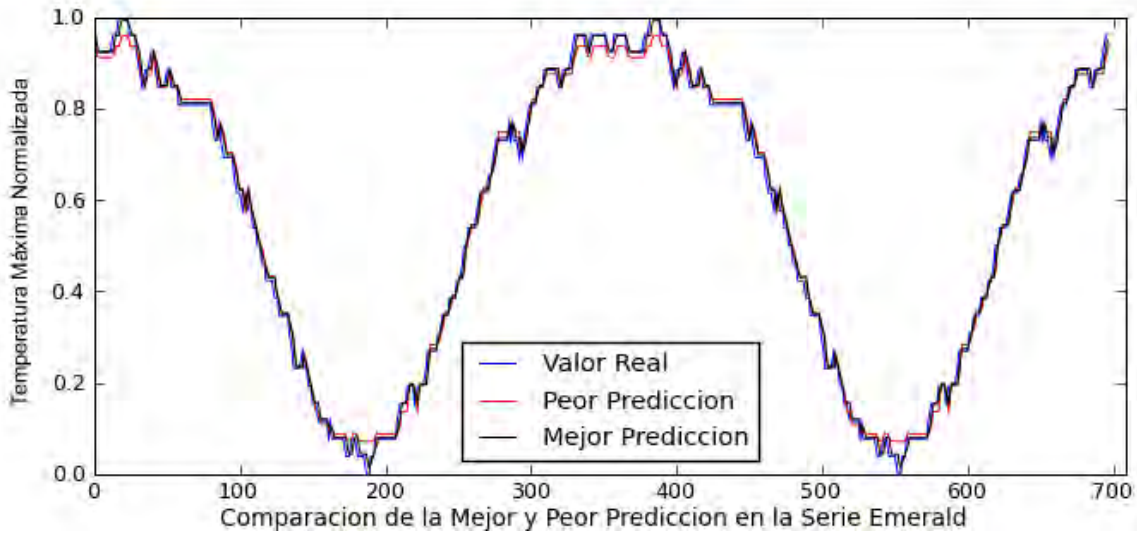

Figura 4.15: Conjunto de entrenamiento de la Serie Emerald, con la mejor y peor predicción dadas por CLARE con red neuronal (2,5,1)

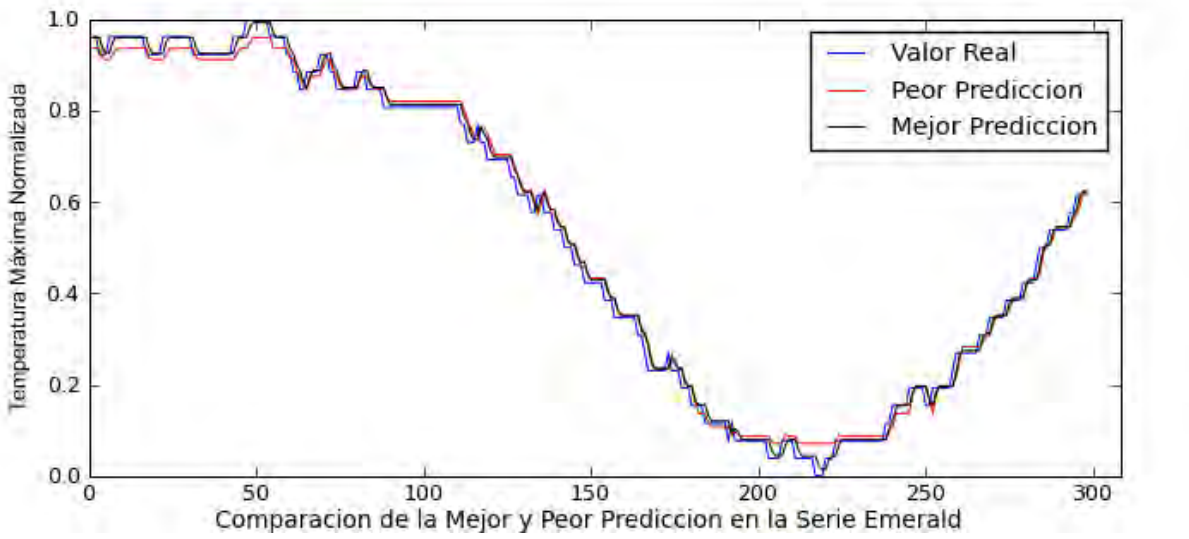

Figura 4.16: Conjunto de prueba de la Serie Emerald, con la mejor y peor predicción dadas por CLARE con red neuronal (2,5,1)

Al observar los errores sobre el conjunto de prueba de los clasificadores con la mejor y peor predicción en la Figura 4.17 se aprecia que los errores de la mejor predicción son más bajos y muchos de ellos forman una línea cerca de 0, pero en ambos no se muestra patrón o estructura evidente.

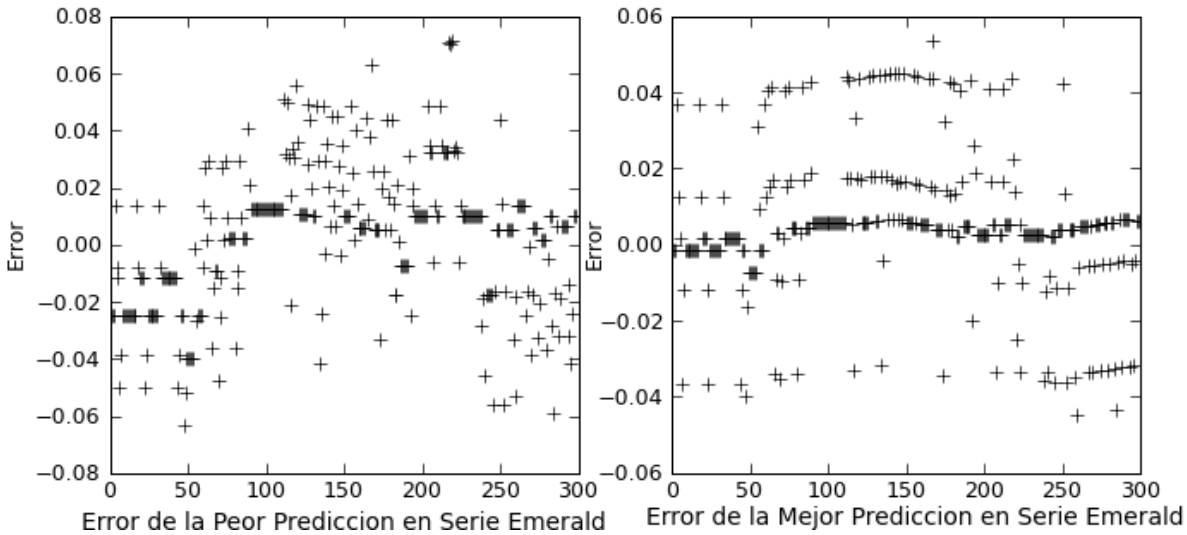

Figura 4.17: Gráfico de error en cada punto de la peor y mejor predicción que obtuvo CLARE en el conjunto de prueba de la Serie Emerald.

La Figura 4.18 muestra el valor real contra el valor predicho. Se observa que en ambos casos, los puntos se encuentran cerca de la recta $f(x) = x$, que indicaría una predicción sin

error. Sin embargo, a la izquierda, se muestra que en la peor predicción, los puntos se alejan de la línea en los extremos, lo que indica que tiene problemas en alcanzar los valores mínimos y máximos de la serie.

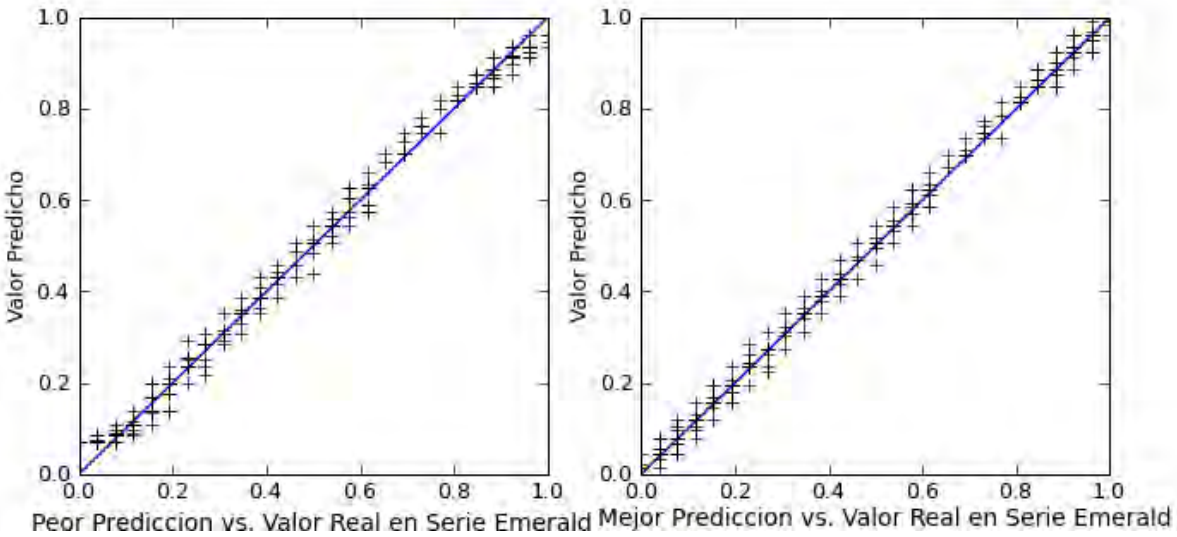

Figura 4.18: Gráfico de valor real contra valor predicho por la peor y mejor predicción que obtuvo CLARE en el conjunto de prueba de la Serie Emerald.

Esta serie es más difícil de aprender para CLARE, lo que se muestra en la cantidad de reglas al finalizar la ejecución. El conjunto de reglas que da la peor clasificación tiene cardinalidad 289 mientras que el que arroja la mejor clasificación tiene 247 reglas.

4.4.3 Observaciones Generales sobre CLARE

Los resultados sobre ambas series permiten apreciar que CLARE logra aprender las características y relaciones sobre las mismas, lo que se refleja en un bajo MSE. Además, las gráficas muestran que el conjunto de clasificadores obtenido por CLARE realiza buenas predicciones, muy cercanas al valor original.

Además, en ambas series, el MSE más alto y el más bajo se encuentran en el mismo orden de magnitud, y los valores de la varianza son pequeños, lo que permite afirmar que CLARE obtiene resultados consistentes en las mismas.

Lo anterior posibilita también afirmar que CLARE es capaz de lograr una partición del espacio del problema que permite obtener redes neuronales adecuadas para predecir cada intervalo.

Una observación interesante es que en ambas series, el conjunto de clasificadores que genera mejores predicciones es de menor cardinalidad de aquel que arroja las peores, lo que sugiere que las buenas predicciones que obtiene CLARE además son buenas generalizaciones.

En las mejores predicciones, CLARE obtiene un tamaño de población de 60 en la Serie DJIA y 247 en la Serie Emerald, lo que indica que con pocas reglas se puede obtener una buena aproximación de una serie de tiempo.

4.5 Comparación: Redes Neuronales vs. SERENA vs. CLARE

En este experimento se realiza un contraste de los resultados arrojados por tres sistemas diferentes sobre el problema de predicción en la Serie DJIA y la Serie Emerald. Las tres arquitecturas a comparar son SERENA, una red neuronal que utiliza estrategias evolutivas; CLARE, sistema de clasificadores genéticos con redes neuronales; y una red neuronal que utiliza el algoritmo de retropropagación.

Para SERENA y CLARE se utilizan los valores del MSE que se obtuvieron en los experimentos anteriores. Para la red neuronal, se realizaron también 10 ejecuciones sobre cada serie, usando una estructura $(1, 5, 1)$ para la Serie DJIA y $(2, 5, 1)$ para la Serie Emerald.

Los valores del MSE sobre el conjunto de prueba de la Serie DJIA y la Serie Emerald se muestran en el Cuadro 4.5 y el Cuadro 4.6, respectivamente.

Cuadro 4.5: Comparación del MSE sobre el conjunto de prueba de la Serie DJIA

Valores del MSE sobre Serie DJIA				
.	Promedio	Varianza	Mínimo	Máximo
Red Neuronal	$5,91\times10^{-3}$	$5,01\times10^{-6}$	$3,41\times10^{-3}$	$9,18\times10^{-3}$
SERENA	$3,62\times10^{-3}$	$5,4\times10^{-5}$	$9,76\times10^{-4}$	$2,56\times10^{-2}$
CLARE	$8,25\times10^{-4}$	$1,56\times10^{-9}$	$7,98\times10^{-4}$	$9,42\times10^{-4}$

Cuadro 4.6: Comparación del MSE sobre el conjunto de prueba de la Serie Emerald

Valores del MSE sobre Serie Emerald				
.	Promedio	Varianza	Mínimo	Máximo
Red Neuronal	$7,77\times10^{-3}$	$9,76\times10^{-6}$	$2,3\times10^{-3}$	$1,37\times10^{-2}$
SERENA	$6,94\times10^{-3}$	$2,17\times10^{-4}$	$1,53\times10^{-3}$	$5,11\times10^{-2}$
CLARE	$4,46\times10^{-4}$	$4,67\times10^{-9}$	$3,86\times10^{-4}$	$6,41\times10^{-4}$

En los cuadros se aprecia que el valor más alto del error promedio pertenece en ambas series a la red neuronal con retropropagación, y el más bajo a CLARE, siendo este un orden de magnitud más bajo que los otros dos. SERENA obtiene un MSE promedio del mismo orden de magnitud que la red neuronal, sin embargo es un poco más bajo.

Observaciones interesantes se derivan de los valores de la varianza. CLARE presenta en ambas series una varianza realmente baja, $1,56\times10^{-9}$ en la Serie DJIA y $4,67\times10^{-9}$ en la Serie Emerald, lo que significa que obtiene resultados consistentes.

Por otro lado, aunque SERENA presenta un error promedio más bajo que la red neuronal, su varianza es más alta, esto indica que si bien SERENA arroja mejores resultados, la exploración que realiza en el espacio de búsqueda la puede conducir a mínimos locales más rápido que el algoritmo de retropropagación. Esta es también la causa de que el MSE máximo de SERENA sea más alto que el de la red neuronal.

La columna del MSE mínimo muestra que CLARE logra obtener la mejor predicción tanto para la Serie DJIA como para la Serie Emerald, seguido por SERENA y finalmente la red neuronal.

De todo esto se puede notar que CLARE presenta un buen desempeño, arrojando mejores resultados que los otros dos sistemas presentado, que obtiene además de forma consistente. SERENA presenta en general mejores resultados que la red neuronal, pero puede obtener también predicciones erradas si no logra encontrar un hiper-plano adecuado para aproximar la función.

En la Figura 4.19 y Figura 4.20 se muestran las mejores y peores predicciones de cada arquitectura sobre el conjunto de entrenamiento de la Serie DJIA. En la Figura 4.21 y la Figura 4.22 se aprecian las mismas sobre el conjunto de prueba de la serie.

Las mejores y peores predicciones de cada arquitectura sobre el conjunto de entrenamiento de la Serie Emerald se aprecian en la Figura 4.23 y Figura 4.24. La Figura 4.25 y la Figura 4.26 muestran las mismas sobre el conjunto de prueba de la serie.

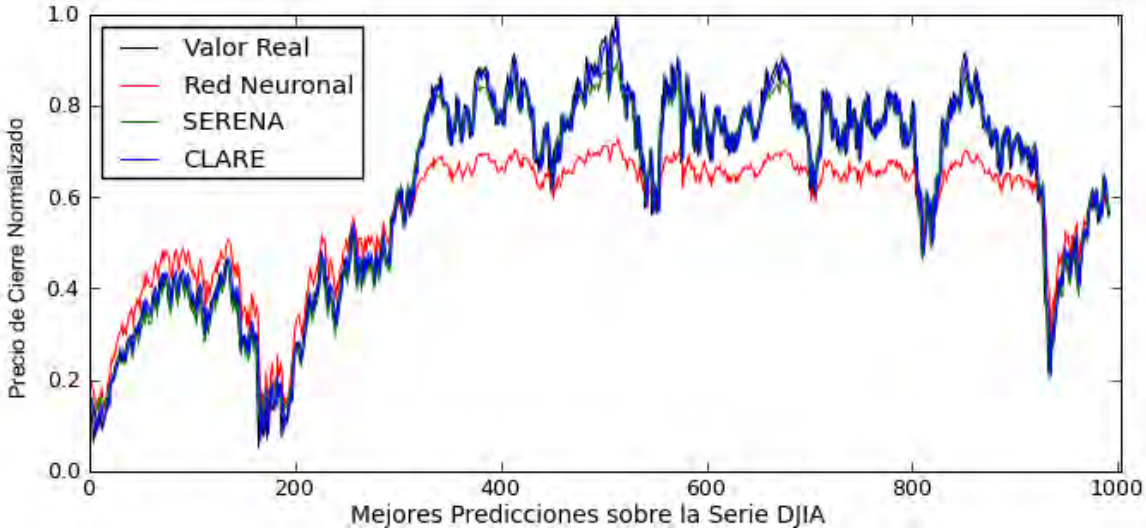

Figura 4.19: Comparación de la mejor predicción obtenida por cada arquitectura sobre el conjunto de entrenamiento de la Serie DJIA.

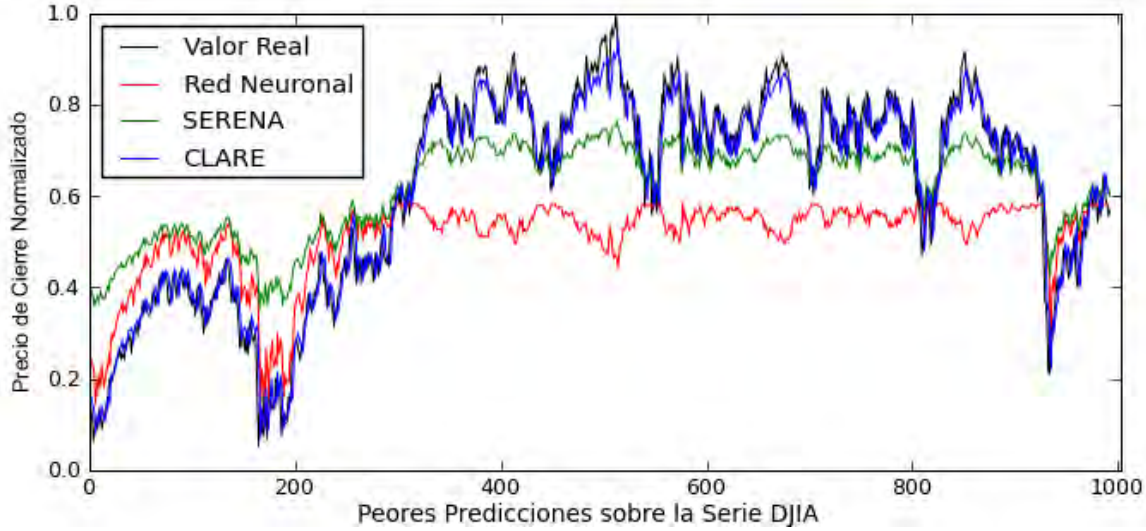

Figura 4.20: Comparación de la peor predicción obtenida por cada arquitectura sobre el conjunto de entrenamiento de la Serie DJIA.

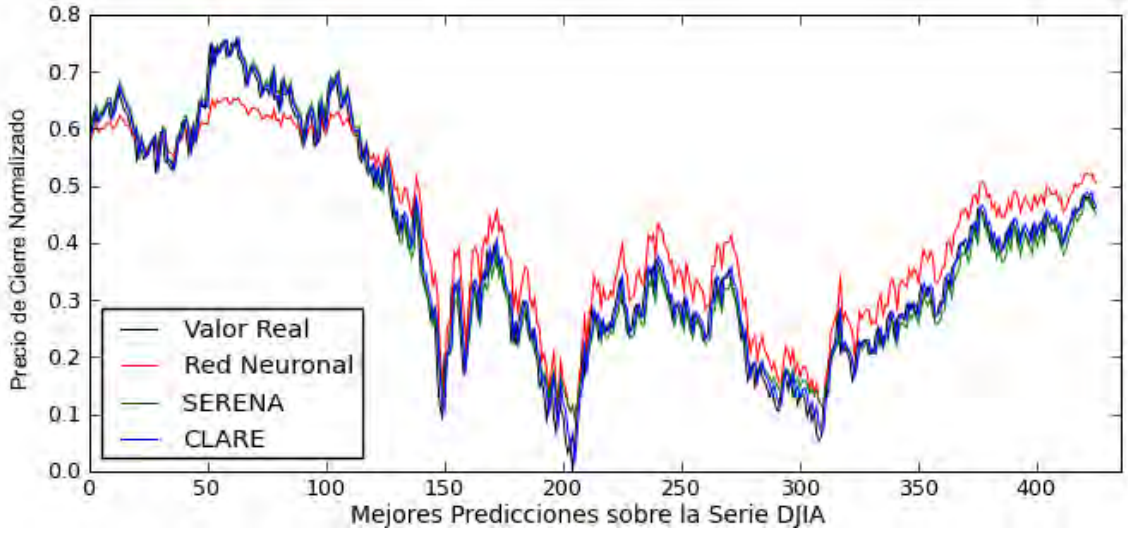

Figura 4.21: Comparación de la mejor predicción obtenida por cada arquitectura sobre el conjunto de prueba de la Serie DJIA.

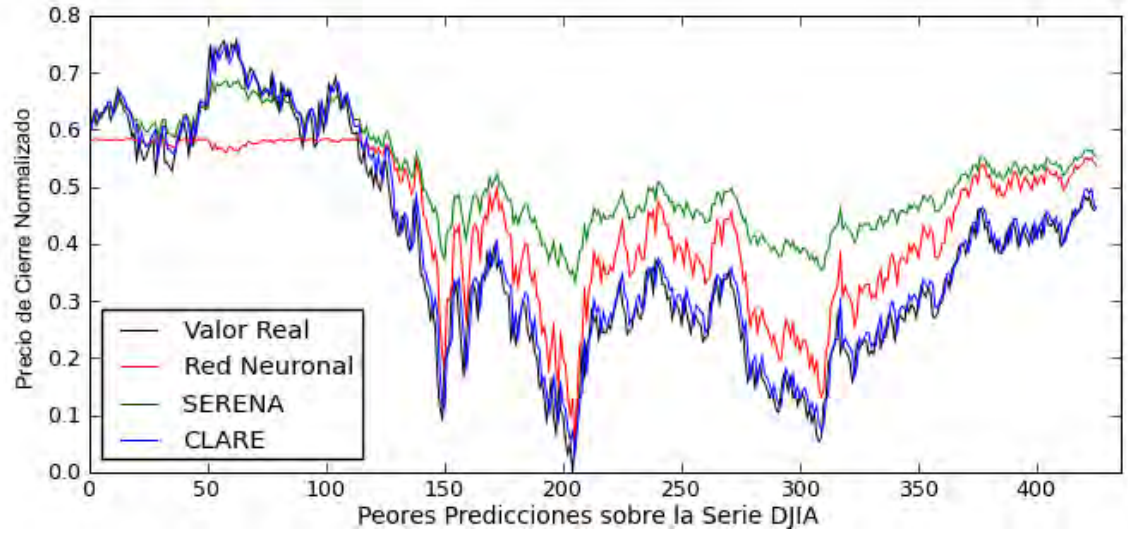

Figura 4.22: Comparación de la peor predicción obtenida por cada arquitectura sobre el conjunto de prueba de la Serie DJIA.

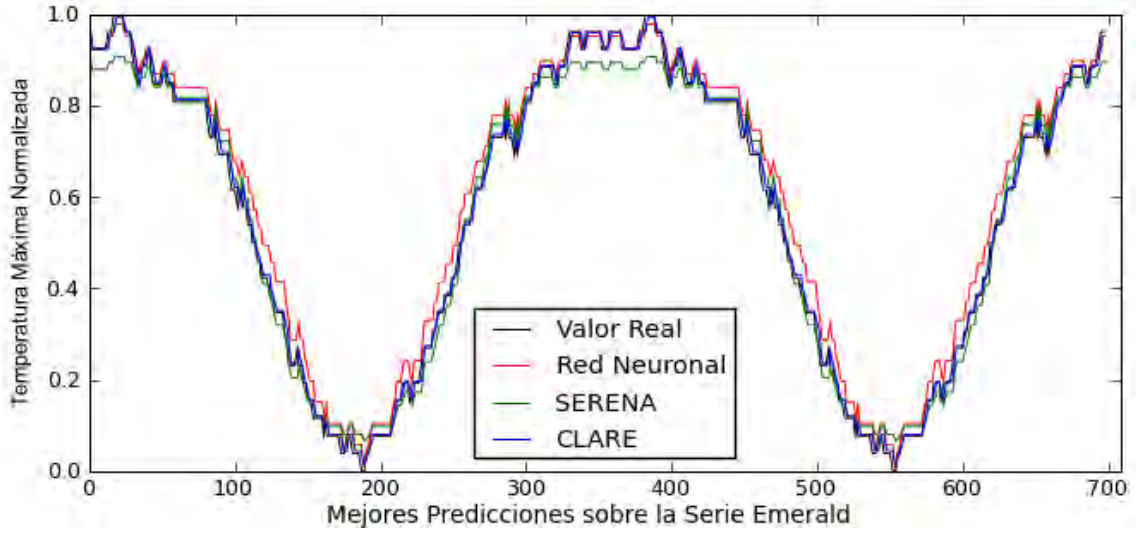

Figura 4.23: Comparación de la mejor predicción obtenida por cada arquitectura sobre el conjunto de entrenamiento de la Serie Emerald.

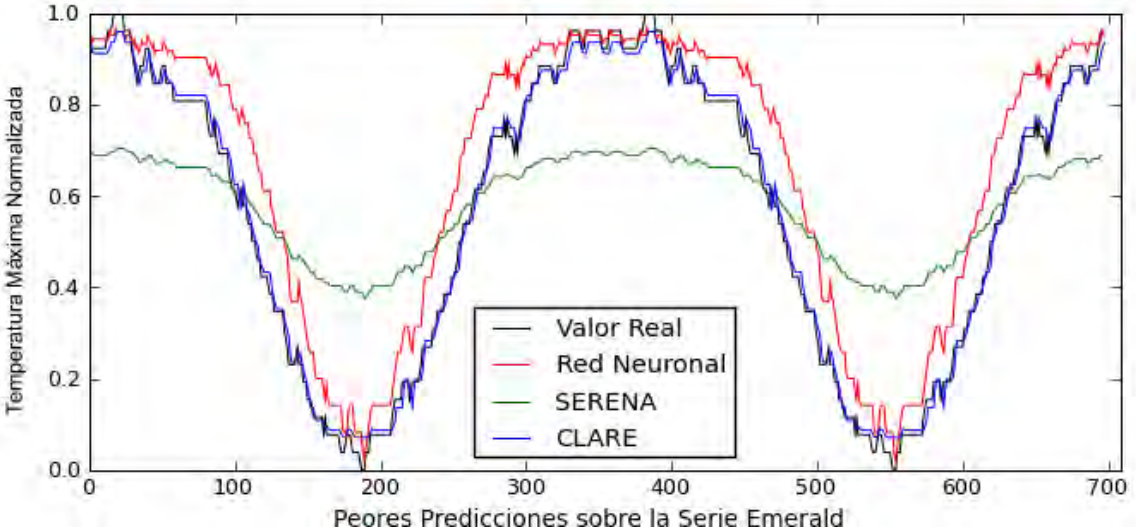

Figura 4.24: Comparación de la peor predicción obtenida por cada arquitectura sobre el conjunto de entrenamiento de la Serie Emerald.

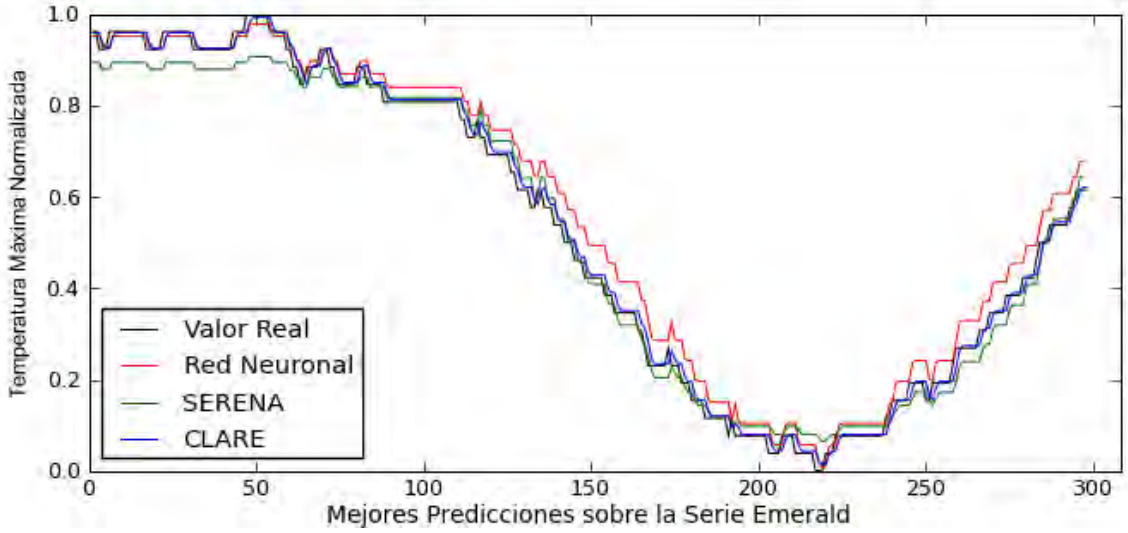

Figura 4.25: Comparación de la mejor predicción obtenida por cada arquitectura sobre el conjunto de prueba de la Serie Emerald.

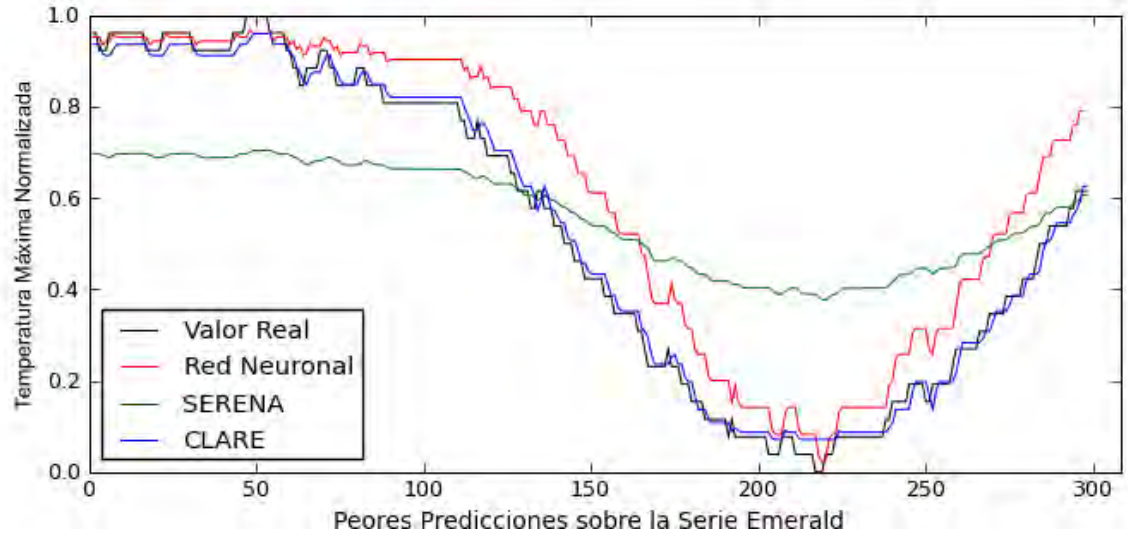

Figura 4.26: Comparación de la peor predicción obtenida por cada arquitectura sobre el conjunto de prueba de la Serie Emerald.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

Se cumplieron los objetivos planteados en la investigación, implementando dos arquitecturas distintas basadas en métodos evolutivos para resolver el problema de la predicción de series de tiempo. La primera de ellas, SERENA, es una red neuronal que utiliza estrategias evolutivas para adaptarse a la serie. La segunda, CLARE, es un sistema de clasificadores genéticos que utiliza redes neuronales para realizar las predicciones.

Ambas arquitecturas logran su propósito, realizando buenas predicciones sobre el comportamiento de las dos series de tiempo de prueba, una de las cuales tiene su origen en el comportamiento del mercado bursátil, la Serie DJIA; mientras que la otra es la temperatura máxima recogida por una estación meteorológica.

Tanto SERENA como CLARE requieren de un proceso de entonación para adecuarse al problema, siendo la estructura de la red neuronal uno de los parámetros que más influye sobre el resultado. Además, un estudio del efecto del algoritmo de compactación sobre CLARE mostró que mejores resultados se obtienen sin ser aplicado, pues cada estado está relacionado con un reducido número de reglas.

Para establecer una base de comparación, se implementó también una red neuronal con retropropagación, una de las técnicas de computación emergente más usada para resolver el problema de la predicción de series de tiempo. Constrastando los resultados obtenidos por esta red, SERENA y CLARE sobre las dos series de tiempo seleccionadas, se observa que CLARE supera considerablemente a los otros dos sistemas en desempeño, obteniendo errores más bajos consistentemente. SERENA, a pesar de no obtener resultados tan buenos como CLARE, presenta una mejora sobre la red neuronal.

La varianza del error muestra también que CLARE obtiene consistentemente buenas predicciones, mientras que SERENA y la red neuronal presentan a veces errores atípicamente altos, como consecuencia de quedarse atrapados en un mínimo local. CLARE combate este problema dividiendo el espacio de búsqueda y manteniendo reglas que hagan predicciones precisas.

De todo esto se entiende que CLARE es la arquitectura que realiza las mejores predicciones sobre las series de tiempo utilizadas en los experimentos, y que es además, una arquitectura estable, capaz de obtener errores bajos en distintas ejecuciones, siempre que se haya escogido una buena estructura para las redes.

Direcciones Futuras

Inicialmente, sería interesante observar el comportamiento de las arquitecturas planteadas sobre series de tiempo diferentes. Además, existen varias modificaciones que se pueden aplicar y que podrían resultar en una mejora en sus predicciones.

En SERENA se observa que ocasionalmente se queda atascado en un mínimo local que arroja malas predicciones, lo que podría arreglarse utilizando mutaciones correlacionadas, donde en vez de un solo parámetro auto-adaptativo que controle el tamaño de la mutacíon, se utiliza uno para cada peso, bajo la idea de que cada uno puede mejorar en dimensiones diferentes del espacio del problema.

Adicionalmente, se podría estudiar el efecto de aplicar diferentes estrategias de selección y agregar un operador de cruce sobre SERENA, con el propósito de realizar una mejor exploración del espacio del problema.

En cuanto a CLARE, un próximo paso podría ser evolucionar la estructura de la red para cada clasificador, de forma que subconjuntos del problema utilicen una red neuronal que se adapte mejor a ella. Otra idea es cambiar el mecanismo de aprendizaje de las redes, sustituyendo el algoritmo de retropropagación por una estrategia evolutiva.

BIBLIOGRAFÍA

[1] Climate indices: Monthly atmospheric and ocean time series. Disponible en http://www.cdc.noaa.gov/data/climateindices/list/ Consultado el 8 de Septiembre de 2009.

[2] N. I. Sapankevych y R. Sankar. Time series prediction using support vector machines: A survey. *Computational Intelligence Magazine, IEEE*, 4(2):24–38, 2009.

[3] Terence C. Mills. *Time Series Techniques for Economists*. Cambridge University Press, 1991.

[4] A.S. Weigend y N.A. Gershenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1993.

[5] Hans-Georg Beyer y Hans-Paul Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1):3–52, March 2002.

[6] Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 2(3):149–175, 1995.

[7] Aranildo Rodrigues Lima Junior. A study for multi-objective fitness function for time series forecasting with intelligent techniques. En *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pag 1843–1846, New York, NY, USA, 2008. ACM.

[8] Stewart W. Wilson. Classifiers that approximate functions. *Natural Computing: an international journal*, 1(2-3):211–234, 2002.

[9] Daniele Loiacono y Pier Luca Lanzi. Evolving neural networks for classifier prediction with xcsf. En *Proceedings of the ECAI'06 Workshop on Evolutionary Computation*, pag 36–40, 2006.

[10] DJI: Historical prices for Dow Jones Industrial Average - Yahoo! Finance. Disponible en http://finance.yahoo.com/q/hp?s=D̂JI Consultado el 6 de Septiembre de 2009.

[11] Meteorological data for some Queensland stations. Disponible en http://www.bom.gov.au/climate/data/weather-data.shtml Consultado el 8 de Septiembre de 2009.

[12] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[13] Zbigniew Michalewicz y David B. Fogel. *How to solve it: modern heuristics*. Springer-Verlag New York, Inc., New York, NY, USA, 2000.

[14] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, March 1997.

[15] Michalewicz Back y David B. Fogel. *Evolutionary computation 1: Basics Algorithms and Operators*. Institute of Physics Publishing, 2000.

[16] John H. Holland. Adaptation. En Robert Rosen y F.M. Snell, editors, *Progress in theoretical biology*, volume 4, pag 263–293. Academic Press, New York, 1976.

[17] John H. Holmes, Pier Luca Lanzi, Wolfgang Stolzmann, y Stewart W. Wilson. Learning classifier systems: new models, successful applications. *Inf. Process. Lett.*, 82(1):23–30, 2002.

[18] Martin Butz y Stewart W. Wilson. An algorithmic description of XCS. En *IWLCS '00: Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems*, pag 253–272, London, UK, 2001. Springer-Verlag.

[19] Stewart W. Wilson. Get real! XCS with continuous-valued inputs. En *Learning Classifier Systems, From Foundations to Applications*, pag 209–222, Londres, 2000. Springer-Verlag.

[20] Christopher Stone y Larry Bull. For real! XCS with continuous-valued inputs. *Evol. Comput.*, 11(3):299–336, 2003.

[21] María Franco y Celso Gorrín. Diseño e implementación de un agente de corretaje en una cadena de suministros en un ambiente simulado. 2007.

[22] Bernard Widrow y Marcian E. Hoff. Adaptive switching circuits. pag 123–134, 1988.

[23] Pier Luca Lanzi, Daniele Loiacono, Stewart W. Wilson, y David E. Goldberg. Extending XCSF beyond linear approximation. En *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pag 1827–1834, New York, NY, USA, 2005. ACM.

[24] Daniele Loiacono, Andrea Marelli, y Pier Luca Lanzi. Support vector regression for classifier prediction. En *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pag 1806–1813, New York, NY, USA, 2007. ACM.

[25] Martin V. Butz. Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. En *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pag 1835–1842, New York, NY, USA, 2005. ACM.

[26] Martin V. Butz, Pier Luca Lanzi, y Stewart W. Wilson. Hyper-ellipsoidal conditions in XCS: rotation, linear approximation, and solution structure. En *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pag 1457–1464, New York, NY, USA, 2006. ACM.

[27] Martin V. Butz, Pier Luca Lanzi, y Stewart W. Wilson. Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Transactions on Evolutionary Computation*, 12:355–376, 2008.

[28] A. E. Eiben y J. E. Smith. *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, October 2008.

[29] Martin V. Butz. Documentation of XCSF-Java 1.1 plus Visualization. MEDAL Report 20070008, Missouri Estimation of Distribution Algorithms Laboratory, University of Missouri in St. Louis, MO, 2007.

[30] Martin V. Butz. The XCSF classifier system in java. *SIGEVOlution*, 2(2):10–13, 2007.

APÉNDICE A

TABLAS DE PARÁMETROS

En esta sección se encuentra el conjunto de parámetros definitivos utilizados en todos los experimentos mostrados en esta investigación. Estos parámetros fueron seleccionados después de un proceso de entonación en el cual se realizaron varias ejecuciones de SERENA y CLARE sobre las series de tiempo, hasta obtener un conjunto que diera buenos resultados.

Parámetros de SERENA

Los parámetros utilizados por SERENA en el Experimento #1 (ver Sección 4.2) se muestran en el Cuadro A.1.

Cuadro A.1: Parámetros de SERENA

Parámetro	Valor	Descripción
num_mutaciones	2500	Número de mutaciones realizadas sobre el cromosoma padre
c	$0,875$	Constante utilizada para escalar el tamaño de la mutación en la regla de 1/5 de Rechenberg
u	$0,001$	Umbral utilizado para determinar si un cromosoma es mejor que otro
g	10	Máximo de iteraciones permitidas sin encontrar un mejor individuo que el padre
ε	$0,01$	Valor más bajo que puede alcanzar el tamaño de la mutación

Parámetros de CLARE

Los parámetros utilizados por SERENA en el Experimento #2 (ver Sección 4.3) y Experimento #3 (ver Sección 4.4) se muestran en el Cuadro A.2. Estos parámetros incluyen los originales de XCSF y los que fueron agregados en CLARE.

Cuadro A.2: Parámetros de CLARE

Parámetros originales de XCSF		
Parámetro	Valor	Descripción
$maxLearningIterations$	100000	Máximo número de iteraciones.
$maxPopSize$	1000	Máximo número de microclasificadores en la población
α	$0,1$	Tasa de aprendizaje para modificar la precisión de un clasificador
β	$0,5$	Tasa de aprendizaje para actualizar la utilidad y el error de predicción de un clasificador
η	$0,05$	Tasa de aprendizaje utilizada en el algoritmo de retro-propagación.
δ	$0,1$	Fracción de la utilidad promedio de la población bajo la cual la utilidad de un clasificador es tomado en cuenta en el cálculo de la probabilidad de eliminarlo.
Parámetros de *Matching* y Compactación		
Parámetro	Valor	Descripción
$startCompaction$	90000	Número de iteraciones antes de comenzar el algoritmo de compactación.
Θ_M	20	Cardinalidad del conjunto de clasificadores más cercanos.
Parámetros de Condiciones		
Parámetro	Valor	Descripción
θ_m	$0,7$	Mínima actividad de un clasificador requerida para decir que cubre un estado y agregarlo al *match set*.
$coverConditionRange$	1	Variación en tamaño de condiciones creadas aleatoriamente.
$minConditionStretch$	0	Tamaño mínimo de una condición.
μ_0	$0,5$	Rango de inicialización para los valores en condiciones hiper-elisoipdales.

Parámetros de Error y Utilidad de Clasificadores		
Parámetro	Valor	Descripción
ν	5	Exponente de la función fuerza para evaluación de la utilidad de un clasificador.
ε_0	$0,1$	Umbral del error a partir del cual la precisión se un clasificador se vuelve 1.
$predictionErrorReduction$	1	Reducción de la predicción del error cuando se crea un nuevo clasificador.
$fitnessReduction$	$0,1$	Reducción de la utilidad cuando se crea un nuevo clasificador.
$predictionErrorIni$	0	Error inicial que se asigna a un clasificador recién creado.
$fitnessIni$	$0,01$	Utilidad inicial que se asigna a un clasificador recién creado.
Parámetros del Algoritmo Genético		
Parámetro	Valor	Descripción
θ_{GA}	20	Umbral para la aplicación del algoritmo genético.
pX	$0,95$	Probabilidad de realizar el cruce entre clasificadores.
pM	$0,05$	Probabilidad de mutar un clasificador.
θ_{del}	20	Umbral sobre el cual la utilidad de un clasificador debe considerarse en la probabilidad de ser borrado.
θ_{sub}	50	La experiencia requerida por un clasificador para subsumir a otros.
$doGASubsumption$	$true$	Determina si se realiza subsunción durante el algoritmo genético.
$doActionSetSubsumption$	$false$	Determina si se realiza subsunción en el *action set*.

APÉNDICE B

SERIES DE TIEMPO

B.1 Precio de Cierre del DJIA

Se escogió como primera serie al conjunto de los precios de cierre del DJIA[1]en el período comprendido entr el 1 de enero de 1998 y el 26 de agosto de 2003, obteniendo una serie de 1420 puntos. Estos datos se obtuvieron del website de Yahoo! Finanzas [10].

Esta serie de tiempo normalizada se muestra en la Figura B.1, tiene como valor mínimo 0 y como máximo 1. También tiene mediana de $0,583$, con una media de $0,546$ y una varianza de $0,055$.

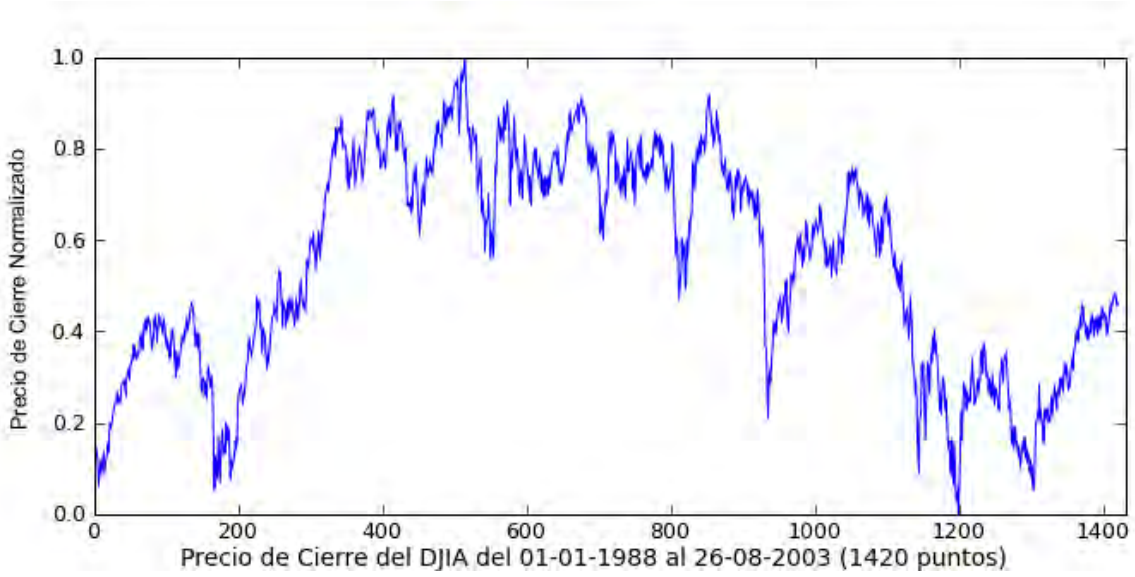

Figura B.1: Serie de tiempo correspondiente al precio de cierre del DJIA en el período comprendido entre 01-01-1988 y 26-08-2003 normalizada al intervalo $[0; 1]$

[1]DJIA: *Dow Jones Industrial Average*. Promedio Industrial Dow Jones, es un índice del mercado bursátil

Además, se puede ver que la serie no presenta algún patrón en particular observable a simple vista, como es característico de las series relacionadas con el comportamiento del mercado bursátil, sino que, al contrario, presenta altos y bajos muy frecuentes.

La escogencia de esta serie pretende evaluar las arquitecturas sobre un conjunto de valores que presenta numerosas fluctuaciones cuyo origen viene de la interacción entre seres humanos.

B.2 Temperatura más alta en la estación climática de Emerald, Australia

En la segunda serie de tiempo escogida, recolectada por el Departamento de Meteorología del Gobierno de Australia [11], cada punto corresponde a la temperatura máxima diaria tomada por una estación meteorológica ubicada en Emerald, Australia.

Es fácil de ver, de acuerdo al gráfico presentado en la Figura B.2 que la serie presenta un comportamiento sinusoidal, lo que se corresponde a las variaciones de la temperatura de acuerdo a las distintas estaciones del año, que ascienden en época de verano y descienden en época de invierno.

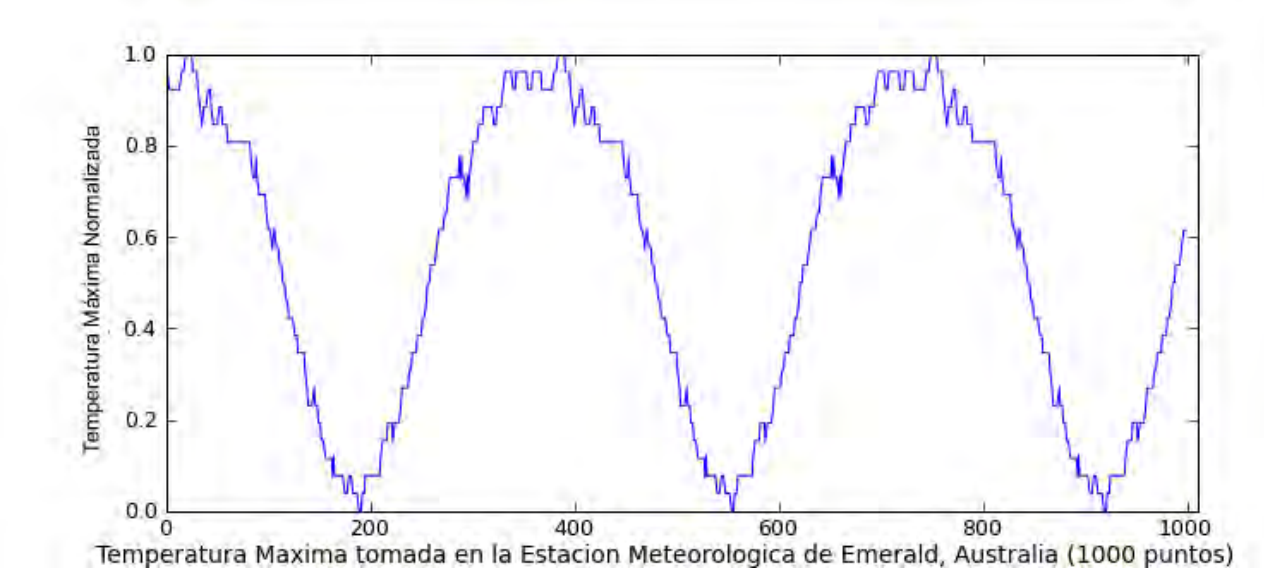

Figura B.2: Serie de tiempo correspondiente a la temperatura máxima tomada en la estación meteorológica de Emerald, Australia normalizada al intervalo $[0; 1]$

El primer punto de esta serie fue tomado el 1 de enero de 1889 y se extiende hasta la actualidad, pero para propósitos de esta investigación se escogieron los mil primeros puntos de la misma, lo que cubre más de dos años de datos. Su valor mínimo es 0 y el máximo es 1. Presenta una mediana de $0,615$ con una media de $0,533$ y una varianza de $0,107$.

APÉNDICE C

EXPERIMENTOS DE ENTONACIÓN

C.1 Experimento de Entonación#1: Estructura de la Red Neuronal en SERENA

El objetivo de este experimento es observar el comportamiento de SERENA con distintas estructuras de red. Con este propósito se escogen valores aleatorios para el número de neuronas en la capa de entrada y en la capa oculta, ambos escogidos con distribución uniforme del intervalo $[1 \ldots 10]$.

Se hacen 30 ejecuciones en total sobre cada una de las series. La escogencia de cuál presenta un mejor desempeño se basa en el valor del MSE sobre el conjunto de prueba.

C.1.1 Resultados sobre Serie DJIA

Al ejecutar SERENA 30 veces sobre la Serie DJIA y calcular el MSE sobre el conjunto de prueba utilizando cada una de las redes neuronales, obtenemos que el MSE promedio es de $8,58\times10^{-3}$, con una varianza de $2,7\times10^{-4}$.

Estos valores indican que en general, la arquitectura funciona muy bien sobre esta serie, pero debido a su naturaleza estocástica, existen estructuras y ejecuciones que generan predicciones mucho mejores que otras.

El error más alto se obtuvo con una red neuronal de estructura $(6, 1, 1)$, que presentó un MSE de 7×10^{-2}; mientras que el error más bajo lo obtuvo una red neuronal de estructura $(1, 5, 1)$, con un MSE de 1×10^{-3}.

Las predicciones de ambas redes sobre el conjunto de entrenamiento y el conjunto de prueba se pueden observar en la Figura C.1 y la Figura C.2, respectivamente.

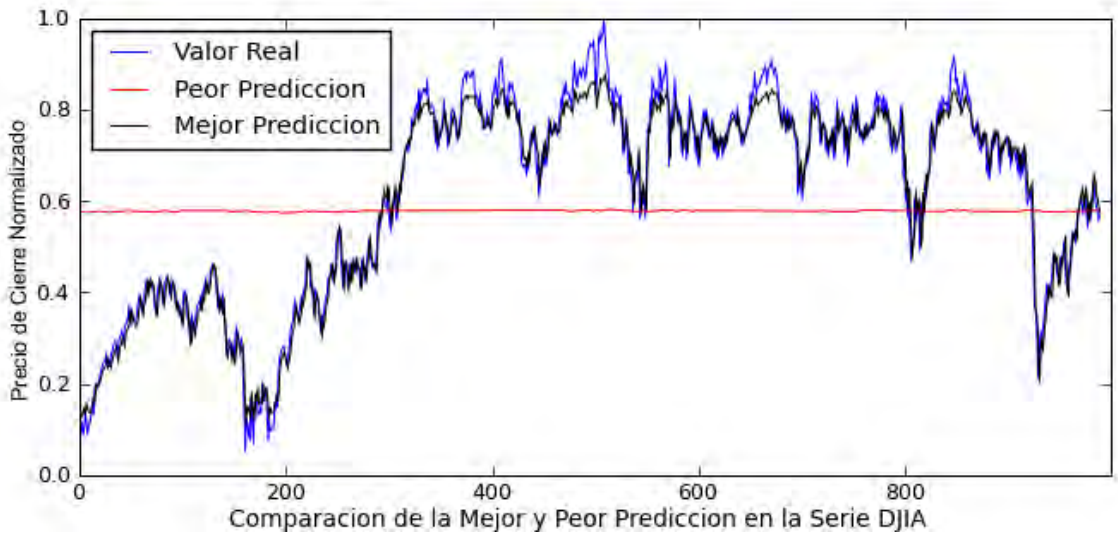

Figura C.1: Conjunto de entrenamiento de la Serie DJIA, con la mejor y peor predicción dadas por SERENA

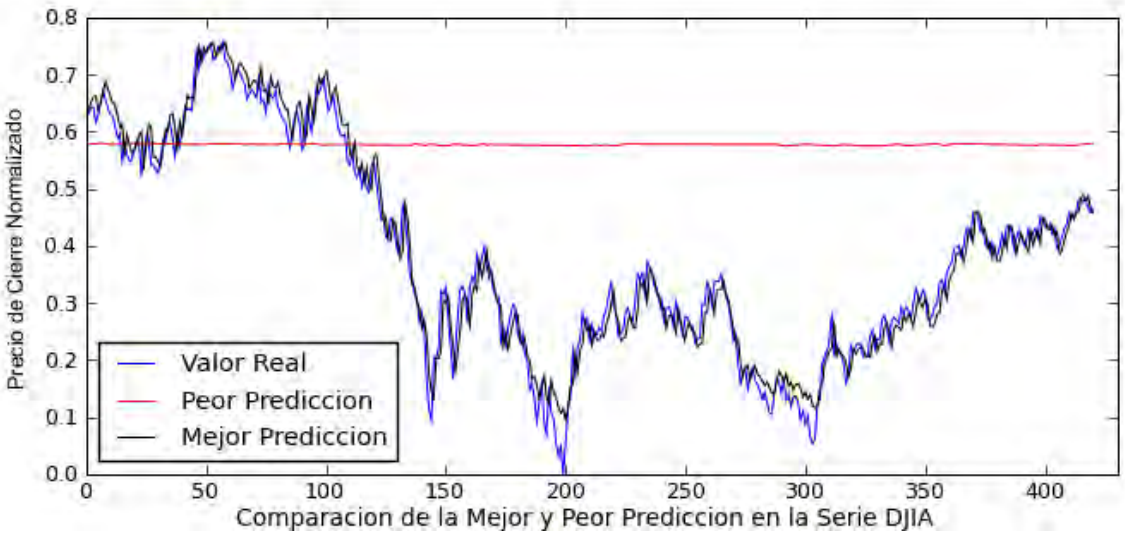

Figura C.2: Conjunto de prueba de la Serie DJIA, con la mejor y peor predicción dadas por SERENA

La peor predicción corresponde a una línea recta, mientras que la mejor pasa muy cerca de la serie original, llegando a superponerse en varios lugares.

Al graficar los errores de la predicción en cada uno de los puntos del conjunto de pruebas se obtiene la Figura C.3, donde es posible observar que en la red neuronal $(6, 1, 1)$ los mismos

presentan cierto patrón, lo que indica que el sistema no logró captar alguna relación o característica de la serie; mientras que la red neuronal $(1, 5, 1)$, que obtuvo la mejor predicción, se puede ver que los errores residuales no presentan estructura notable.

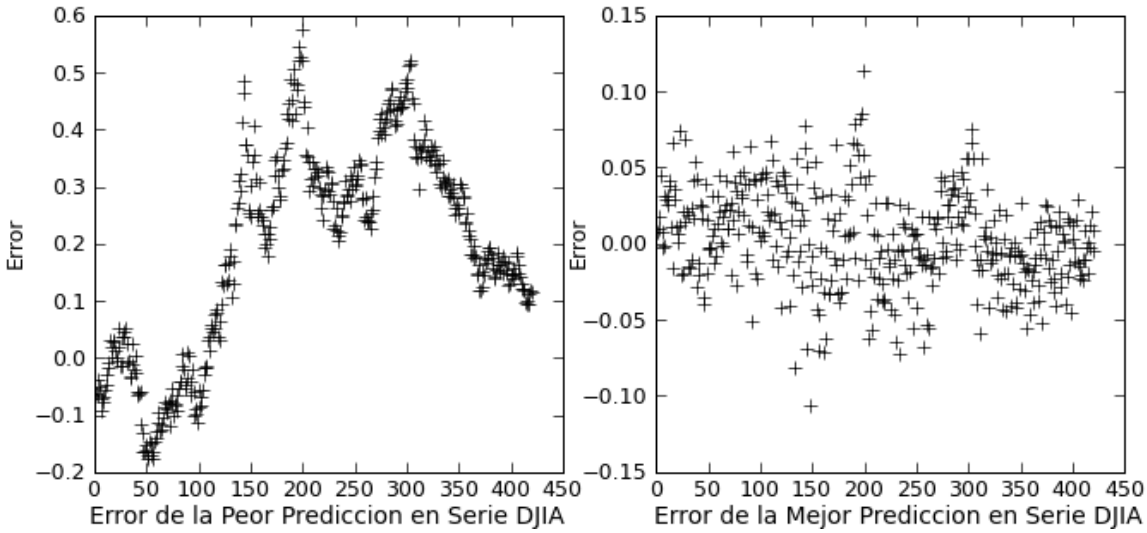

Figura C.3: Gráfico de error en cada punto de la peor y mejor predicción que obtuvo SERENA en el conjunto de prueba de la Serie DJIA

Otra evaluación del desempeño del sistema consiste en graficar los valores reales contra los valores predichos por el mismo. En el caso de una predicción perfecta, estos valores serían iguales, por lo que se ajustarían a la recta $f(x) = x$.

En la Figura C.4 se presenta la recta ideal, y se puede observar que la red neuronal que genera la peor predicción presenta una línea horizontal que apenas la corta, mientras que en la gráfica de la red neuronal que obtiene la mejor predicción se ve que los puntos se acercan mucho a la recta, lo que muestra que da buenas predicciones.

Sin embargo, se puede notar que para valores menores a $0,1$, las predicciones de la red $(1, 5, 1)$ se alejan de la recta, lo que nos permite advertir que la red neuronal no detecta picos que descienden a valores muy bajos, lo que se confirma en la Figura C.2, alrededor de los puntos 200 y 300 del conjunto de prueba.

Al hacer un análisis detallado de las estructuras y los MSE de las mismas, se puede observar que aquellas que tienen sólo una neurona de entrada son las que presentan mejor desempeño, lo que implica que para la Serie DJIA basta solamente tener conocimiento del valor de un día para hacer una buena predicción del día siguiente.

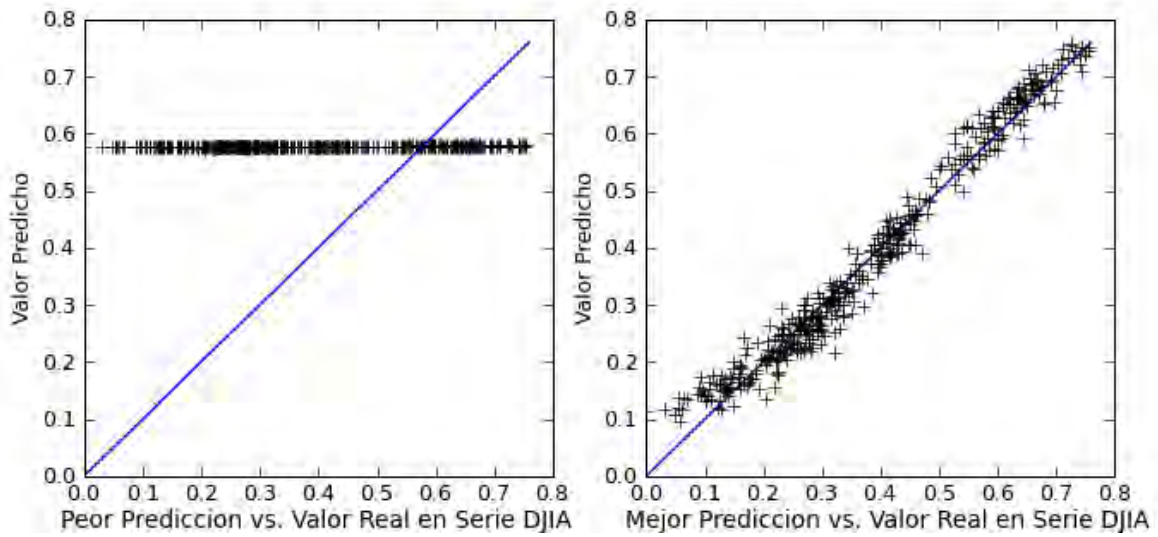

Figura C.4: Gráfico de valor real contra valor predicho por la peor y mejor predicción que obtuvo SERENA en el conjunto de prueba de la Serie DJIA. Se presenta la recta $f(x) = x$ como la línea de color azul.

C.1.2 Resultados sobre Serie Emerald

En la Figura C.5 y la Figura C.6 se presentan las predicciones sobre el conjunto de entrenamiento y de prueba, respectivamente, de la mejor y peor red obtenidas en 30 ejecuciones de SERENA sobre la Serie Emerald.

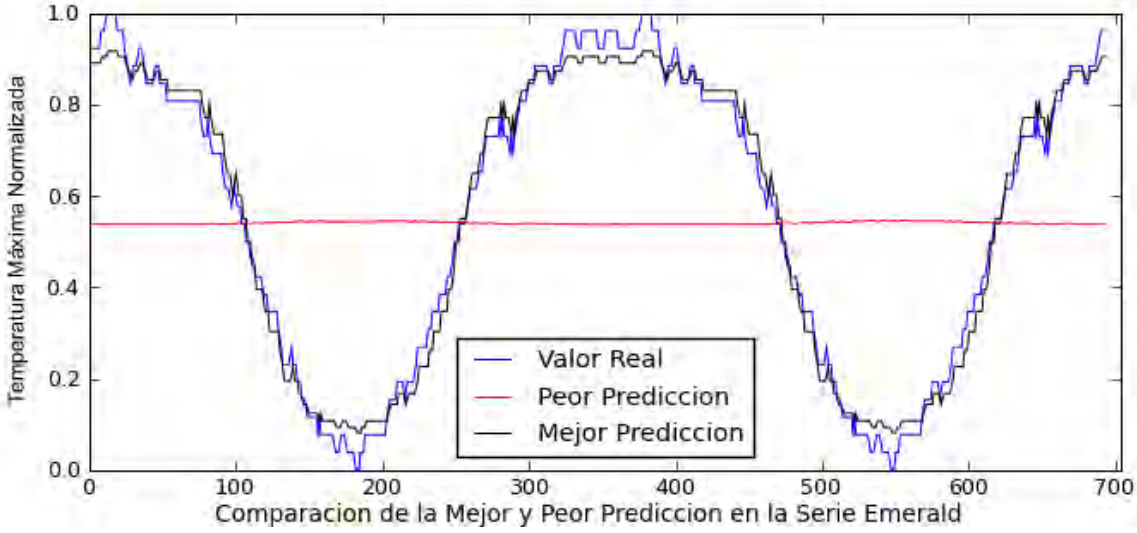

Figura C.5: Conjunto de entrenamiento de la Serie Emerald, con la mejor y peor predicción dadas por SERENA

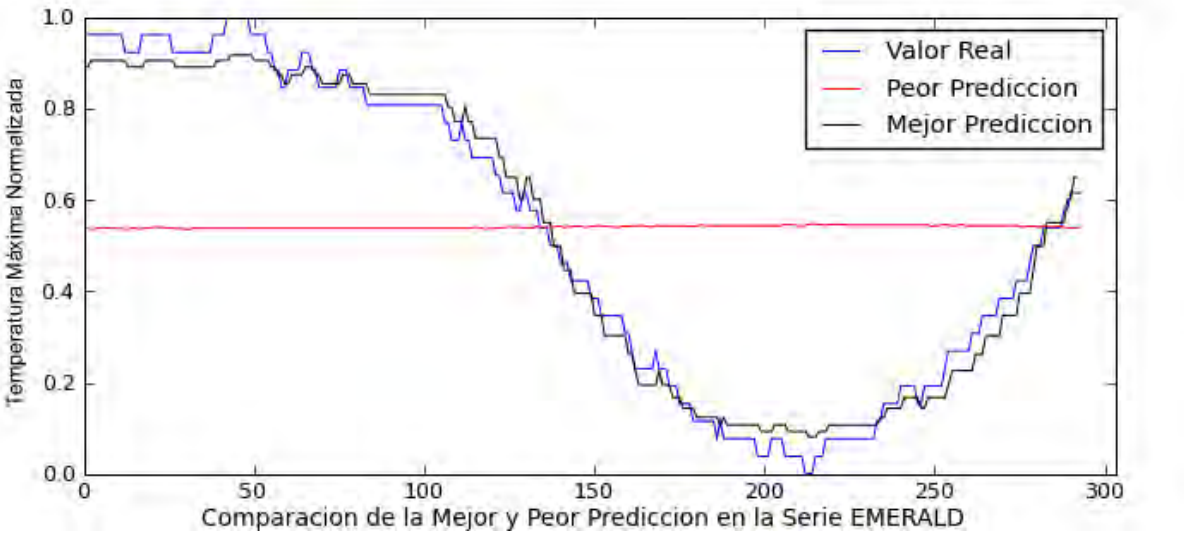

Figura C.6: Conjunto de prueba de la Serie Emerald, con la mejor y peor predicción dadas por SERENA

Con las 30 ejecuciones de SERENA, se procedió a calcular el MSE de las predicciones en el conjunto de prueba de la misma, obteniendo un MSE promedio $9,24\times10^{-3}$, con una varianza de $5,38\times10^{-4}$. La red neuronal que realizó las mejores predicciones tiene una estructura $(2,2,1)$, y presentó un MSE sobre el conjunto de prueba de $1,67\times10^{-3}$.

Cabe destacar que de las 30 ejecuciones, dos de ellas presentaron valores de MSE atípicos, una de las cuales mostró el MSE más alto, de $1,14\times10^{-1}$, obtenido por una red neuronal de forma $(7,7,1)$, mientras que otra red neuronal, de estructura $(3,5,1)$, obtuvo un MSE de $7,23\times10^{-2}$.

Pero a pesar de que en los gráficos muestran que la mejor predicción está siempre muy cercana al valor real de la serie, es necesario hacer un mejor estudio de los errores, comenzando por el error en cada punto. Al calcular estos valores y realizar un gráfico de los mismos, se obtiene la Figura C.7, de donde se puede obtener varias observaciones interesantes.

Tanto en el gráfico de la peor predicción como en la de mejor predicción se distinguen claramente ciertos patrones formados por los errores, pero diferentes entre sí. A la izquierda se tienen errores en el intervalo $(-0,6\ldots0,6)$, que representan diferencias muy grandes. El patrón que presentan los errores se puede observar claramente, y es similar al patrón de onda que presenta la serie original, lo que indica que la red no logró captar una de las características fundamentales de la misma.

Los errores de la mejor predicción son mucho más pequeños, en el intervalo $(-0,1 \ldots 0,1)$, y presentan también cierto patrón, menos evidente, que indica que SERENA falló en contemplar alguna cualidad de la serie, pero de menor importancia, por lo que el MSE es más bajo.

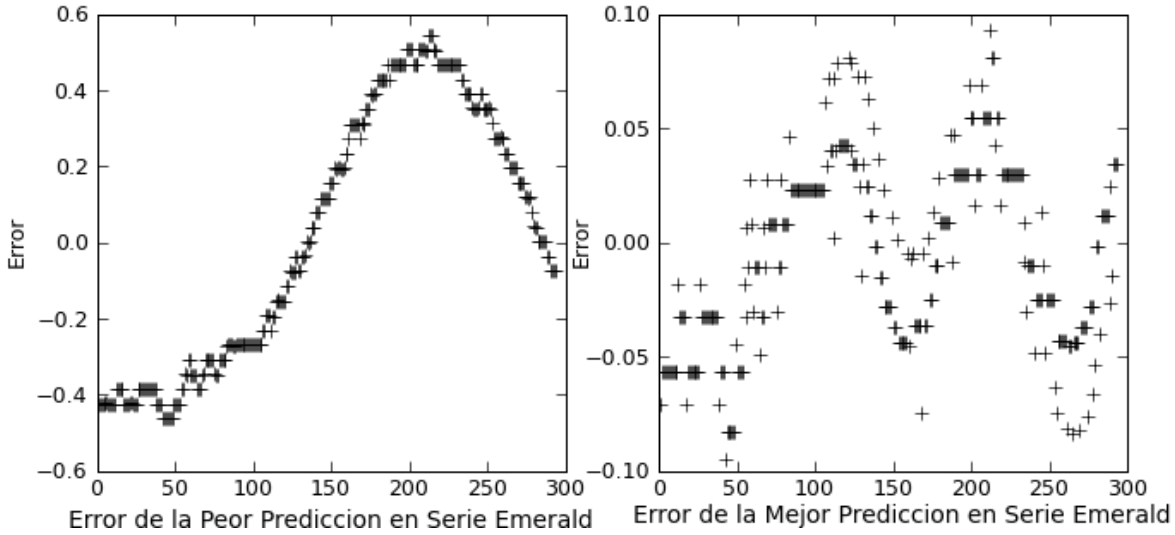

Figura C.7: Gráfico de error en cada punto de la peor y mejor predicción que obtuvo SERENA en el conjunto de prueba de la Serie Emerald

El gráfico de los valores reales contra los valores predichos en el conjunto de prueba de la Serie Emerald, presentado en la Figura C.8 , puede dar más datos sobre el comportamiento del sistema, al determinar la relación que tienen con la recta $f(x) = x$, que corresponde a una predicción perfecta.

Se aprecia que en la predicción de la red $(7, 7, 1)$, se forma una línea horizontal que indica que para cualquier valor real, se obtienen predicciones similares, lo que es consistente con lo observado en la Figura C.5 la Figura C.6, donde la peor predicción genera una línea horizontal.

Es más interesante lo mostrado por la gráfica de los valores predichos por la red $(2, 2, 1)$, donde se observa que los puntos son cercanos a la línea ideal, pero presentan también cierto patrón simétrico; para valores cercanos a 0 y en el intervalo $(0,6 \ldots 0,8)$, la arquitectura tiende a sobrestimar los valores, mientras que para valores cercanos a 1 y en el intervalo $(0,2 \ldots 0,4)$, los valores son subestimados. Esto, nuevamente, es indicativo de que el sistema no logró captar una cualidad de la serie.

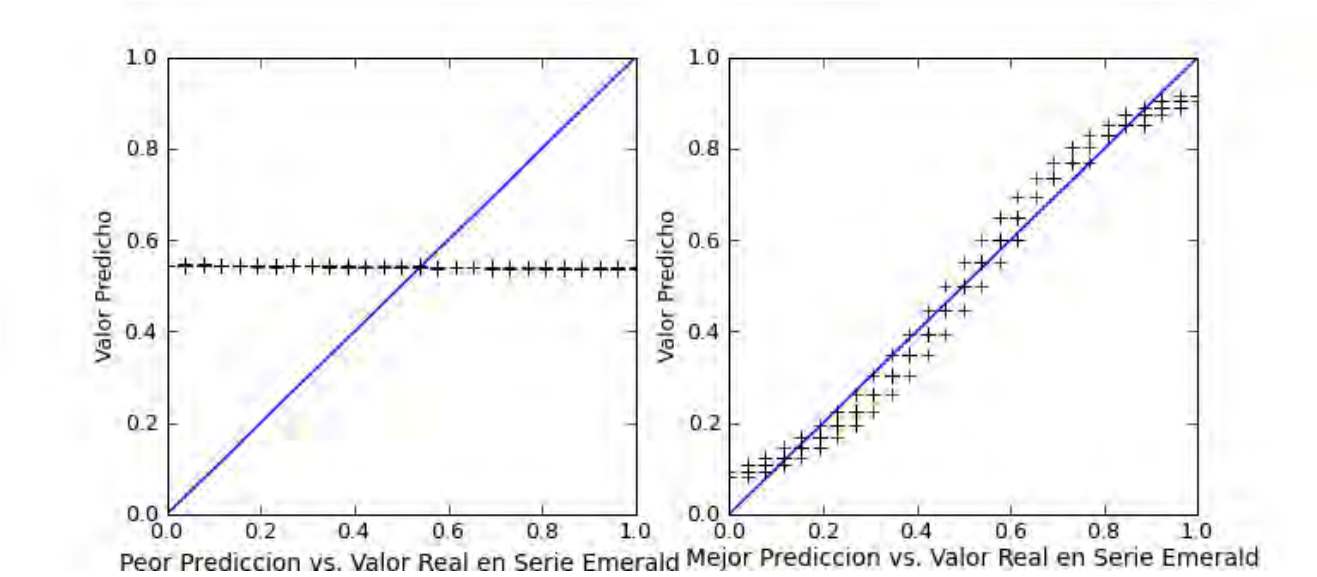

Figura C.8: Gráfico de valor real contra valor predicho por la peor y mejor predicción que obtuvo SERENA en el conjunto de prueba de la Serie Emerald. Se presenta la recta $f(x) = x$ como la línea de color azul.

C.1.3 Observaciones Generales sobre la Entonación

De acuerdo a las gráficas que muestran las mejores predicciones contra los valores reales de las series, SERENA logra encontrar estructuras de red neuronal y valores para sus pesos que permiten tener una buena idea del comportamiento de la serie, pues a pesar de no dar siempre sus valores exactos, permite observar las tendencias de la misma.

Además, el análisis de los errores de SERENA sobre ambas series nos permite determinar que la peor predicción, tanto para la Serie DJIA como para la Serie Emerald, forman una línea horizontal, indicativo de que la red no logró aprender nada sobre el comportamiento de la serie.

En el caso de la mejor predicción de cada serie, se puede observar que para SERENA fue más fácil aprender la Serie DJIA que la Serie Emerald, pues los errores de la primera no forman patrón alguno, mientras que para la segunda se muestran ondas que indican que la arquitectura no logró captar todas las características de la misma.

También se puede afirmar que la escogencia de una estructura para la red neuronal afecta significativamente los resultados de la predicción, pues depende de la cantidad de neuronas que se encuentran en la capa en entrada y en la capa oculta, la red puede ser incapaz de

aprender alguna característica de la serie o puede tener dificultades para generalizar.

Como en este experimento solamente se ejecutó SERENA 30 veces, tomando la cantidad de neuronas de entrada y ocultas un valor aleatorio, es claro que no fueron analizadas todas las estructuras de red posibles, por lo que la dificultad mostrada para predecir la Serie Emerald puede ser consecuencia que la red neuronal que presentó mejores resultados no posee la estructura óptima para el aprendizaje de la misma.

Escoger el tamaño de la red es un arte más que una ciencia, pero evaluando estructuras diferentes escogidas aleatoriamente, como hace SERENA en este experimento, es posible obtener información sobre posibles configuraciones que deban ser mejor exploradas, observando las características en común que presentan las redes con mejor desempeño.

En este experimento se pudo observar que en ambas series, las redes neuronales con un número pequeño de neuronas de entradas, en el intervalo $[1 \ldots 3]$, daban valores de MSE más bajos tanto en el conjunto de entrenamiento como en el conjunto de prueba, indicativo de que se necesitan pocos valores de la serie para determinar el siguiente. Esta observación es muy importante, pues de querer obtener una mejor predicción, se puede reducir el espacio de búsqueda a las redes neuronales que tengan una cantidad de neuronas de entrada en ese intervalo.

C.2 Experimento de Entonación#2: Estructura de la Red Neuronal en CLARE

El objetivo de este experimento es observar el comportamiento de CLARE con distintas estructuras de red. En este experimento se realizaron 5 ejecuciones de CLARE usando redes neuronales de configuración $(1, 5, 1)$; $(2, 2, 1)$; $(2, 5, 1)$; $(2, 8, 1)$; $(5, 5, 1)$; $(5, 8, 1)$ y $(7, 7, 1)$.

Tomando como base el MSE sobre el conjunto de prueba, se determina cuál las estructuras que obtienen la mejor y la peor predicción.

C.2.1 Resultados sobre Serie DJIA

De todas las estructuras evaluadas, el error más alto se obtuvo utilizando redes neuronales $(2, 2, 1)$, donde se observó un MSE de $0,85$; mientras que el error más bajo se consiguió con una configuración $(1, 5, 1)$, que presentó MSE de $7,48 \times 10^{-4}$.

Las predicciones de ambas redes sobre el conjunto de entrenamiento y el conjunto de prueba se pueden observar en la Figura C.9 y la Figura C.10, respectivamente.

Figura C.9: Conjunto de entrenamiento de la Serie DJIA, con la mejor y peor predicción dadas por CLARE

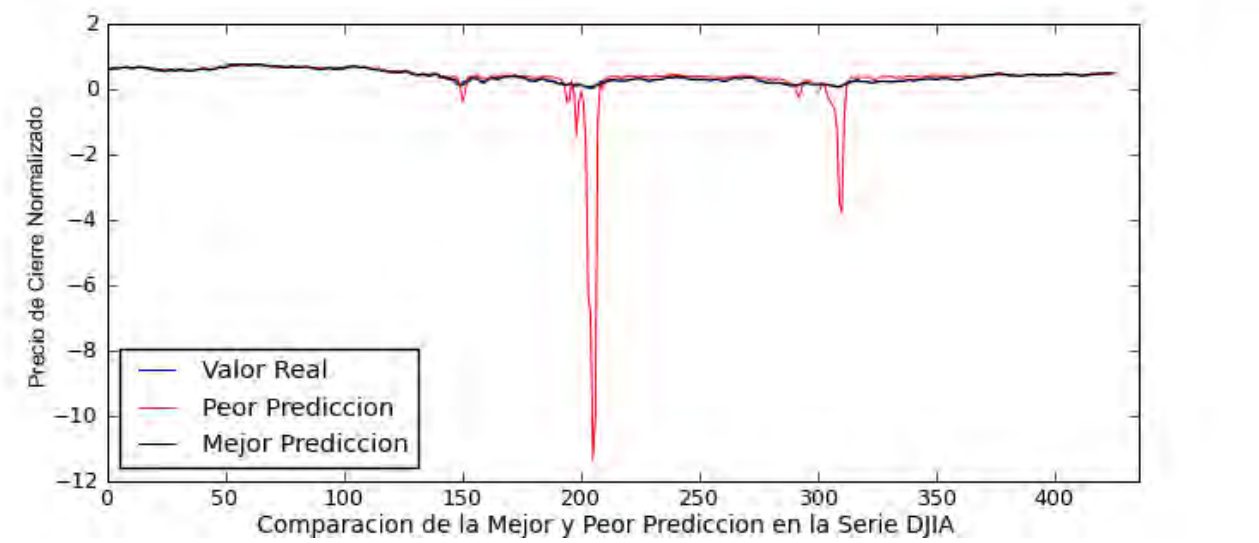

Figura C.10: Conjunto de prueba de la Serie DJIA, con la mejor y peor predicción dadas por CLARE

En la gráfica de la peor predicción no se aprecia la serie, pues los picos toman valores muy bajos, comparados con los valores originales de la misma. Sin embargo, se observa claramente que en ambos conjuntos, la peor predicción presenta picos descendientes donde se aleja mucho del valor real, pero fuera de estas zonas, se ajusta adecuadamente a la serie.

Los errores en cada punto, que se aprecian en la Figura C.11, confirman las observaciones anteriores, pues en el caso de la peor predicción, la mayoría de los errores son cercanos a 0, excepto en el punto donde se forman los picos.

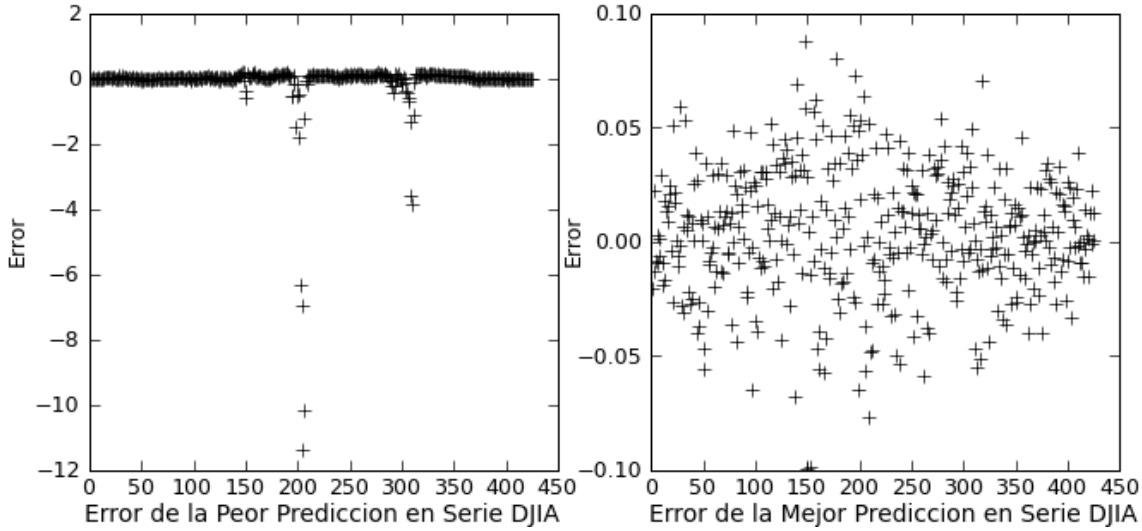

Figura C.11: Gráfico de error en cada punto de la peor y mejor predicción que obtuvo CLARE en el conjunto de prueba de la Serie DJIA.

En la Figura C.12 se presenta la recta ideal, y se puede observar que en el mejor caso, las predicciones se acercan a la misma. Una observación más interesante se obtiene de la peor predicción, pues los valores predichos son cercanos reales, excepto para aquellos menores a $0,3$, donde se aleja mucho.

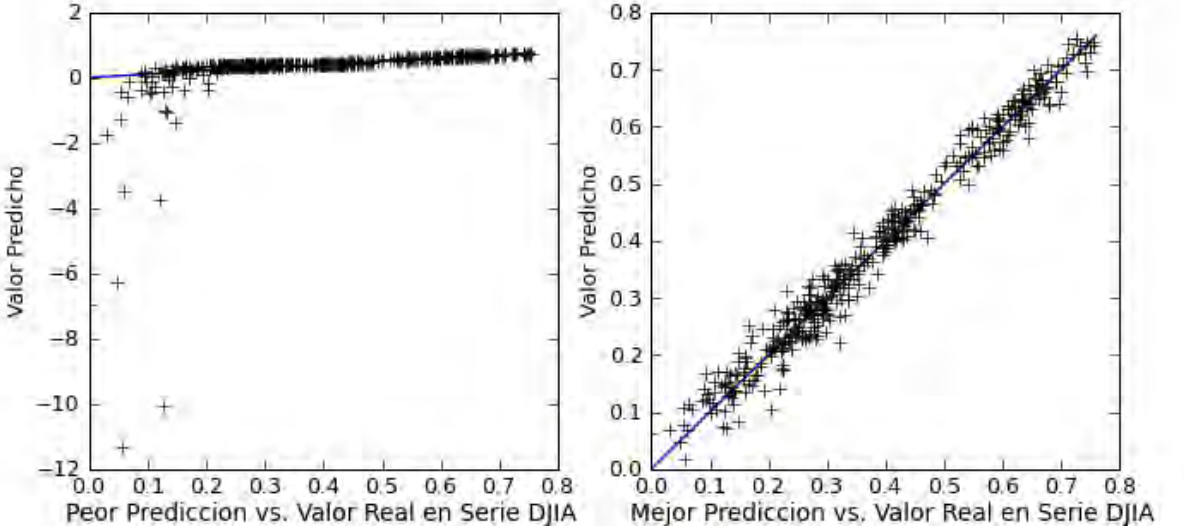

Figura C.12: Gráfico de valor real contra valor predicho por la peor y mejor predicción que obtuvo CLARE en el conjunto de prueba de la Serie DJIA.

C.2.2 Resultados sobre Serie Emerald

De todas las estructuras evaluadas, el error más alto se obtuvo utilizando redes neuronales $(2, 2, 1)$, donde se observó un MSE de $32,72$; mientras que el error más bajo se consiguió con una configuración $(2, 5, 1)$, que presentó MSE de $3,86{\times}10^{-4}$.

Las predicciones de ambas redes sobre el conjunto de entrenamiento y el conjunto de prueba se pueden observar en la Figura C.13 y la Figura C.14, respectivamente.

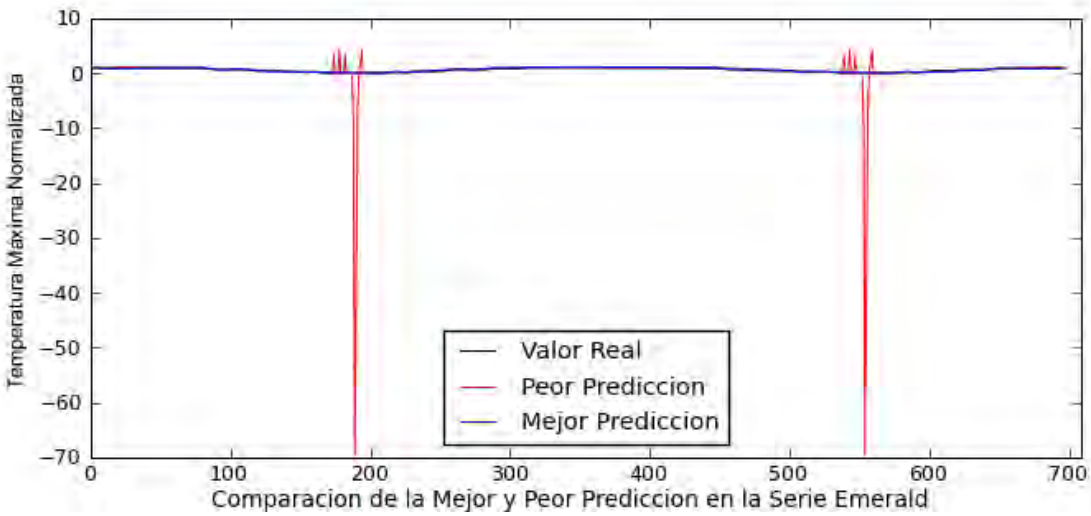

Figura C.13: Conjunto de entrenamiento de la Serie Emerald, con la mejor y peor predicción dadas por CLARE

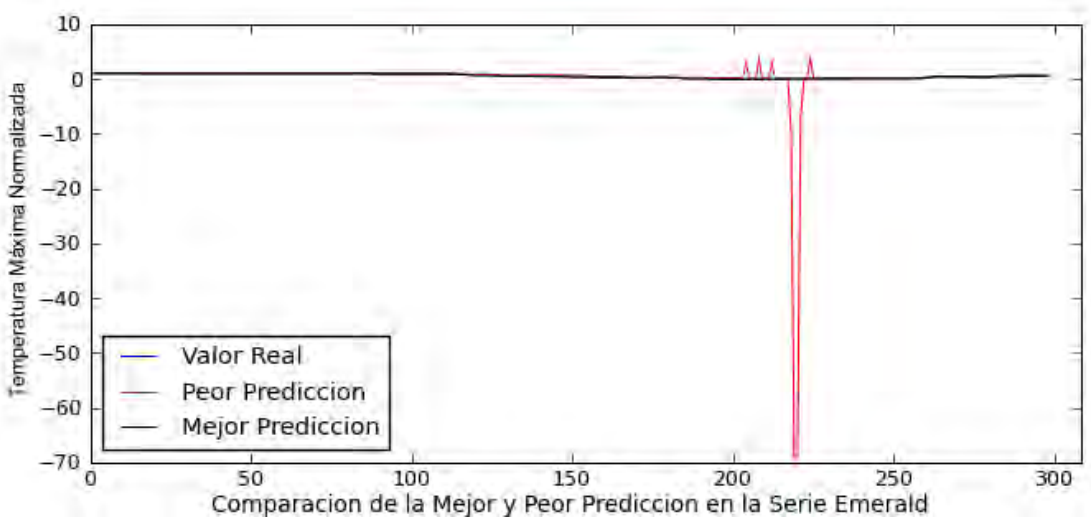

Figura C.14: Conjunto de prueba de la Serie Emerald, con la mejor y peor predicción dadas por CLARE

Se observa claramente en ambos conjuntos la presencia de picos descendientes que se alejan mucho de la serie, aunque fuera de estas áreas, tanto la peor como la mejor predicción son cercanas al valor real. Nótese además que los picos se presentan en los mismos intervalos de tiempo que existen curvas descendientes en la serie original. (Ver Apéndice B).

Los errores en cada punto, que se aprecian en la Figura C.15, muestran que la mayoría de los errores son cercanos a 0, pero se distingue claramente el error del pico, que se encuentra cercano a 70.

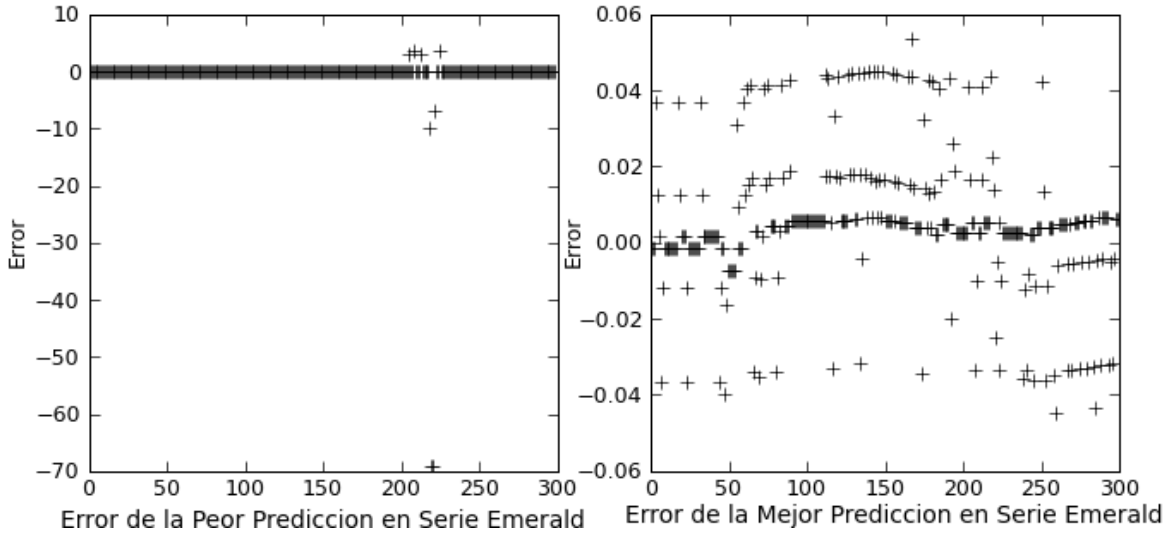

Figura C.15: Gráfico de error en cada punto de la peor y mejor predicción que obtuvo CLARE en el conjunto de prueba de la Serie Emerald.

En la Figura C.16 se presenta la recta ideal, y se puede observar que ambos casos, la mayoría de los puntos son cercanos a la misma, pero en la peor predicción se alejan en los valores negativos.

Una observación interesante de estos resultados es que el MSE de la predicción es muy alto: $32,72$, pero la gráfica de valores reales contra valores predichos muestra que no difieren mucho, a excepción de un valor particular que forma el pico. Sin embargo, al ser grande la diferencia entre el valor original de la serie y éste, se obtiene un valor alto de MSE, pues esta medida al ser cuadrática, es afectada por grandes diferencias, que la hacen subir rápidamente.

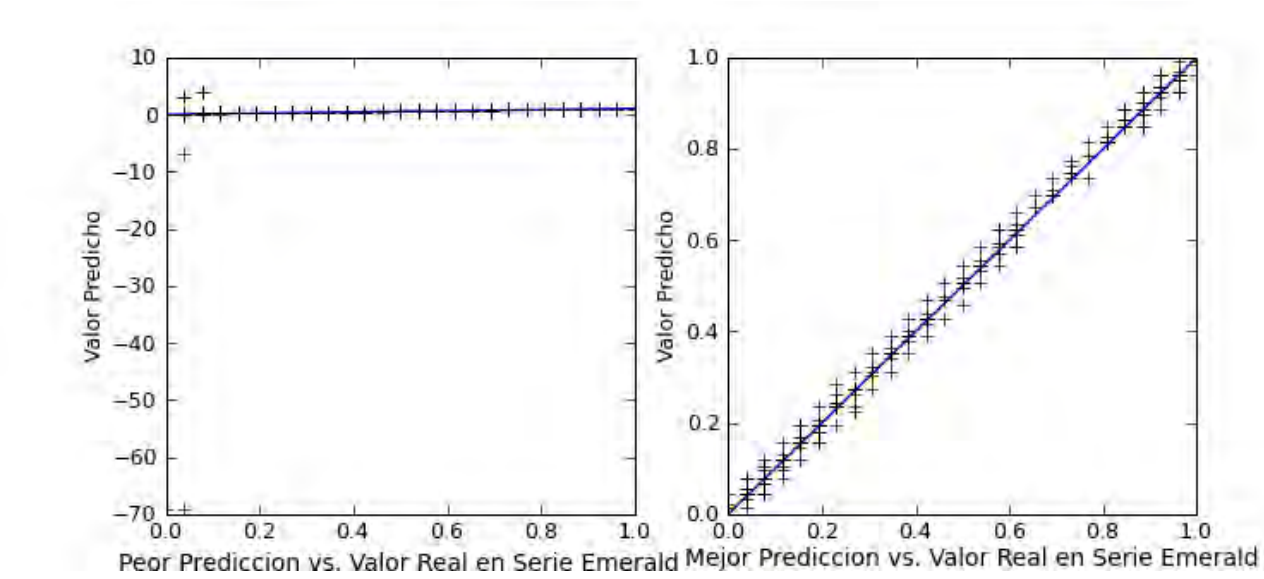

Figura C.16: Gráfico de valor real contra valor predicho por la peor y mejor predicción que obtuvo CLARE en el conjunto de prueba de la Serie Emerald.

C.2.3 Observaciones Generales sobre la Entonación

Se observa que CLARE logra obtener muy buenas predicciones sobre las dos series de prueba, cuando se escoge una estructura adecuada para la red neuronal, lo que indica que este parámetro afecta significativamente los resultados de la predicción.

También se observa que con una configuración que no sea adecuada para la serie, CLARE puede arrojar valores de MSE muy altos, como consecuencia de obtener diferencias grandes en algunos puntos. Hay que hacer notar que CLARE, aún en estos casos, logra aprender reglas útiles en algunos sub-espacios del problema.

Esta observación es importante, pues muestra que dividir el espacio del problema es una idea recomendable, pues se puede obtener información sobre algunos fragmentos de la serie. Por lo tanto, se obtiene información aun de las peores predicciones, pues los picos que se observaron muestran que para ese sub-espacio en particular hace falta una estructura de red diferente y puede indicar que existe allí una característica importante que vale la pena estudiar a fondo.

Estos resultados sugieren también que CLARE podría obtener mejores resultados si se permite que cada regla tenga una estructura de red neuronal diferente para aprender las características del sub-espacio dado por su condición.

En este experimento se pudo observar que en ambas series, las redes neuronales $(2, 2, 1)$ dieron las peores predicciones, lo que puede ser consecuencia de que posee pocos nodos en la capa oculta, que no son capaces de aprender con la complejidad de las series estudiadas.