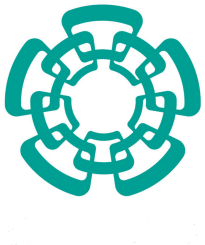


Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Cinvestav Tamaulipas

Entrenamiento Evolutivo de una Red de Función de Base Radial

Tesis que presenta:

Samuel Omar Tovias Alanis

Para obtener el grado de:

**Maestro en Ciencias
en Ingeniería y Tecnologías
Computacionales**

Director de la Tesis:
Dr. Wilfrido Gómez Flores

Cd. Victoria, Tamaulipas, México. Agosto, 2017

La tesis presentada por Samuel Omar Tovias Alanis fue aprobada por:

_____

Dr. Iván López Arévalo

Dr. César Torres Huitzil

Dr. Wilfrido Gómez Flores, Director

Cd. Victoria, Tamaulipas, México., 24 de Agosto de 2017

A mi familia

Agradecimientos

- Agradezco a Dios por permitirme cumplir este objetivo.
- A mi familia por su apoyo incondicional.
- Al Dr. Wilfrido Gómez Flores por su paciencia, apoyo y dirección a lo largo del trabajo de tesis.
- A mis revisores, Dr. Edwyn Javier Aldana Bobadilla, Dr. Iván López Arévalo y Dr. César Torres Huitzil por sus valiosos comentarios y sugerencias.
- A los profesores que con paciencia y profesionalismo me compartieron de su conocimiento en sus respectivas asignaturas.
- Al personal del CINVESTAV-Tamaulipas por su apoyo y dedicación.
- Agradezco al CINVESTAV-Tamaulipas por la oportunidad que me brindó para estudiar su posgrado y los recursos proporcionados para la realización del mismo.
- Al CONACyT por el apoyo económico con el cual pude concentrarme en mis estudios.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Publicaciones

Samuel Tovias, Wilfrido Gómez. *Automatic Construction of the Complete Architecture of a Radial Basis Function Network Using Differential Evolution*. 14º Conferencia Internacional de Ingeniería Eléctrica, Ciencias de la Computación y Control Automático (CCE 2017), Ciudad de México, México, Septiembre 2017.

Resumen

Entrenamiento Evolutivo de una Red de Función de Base Radial

por

Samuel Omar Tovias Alanis
Unidad Cinvestav Tamaulipas
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2017
Dr. Wilfrido Gómez Flores, Director

En el entrenamiento de una red de función de base radial (RBFN, por sus siglas en Inglés), la selección de características y el diseño del clasificador son dos tareas que comúnmente se tratan en procesos separados. La primera está relacionada con el número de nodos de entrada y tiene como objetivo seleccionar las características que provean la información más relevante para ayudar en la tarea de clasificación. Por otra parte, el diseño del clasificador se asocia con la capa oculta y se encarga de obtener los parámetros de las funciones de base radial (RBF, por sus siglas en Inglés) que maximicen el desempeño del clasificador. En esta tesis se propone un algoritmo de entrenamiento evolutivo para obtener una arquitectura de RBFN considerando simultáneamente a los problemas de selección de características y diseño del clasificador utilizando el algoritmo de evolución diferencial (DE, por sus siglas en Inglés). El enfoque propuesto fue comparado con un algoritmo de entrenamiento exhaustivo, el cual genera todas las posibles arquitecturas de RBFN con el objetivo de encontrar la mejor configuración de arquitectura de red. En los experimentos se utilizaron 10 conjuntos de datos del mundo real empleados en tareas de clasificación. Los resultados indican que el enfoque propuesto es competitivo y en algunos casos mejora las soluciones del algoritmo de entrenamiento exhaustivo en términos de desempeño de clasificación.

Abstract

Evolutive Training of a Radial Basis Function Network

by

Samuel Omar Tovias Alanis
Cinvestav Tamaulipas
Center for Research and Advanced Studies of the National Polytechnic Institute, 2017
Dr. Wilfrido Gómez Flores, Advisor

In the training of the radial basis function network (RBFN), feature selection and classifier design are two tasks commonly addressed as separated processes. The former is related to the number of input nodes and aims to select the features that provide most relevant information for classification task. Moreover, classifier design is associated with the hidden layer and is responsible for obtaining the parameters of the radial basis functions (RBF) that maximize classification performance. This thesis proposes an algorithm of evolutive training to get a RBFN architecture considering simultaneously feature selection and classifier design using differential evolution (DE) algorithm. The proposed approach was compared with an exhaustive algorithm, which generates all possible architectures of RBFN in order to find the best network. The experiments consider 10 real-world datasets for classification. The results suggest that the proposed approach performed similar or better than the exhaustive training algorithm in terms of classification rate.

Nomenclatura

RBFN	Radial Basis Function Network
RBF	Radial Basis Function
DE	Differential Evolution
MCC	Matthews Correlation Coefficient
MRMR	Minimum Redundancy Maximum Relevance
E	Efficiency
ANOVA	Analysis of Variance

1

Introducción

Este capítulo describe los antecedentes y la motivación del problema que se aborda en el proyecto de tesis. También, se describe formalmente el planteamiento del problema, se establece la hipótesis de investigación y se definen los objetivos que se desean cumplir. Finalmente, se muestra la metodología de investigación que se seguirá en el desarrollo del proyecto.

1.1 Antecedentes

A lo largo de la historia las personas han utilizado su capacidad de razonamiento y toma de decisiones para producir tecnología que les permita resolver problemas de la vida cotidiana. Un ejemplo de lo anterior es la construcción de diferentes máquinas y dispositivos que emulan algunos comportamientos de los seres humanos y de otros seres vivos con el fin de realizar tareas de manera más eficiente.

Un avance notable en la carrera tecnológica de la humanidad se logró con el desarrollo de la

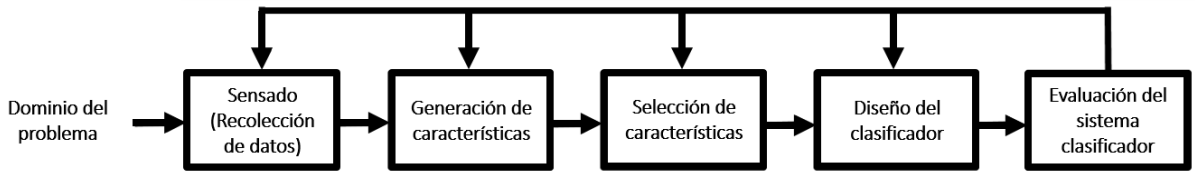


Figura 1.1: Etapas básicas implicadas en el diseño de un clasificador.

computadora digital, la cual es una máquina electrónica que puede ejecutar millones de operaciones lógicas y matemáticas en un segundo. Con el surgimiento de las ciencias computacionales comenzó a utilizarse el término de Inteligencia Artificial (IA) para referirse al conjunto de técnicas y métodos computacionales que permiten emular procesos de aprendizaje y toma de decisiones propias de los seres humanos y algunos otras especies en la naturaleza [37].

El Reconocimiento de Patrones (RP) es un área de la IA que se encarga de clasificar objetos en un determinado número de clases, en donde se abordan problemas del mundo real empleando técnicas especiales de Aprendizaje Automático (del término en Inglés *Machine Learning*), las cuales construyen sistemas de clasificación considerando una etapa de aprendizaje para adaptar sus parámetros y obtener los resultados deseados.

En el RP a los objetos que se quieren clasificar se les conoce con el término genérico de patrones y se representan matemáticamente con un vector, en el que sus elementos son una serie de medidas conocidas como características, las cuales describen a un objeto. De esta forma, un conjunto de objetos que tienen características similares pertenecen a una misma clase de patrones.

Las etapas básicas del diseño de un sistema de clasificación se muestran en el diagrama de bloques de la Figura 1.1 [42]. En la recolección de datos se obtienen las mediciones de las variables mediante sensores, los cuales pueden ser micrófonos, cámaras digitales, termómetros, entre otros. Los datos que se recolectan dependen de la aplicación y del dominio del problema. Desde el punto de vista

estadístico, es recomendable que se obtengan un número grande de muestras según el fenómeno de interés, esto con el fin de alcanzar una buena generalización del sistema de clasificación.

El objetivo de la generación de características es obtener una representación matemática de los datos extraídos por los sensores para describir a un objeto de manera cuantitativa, es decir, conseguir un vector de características invariantes y discriminantes mediante la aplicación de modelos y técnicas matemáticas.

La etapa de la selección de características se encarga de reducir el espacio de características, de forma que se seleccionen solamente aquellos atributos que proporcionen la mayor relevancia con respecto a la variable de clase y la mínima redundancia entre cada par de atributos. Los principales enfoques para la etapa de selección de características son los métodos *filter* y *wrapper* [21]. En los métodos *filter* se utilizan medidas de interdependencia estadística entre las características y las variables de clase para asignar una puntuación a cada atributo, de esta forma las características se ordenan por su importancia y algunas son aceptadas para formar parte del subconjunto de características seleccionado. Los métodos *wrapper* consideran a la etapa de selección de características como un problema de búsqueda en donde diferentes combinaciones de atributos son elegidas para ser evaluadas y de esta forma comparar su desempeño con otras combinaciones. Se emplean modelos de clasificación para evaluar las diferentes combinaciones de características y asignar puntuaciones basadas en la exactitud del modelo [14].

Un clasificador se encarga de relacionar patrones desconocidos con las clases establecidas previamente, mediante la división del espacio de características en secciones que cuenten con altas concentraciones de patrones que posean atributos semejantes. Por lo anterior, en la fase del diseño del clasificador se toman en cuenta algunas consideraciones referentes a la distribución de los datos con el objetivo de elegir el tipo de clasificador más adecuado para la tarea de reconocimiento. Dentro

de los tipos de clasificadores se encuentran los lineales, no lineales, y probabilísticos, como ejemplo de cada uno de ellos están el Análisis Lineal Discriminante (LDA, del Inglés *Linear Discriminant Analysis*) [17], las Redes Neuronales Artificiales (ANN, del Inglés *Artificial Neural Networks*) [34] y los clasificadores Bayesianos [38], respectivamente.

La etapa final en el diseño de un sistema de clasificación consiste en la evaluación del clasificador, en la cual se utilizan métodos que proporcionen una medida cuantitativa de la calidad del clasificador. En la práctica se tienen conjuntos de datos finitos; por tanto, para evaluar el clasificador se generan conjuntos de entrenamiento y prueba.

La Red de Función de Base Radial (RBFN, del Inglés *Radial Basis Function Network*) fue propuesta por Broomhead y Lowe en 1988 y cuenta con una topología fija de tres capas [5]. La capa de entrada se encarga de recibir y distribuir los datos desde el exterior. La capa oculta es activada por Funciones de Base Radial (RBF, del Inglés *Radial Basis Functions*), generando así respuestas no lineales de los valores provenientes de la capa de entrada. Los nodos de esta capa exhiben un comportamiento local, ya que éstos se activan cuando los elementos de la capa de entrada están dentro de su vecindario, es decir, son cercanos a los centros de las RBF. Finalmente, la capa de salida es activada por funciones lineales continuas, por lo que se realiza una combinación lineal de las salidas de los nodos de la capa oculta. Este tipo de clasificador no lineal tiene la ventaja de ser muy simple en cuanto a su arquitectura, a diferencia de otro tipo de redes neuronales como la Red Neuronal Prealimentada o la Red Neuronal Recurrente [3, 24] que pueden llegar a tener más de tres capas. Otra de las ventajas importantes de este tipo de red es que puede generar hiperesferas de decisión al utilizar RBF en los nodos de la capa oculta, por lo que es capaz de clasificar conjuntos de datos no linealmente separables.

En relación a la construcción de la arquitectura de la RBFN, la etapa de selección de características

está vinculada con la capa de entrada, debido a que la cantidad de características de los patrones a clasificar coincide con el número de nodos en dicha capa. El objetivo de la etapa de selección de características en el diseño de una RBFN es encontrar un subconjunto de atributos que sean relevantes en el desempeño de clasificación. Por su parte, la etapa de diseño del clasificador se relaciona con los nodos de la capa oculta, en donde es necesario determinar los parámetros de las RBF, los pesos entre la capa oculta y la capa de salida, y la cantidad de nodos en la capa oculta.

La etapa de diseño del clasificador en una RBFN se desarrolla en dos procesos independientes mediante un algoritmo de aprendizaje híbrido:

1. *Aprendizaje no supervisado:* Los centros de las RBF se determinan a partir de un algoritmo de agrupamiento y posteriormente se calculan los radios con base en los centros obtenidos. El número de nodos ocultos determina la cantidad de grupos que se desean obtener con el algoritmo de agrupamiento.

2. *Aprendizaje supervisado:* Dadas las respuestas de la capa oculta, los pesos se ajustan mediante un algoritmo supervisado que realiza el mapeo entrada-salida de los nodos de salida. La cantidad de nodos en la capa de salida corresponde al número de clases.

1.2 Motivación

En el diseño de la RBFN no existe una metodología estándar y definida para encontrar sus parámetros de tal forma que se maximice el desempeño de clasificación y se minimice la complejidad de la arquitectura de la red de forma simultánea. Además, son pocos los trabajos publicados que abordan los problemas de selección de características y diseño del clasificador al mismo tiempo [2, 6, 19, 27, 29]. Por lo tanto se plantea trabajar con ambos problemas simultáneamente con el objetivo de maximizar el desempeño de clasificación y minimizar la complejidad de la arquitectura de

la RBFN utilizando un algoritmo de cómputo evolutivo conocido como Evolución Diferencial (DE, del Inglés *Differential Evolution*).

El cómputo evolutivo se considera, junto con las redes neuronales artificiales, una de las principales áreas de investigación y aplicación de la IA [44]. Para el caso del entrenamiento de la RBFN, los distintos individuos que forman parte del proceso evolutivo contienen información de los parámetros de la arquitectura de la RBFN, pudiendo incluir el conjunto de características con el que se entrenará el clasificador.

El algoritmo DE, a diferencia de los algoritmos evolutivos tradicionales, utiliza un esquema de selección voraz, el cual hace una selección óptima de forma local en cada paso del proceso con la esperanza de que se encuentre un óptimo global [40]. Se ha demostrado que el algoritmo de evolución diferencial y sus variantes tienen un buen desempeño relativo a otros métodos de optimización para la resolución de problemas de optimización global, tanto en problemas que consideran funciones separables y unimodales, como en problemas con funciones multimodales y no separables [26]. Además, DE se ha comparado con otras técnicas de optimización global tales como la Optimización por Enjambre de Partículas (PSO, del Inglés *Particle Swarm Optimization*) y con Algoritmos Evolutivos Simples (SAE, del Inglés *Simple Evolutionary Algorithm*). En general el algoritmo DE ha demostrado tener un mejor rendimiento en diferentes tipos de problemas de optimización que los otros algoritmos mencionados, tal como lo describen Vesterstrom y Thomsen en su estudio [43], quienes además mostraron que DE es un algoritmo robusto y consistente.

Por lo anterior, en este trabajo de tesis, se implementará el algoritmo DE para el entrenamiento de la arquitectura de la RBFN considerando simultáneamente las etapas de selección de características y diseño del clasificador.

1.3 Planteamiento del problema

El principal objetivo del RP es clasificar objetos en un conjunto de categorías o clases. Cada objeto o patrón se describe de forma cuantitativa por medio de un vector D-dimensional denotado como $\mathbf{x} = [x_1, x_2, ..., x_D]^T$, donde D es la cantidad de características o atributos que posee el patrón $\mathbf{x}$. Un grupo de patrones que comparten características similares componen una clase ω_k dentro del conjunto $\Omega = \{\omega_1, \omega_2, ..., \omega_c\}$ con c clases.

La tarea de clasificación se puede plantear de manera formal como sigue: dado un conjunto de objetos de entrada conformado por N muestras de la forma $X = \{(\mathbf{x}_i, y_i)|i = 1, 2, ..., N\}$, donde $\mathbf{x}_i \in \mathbb{R}^D, y_i \in \Omega$ y x_{ij} es el valor del i-ésimo patrón correspondiente a su j-ésima característica, se busca obtener una función $f : \mathbb{R}^D \rightarrow \Omega$ que dado como entrada un patrón desconocido $\mathbf{x}$, proporcione una salida en Ω, la cual es una aproximación a la clase a la que pertenece.

En ocasiones, el vector de características posee un número muy grande de atributos y algunos de ellos no aportan información relevante, por lo que el desempeño de clasificación disminuye. Además, para conjuntos de datos con patrones de alta dimensionalidad es computacionalmente costoso, en tiempo y espacio, realizar una búsqueda exhaustiva en todo el espacio de características para encontrar un subconjunto de éstas que permitan obtener el mejor desempeño de clasificación. Por tanto, es necesario emplear técnicas de selección de características para elegir un subconjunto adecuado de atributos considerando únicamente aquellos que proporcionen información discriminante en relación a la tarea de clasificación.

Formalmente, el problema de selección de características se define de la siguiente manera: dado un conjunto de datos de entrada en el espacio $\mathbb{R}$ organizado en una matriz de N muestras por D características, denotada como X, y una variable de clase o salida deseada, en el conjunto Ω para

cada muestra, se debe encontrar a partir de un espacio D-dimensional, $\mathbb{R}^D$, un subespacio de d características, $\mathbb{R}^d$ que caracterice adecuadamente a las clases en Ω, donde $d < D$.

La RBFN es un clasificador que tiene la capacidad de etiquetar conjuntos de datos no linealmente separables. Con respecto a la arquitectura de la red, el problema de selección de características radica en encontrar el subconjunto de atributos que sean relevantes en el desempeño de clasificación. Por su parte, el problema de diseño del clasificador consiste en encontrar los valores de los parámetros de los centros y radios de las funciones de base radial, los valores de los pesos entre la capa oculta y la capa de salida, así como también la cantidad total de nodos que conforman la estructura de la capa oculta. Ambos problemas se pueden plantear como uno de optimización bajo el esquema del algoritmo DE, en donde se pretende maximizar el desempeño de clasificación, minimizar la complejidad de la arquitectura de la RBFN (capa oculta) y seleccionar el subconjunto de características más relevantes para la tarea de clasificación.

1.4 Hipótesis

La hipótesis planteada para el desarrollo del proyecto de tesis es la siguiente:

Es posible entrenar la arquitectura de una red de función de base radial mediante un algoritmo evolutivo que de manera simultánea trate con el problema de la selección de características y el diseño del clasificador para maximizar el desempeño de clasificación y minimizar la complejidad de la arquitectura de la red.

1.5 Objetivos

1.5.1 Objetivo general

Obtener un algoritmo de entrenamiento de una red de función de base radial empleando evolución diferencial considerando los problemas de selección de características y diseño del clasificador de manera simultánea.

1.5.2 Objetivos específicos

1. Definir un esquema de representación del individuo que codifique la estructura de la red.
2. Establecer una función objetivo que evalúe el desempeño de clasificación y la complejidad de la arquitectura de la red de manera simultánea.
3. Seleccionar la variante del algoritmo de evolución diferencial que se utilizará en el algoritmo propuesto.

1.6 Metodología de investigación

En esta sección se describen las actividades a realizar para corroborar la hipótesis de investigación.

1. **Revisión de la literatura:**
   - Investigar los trabajos en el estado del arte relacionados al entrenamiento evolutivo de la RBFN utilizando evolución diferencial, identificando las características más importantes de los algortimos propuestos.
   - Investigar y seleccionar adecuadamente los diferentes casos de prueba que se utilizarán para la etapa de experimentación.

- Investigar métricas estadísticas de evaluación para medir el desempeño de clasificación y la complejidad de la arquitectura de la RBFN.

2. **Diseño del algoritmo propuesto:**

   - Establecer un esquema de representación del individuo para el proceso de entrenamiento evolutivo considerando una codificación real.
   - Diseñar una función objetivo para evaluar el desempeño de clasificación y la complejidad de la arquitectura de la RBFN.
   - Seleccionar las variantes del algoritmo de evolución diferencial que se utilizarán en la experimentación.

3. **Experimentación y resultados:**

   - Diseñar e implementar un algoritmo para construir y entrenar una RBFN utilizando un enfoque exhaustivo.
   - Comparar los resultados del entrenamiento evolutivo y el entrenamiento exhaustivo.
   - Seleccionar la variante del algoritmo de evolución diferencial.
   - Detectar ventajas y desventajas del entrenamiento evolutivo con respecto al entrenamiento exhaustivo.
   - Realizar un análisis de convergencia para determinar si es posible reducir el número de generaciones del algoritmo de entrenamiento evolutivo.

4. **Conclusiones y publicación de resultados:**

   - Validar la hipótesis de investigación.
   - Mencionar las conclusiones y contribuciones del proyecto de tesis.
   - Definir el trabajo futuro.

- Publicar los resultados obtenidos en un artículo de congreso detallando la metodología propuesta.

1.7 Organización del trabajo de tesis

El contenido de este documento de tesis se complementa con los cinco capítulos siguientes que a continuación se describen. El Capítulo 2 detalla el estado del arte relacionado a los trabajos recientes en donde se aborda la construcción de la arquitectura de RBFN utilizando evolución diferencial y otros algoritmos inspirados en la naturaleza. En el Capítulo 3 se describen las bases teóricas de los diferentes métodos y técnicas empleados en el desarrollo de este trabajo de investigación. La metodología propuesta para el entrenamiento evolutivo de una RBFN utilizando evolución diferencial es descrita en el Capítulo 4. En el Capítulo 5 se detalla la experimentación y se muestran los resultados obtenidos con el método propuesto. Finalmente, en el Capítulo 6, se exponen las conclusiones de la investigación y se plantea el trabajo futuro.

2

Estado del arte

2.1 Introducción

En el diseño de sistemas de clasificación existen dos etapas que normalmente se estudian de forma independiente en la literatura, estas son la selección de características y el diseño del clasificador. La red de función de base radial (RBFN) relaciona ambas fases directamente por causa de su arquitectura de red; la selección de características se vincula con la capa de entrada, mientras que el diseño del clasificador se relaciona con la capa oculta.

El objetivo de la selección de características es obtener un subconjunto de atributos relevantes para la tarea de clasificación. Por su parte, la etapa de diseño del clasificador consiste en la construcción de fronteras de decisión, las cuales dividen al espacio de características para separar los patrones de las diferentes clases.

En la literatura publicada son pocos los trabajos que abordan ambos problemas simultáneamente. En seguida se revisan los artículos más relevantes relacionados a la construcción de la arquitectura de la RBFN empleando algoritmos bioinspirados.

2.2 Enfoques orientados al diseño de la capa oculta

La mayoría de los trabajos publicados en esta área de investigación tratan solamente con el diseño de la capa oculta. El enfoque principal consiste en encontrar la cantidad de nodos ocultos que maximice el desempeño de clasificación; sin embargo, también hay propuestas que consideran métricas para cuantificar la complejidad de la arquitectura de red en relación a la cantidad de nodos ocultos. Es importante señalar que en algunos no se considera un límite máximo para la cantidad de unidades ocultas lo cual se considera una desventaja, ya que la complejidad de la arquitectura de la red puede incrementar de forma considerable al elegir un número grande de nodos en la capa intermedia. Otro aspecto que se debe indicar es que en estos trabajos no se emplea ningún índice para medir la uniformidad de la distribución de los centros de las funciones de base radial en el espacio de características, lo cual también implica un inconveniente si se desea un modelo de arquitectura de RBFN que maximice el desempeño de clasificación. Esto es debido a que una ubicación adecuada de este parámetro permite obtener mejores valores de los pesos de la red, por tanto, un mejor modelo de RBFN. En algunas de las propuestas los pesos de la red forman parte de la representación de la solución, aunque en otras se opta por calcular estos parámetros empleando el método de mínimos cuadrados mediante la matriz pseudoinversa. En la Tabla 2.1 se muestra un resumen de los trabajos que a continuación se describen.

Dhahri y Alimi [8] emplean un algoritmo de evolución diferencial modificado que utiliza un operador de mutación basado en optimización por enjambre de partículas [18]. En el entrenamiento evolutivo no se considera la búsqueda del número óptimo de nodos ocultos, puesto que este dato se

establece como un parámetro de configuración inicial. Se emplea la raíz cuadrada del error cuadrático medio (RMSE, del Inglés *Root Mean Square Error*), como métrica principal en la función objetivo para cuantificar la taza de error. Muestran el desempeño de su algoritmo en la predicción de series de tiempo de tres diferentes sistemas dinámicos con comportamientos caóticos. Es importante señalar que en la función objetivo no se considera un criterio relacionado a la complejidad de la arquitectura de la red. Además de lo anterior, la cantidad de nodos en la capa oculta se establece como un parámetro fijo, impidiendo así comparar resultados obtenidos con diferente número de nodos ocultos.

Yu y He [46] mencionan que determinar todos los parámetros de la arquitectura de la red no es una tarea trivial y representa una motivación para implementar estrategias de cómputo evolutivo. Sin embargo, si se codifican todos los parámetros, la longitud del individuo puede ser muy grande y consecuentemente también el espacio de búsqueda, con una tasa de convergencia muy lenta. Emplean un algoritmo de evolución diferencial para optimizar la cantidad de nodos ocultos y los valores de sus centros. La longitud del vector de codificación del individuo es fija, y se divide en dos partes que se concatenan. La primera consiste en una sección de variables de bandera que indican la presencia o ausencia de nodos ocultos. En la segunda sección los centros se codifican con valores reales. Se emplea el error cuadrático medio (MSE, del Inglés *Mean Square Error*) y un factor que mide la complejidad de la red, como métricas principales en el diseño de la función objetivo. Consideran que una buena configuración de la RBFN debe tener la mínima cantidad de nodos ocultos maximizando la exactitud de clasificación. Cabe señalar que no se menciona la cantidad máxima de nodos ocultos considerados por cada experimento; además no consideran una medida para cuantificar la correcta distribución de los centros en el espacio de características.

Yu y Zhu [47] desarrollan un método de entrenamiento de una RBFN que combina un algoritmo genético de multi-codificación (MGA, del Inglés *Multi-Encoding Genetic Algorithm*) con un algoritmo de retro-propagación (BP, del Inglés *Backpropagation*) para conformar un aprendizaje híbrido

denominado MGA-BP. Este algoritmo es implementado en la identificación de sistemas no lineales. Cada individuo contiene la información codificada de la cantidad de nodos ocultos, los valores de los centros y los radios de las funciones de base radial, y los pesos entre la capa oculta y la capa de salida. El vector de codificación se divide en dos partes. La primera sección corresponde a una codificación binaria, en donde se emplea una regla de desición para mantener o descartar nodos ocultos. En la parte real se codifican los valores de los parámetros de las funciones de base radial (correspondientes a los centros que no fueron descartados) y los valores de los pesos. En la función objetivo se considera un parámetro que mide la complejidad de la red, penalizando a los individuos que tengan una cantidad alta de nodos ocultos, el cual se combina con el MSE, que es empleado para medir el error de predicción de la red. Es importante remarcar que en la función objetivo solamente consideran la cantidad de nodos ocultos, sin tomar en cuenta algún factor que pueda cuantificar el efecto que tienen los valores de los parámetros de las funciones de base radial, esto es, la ubicación de los centros y el tamaño de los radios.

Zhou et al. [51] comparan un algoritmo de evolución diferencial con un método de optimización por descenso de gradiente para el entrenamiento de una RBFN. El algoritmo es implementado en la predicción de los valores de presión de mantos acuíferos, los cuales muestran características complejas no lineales. Mencionan que al emplear el método de descenso de gradiente se obtienen resultados de óptimos locales. Por lo anterior, se utiliza el algoritmo de evolución diferencial para entrenar la red y obtener los parámetros de los centros y radios de las funciones de base radial, así como los valores de los pesos. La codificación del individuo se hace con valores reales. Con el fin de mejorar la diversidad de la población y la capacidad de escapar de óptimos locales se emplea un factor de probabilidad de cruza auto-adaptativa. En la función objetivo sólo consideran el error de predicción y no toman en cuenta la complejidad de la arquitectura de la red.

Zhang et al. [50] emplean un algoritmo genético para obtener los parámetros de la arquitectura

de una RBFN y evalúan su desempeño en predicciones de series de tiempo. Utilizan una codificación mixta en la representación de la solución, únicamente se considera la cantidad de nodos en la capa oculta y el valor de los centros de las funciones de base radial, codificados con valores enteros y reales, respectivamente. El cálculo de los pesos se realiza empleando el método de mínimos cuadrados mediante la matriz pseudoinversa, ya que este método proporciona una solución rápida y directa. Emplean RMSE como criterio principal en la función objetivo para medir la taza de error de predicción. En este trabajo no se considera un criterio especial para cuantificar la complejidad de la arquitectura de red.

Huang et al. [16] proponen un algoritmo de evolución diferencial mejorado para entrenar la arquitectura de una RBFN. Este algoritmo se emplea en la industria eléctrica para la restauración rápida de sistemas de distribución. En la fase de creación de los datos de entrenamiento se emplea un método de inferencia difusa basado en heurística (HBFI, del Inglés *Heuristic-Based Fuzzy Inference*) para construir los planes de restauración bajo diferentes niveles de carga. Posteriormente la solución de arquitectura de red obtenida por el algoritmo de evolución diferencial se utiliza para la etapa de restauración. En la representación de la solución se codifican los parámetros de las funciones de base radial y los valores de los pesos. La única métrica que se considera en el diseño de la función objetivo es el MSE. Cabe señalar que en este trabajo no se considera un criterio para medir la complejidad de la arquitectura de la red y no proponen una restricción en cuanto al máximo número de nodos ocultos que se pueden considerar.

Xue et al. [45] desarrollan un método para la identificación de sistemas no lineales. Emplean un algoritmo de evolución diferencial con parámetros adaptativos para la construcción de la arquitectura de una RBFN. En la representación de la solución se codifican la cantidad de nodos ocultos, los parámetros de las funciones de base radial, y los valores de los pesos. La función objetivo calcula la inversa de la sumatoria de los cuadrados de los errores de clasificación. En este trabajo no se

menciona el número máximo de nodos ocultos que puede elegir el algoritmo, tampoco se considera una métrica de evaluación para medir la complejidad de la arquitectura de la red.

Ding et al. [11] proponen un método para la construcción de la arquitectura de una RBFN. Emplean un algoritmo genético y utilizan una codificación híbrida para la representación de la solución. En la sección correspondiente a la codificación de los nodos ocultos implementan una representación binaria. Se hace uso de una regla de desición simple para seleccionar o descartar los nodos de la capa oculta que consiste en seleccionar un nodo si este se inicializa con el valor de "1", mientras que si tiene el valor de "0" significa que el nodo se descarta. Por otra parte, se emplea una representación real para codificar el conjunto de pesos de la red. Los parámetros de las funciones de base radial no se codifican en la representación de la solución. Los centros se obtienen con el algoritmo de agrupamiento $k-$means y los radios se calculan mediante una heurística. Es importante señalar que no se menciona la cantidad máxima de nodos ocultos que puede seleccionar el algoritmo. En la función objetivo se considera el error de entrenamiento y el número de nodos ocultos, de esta forma se toman en cuenta criterios que evalúan tanto el desempeño de clasificación como la complejidad de la arquitectura de la red. Sin embargo, en este enfoque propuesto no se considera una métrica de evaluación para medir la correcta distribución de los centros de las funciones de base radial en el espacio de características. También es importante señalar que al no considerarse a los centros como parámetros codificados en el individuo, se limita a la arquitectura de red a no poder mejorar la distribución de los centros una vez inicializados con el algoritmo de agrupamiento.

Bajer et al. [1] diseñan un algoritmo para el entrenamiento de una RBFN empleando evolución diferencial. El enfoque principal es la construcción de la capa oculta, y solamente codifican en la representación de la solución a los centros y radios de las funciones de base radial. Los pesos de la red son calculados analíticamente con el método de mínimos cuadrados utilizando la matriz pseudoinversa. El único criterio considerado en la función objetivo es el desempeño de clasificación,

y la métrica que emplean es el MSE. Es importante considerar que no se menciona la cantidad de nodos ocultos utilizados en el entrenamiento; sin embargo, explican que este dato no se considera por cuestiones de simplicidad, puesto que el objetivo es mostrar el desempeño del algoritmo de evolución diferencial en la creación del modelo de la red.

Año	Título	Referencia	Enfoque	Métrica	Método de evaluación	Aplicación
2006	The modified differential evolution and the rbf (mde-rbf) neural network for time series prediction.	[8]	Evolución diferencial	RMSE	Media de 10 ejecuciones	Predicción de series de tiempo
2006	Training radial basis function networks with differential evolution.	[46]	Evolución diferencial	MSE y cantidad de nodos ocultos	Media de 20 ejecuciones	Clasificación
2009	A hybrid mga-bp algorithm for rbfns self-generate.	[47]	Algoritmo genético	Error de clasificación y cantidad de nodos ocultos	Media de 10 ejecuciones	Identificación de sistemas no lineales
2010	Rbf neural network using improved differential evolution for groundwater table prediction.	[51]	Evolución diferencial	Error de predicción	72 muestras para entrenamiento y 12 para verificación	Predicción de series de tiempo
2010	The optimization of radial basis function network based on chaos immune genetic algorithm.	[50]	Algoritmo genético	RMSE	Media de 50 ejecuciones	Predicción de series de tiempo
2011	Evolving radial basic function neural network for fast restoration of distribution systems.	[16]	Evolución diferencial	MSE	Media de 100 ejecuciones	Clasificación
2012	Nonlinear system identification with modified differential evolution and rbf networks.	[45]	Evolución diferencial	Error de clasificación	No especificado	Identificación de sistemas no lineales
2012	An optimizing method of rbf neural network based on genetic algorithm.	[11]	Algoritmo genético	Error de clasificación y cantidad de nodos ocultos	Datos de entrenamiento y prueba divididos en una razón de porcentaje 70/30	Clasificación
2016	Effectiveness of differential evolution in training radial basis function networks for classification.	[1]	Evolución diferencial	MSE	*Hold out* dividiendo los datos de entrenamiento, validación y prueba en una razón de porcentaje de 50/25/25	Clasificación

Tabla 2.1: Principales avances en la construcción de la arquitectura de la RBFN empleando algoritmos bioinspirados considerando los enfoques orientados al diseño de la capa oculta.

2.3 Enfoques orientados al diseño de las capas de entrada y oculta

Son pocos los trabajos publicados que consideran simultáneamente el problema de la selección de características y el diseño del clasificador. Las metaheurísticas que se emplean están generalmente basadas en cómputo evolutivo. Estos algoritmos codifican en la representación de la solución diferentes subconjuntos de atributos de forma binaria, de esta manera se descartan las características que tengan valores iguales a "0" y se mantienen aquellas que tengan un valor igual a "1". En relación al diseño de la capa itermedia, en la mayoría de los enfoques, la cantidad de nodos ocultos se representa con un entero positivo y los parámetros de las funciones de base radial se codifican con valores reales. Con respecto a los trabajos que emplean evolución diferencial, la representación de la solución consiste de valores reales, y en uno de los enfoques propuestos los indivudos que forman parte del proceso evolutivo son de longitud variable. En la mayoría de estas propuestas se agrega un factor de penalización para descartar las soluciones que tengan una cantidad grande de nodos ocultos y un alto porcentaje de características seleccionadas. En la Tabla 2.2 se muestran en forma resumida los trabajos que se describen a continuación.

Klimek y Sick [19] emplean un algoritmo evolutivo (EA, del Inglés *Evolutionary Algorithm*), con la finalidad de entrenar los parámetros de la arquitectura de la RBFN, y evaluar su rendimiento en tareas de clasificación. La representación de la solución se codifica de forma mixta. El vector de características es binario, en el cual cada bit indica la presencia o ausencia de un determinado atributo, el número de nodos ocultos es representado por un entero, cada centro es descrito por un vector de valores reales, y los radios se codifican como escalares reales positivos. Los inconvenientes notables de esta metodología es que no aplica restricciones en el número máximo de nodos en la capa oculta, y no se define un criterio de paro, por lo cual la ejecución del algoritmo evolutivo debe

ser detenida por el usuario.

Buchtala et al. [6] proponen un enfoque en donde emplean un algoritmo evolutivo para el entrenamiento de la RBFN. En la representación de la solución se utiliza una codificación mixta. La sección de características se representa de forma binaria, el número de nodos en la capa oculta emplea codificación entera y el tiempo de entrenamiento en segundos es representado con codificación real. Cabe destacar que en este enfoque no se consideran para el entrenamiento evolutivo los pesos de la red, ni los valores de los centros y radios de las funciones de base radial. Al principio de la ejecución, los centros se calculan mediante el algoritmo de agrupamiento $k-$means. Los pesos de la red se computan con descomposición QR y se recalculan en cada iteración por el método del gradiente conjugado (SCG, del Inglés *Scaled Conjugated Gradient*). En la función objetivo únicamente se considera el criterio de desempeño de clasificación. No se menciona explícitamente una métrica para medir la relevancia del subconjunto de atributos seleccionados; sin embargo, se consideran factores de penalización que afectan el valor de desempeño de las soluciones con un mayor número de características seleccionadas.

Bauer et al. [2] utilizan un algoritmo evolutivo para entrenar la arquitectura de la RBFN y realizar selección de características. Emplean una codificación mixta, donde el conjunto de posibles atributos es representado por un vector de bits y el número de nodos en la capa oculta es codificado con un número entero positivo; sin embargo, no se define un límite máximo de nodos ocultos. No se incluye en el entrenamiento evolutivo a los parámetros de los centros y los radios de las funciones de base radial. La evaluación de la red se utiliza empleando la taza de aciertos del clasificador. En relación a la función objetivo, se menciona que se puede agregar de manera opcional un factor de penalización para el número de características seleccionadas y la cantidad de nodos ocultos.

O'Hora et al. [29] implementan un algoritmo de evolución diferencial para encontrar la

arquitectura adecuada de una RBFN que maximice el desempeño del clasificador. En la representación de la solución se emplea una codificación real. Los parámetros que se consideran en el individuo para ser parte del proceso evolutivo son las características, los centros y radios de las funciones de base radial, los pesos de la red, y el punto de corte que convierte la salida continua de la red en etiquetas de clase. En este trabajo se implementa una interfaz gráfica que permite al usuario seleccionar un amplio número de parámetros que configuran el proceso de construcción de la arquitectura de la red, entre los cuales están la posibilidad de entrenar la red sin considerar la selección de características, la forma de selección, número de vectores diferentes, tipo de cruza, entre otros factores. Es importante señalar que la cantidad de nodos en la capa oculta no es considerada como un parámetro que forma parte del proceso evolutivo.

Montero y Gómez [27] tratan con los problemas de selección de características y diseño del clasificador en la construcción de la arquitectura de la red de función de base radial. Mencionan que los dos aspectos más importantes en el diseño de la red son encontrar la ubicación óptima de los centros de las funciones de base radial y seleccionar un subconjunto de características discriminantes que ayuden en la tarea de clasificación. Proponen la integración de dos algoritmos que emplean evolución diferencial, el primero llamado FSDE (del Inglés *Feature Selection with Differential Evolution*), el cual evoluciona una población de individuos codificando un subconjunto de características con el objetivo de construir la capa de entrada de la red. Durante cada iteración, el segundo algoritmo llamado CCDE (del Inglés *Classifier Design with Differential Evolution*), realiza un proceso evolutivo por cada individuo con el propósito de construir la capa oculta de la red. En consecuencia, en cada iteración del algoritmo FSDE, se obtiene una población de redes de función de base radial. Las desventajas observadas en este trabajo son que la metodología propuesta es muy compleja y computacionalmente costosa, ya que se necesita de la implementación de dos algoritmos especiales, en donde se utiliza una codificación de longitud variable para representar a los individuos de ambos procesos evolutivos.

Año	Título	Referencia	Enfoque	Métrica	Método de evaluación	Aplicación
2003	Architecture optimization of radial basis function networks with a combination of hard- and soft-computing techniques.	[19]	Algoritmo evolutivo	Lift factor	Validación cruzada de $5-$dobleces	Clasificación
2005	Evolutionary optimization of radial basis function classifiers for data mining applications,	[6]	Algoritmo evolutivo	Lift factor y error de clasificación	Validación cruzada de $n-$dobleces	Clasificación
2005	Process identification and quality control with evolutionary optimized rbf classifiers.	[2]	Algoritmo evolutivo	Error de clasificación	Validación cruzada de $10-$dobleces	Clasificación
2006	Designing radial basis function networks for classification using differential evolution.	[29]	Evolución diferencial	Error de clasificación	Validación cruzada de $5-$dobleces	Clasificación
2014	Evolutionary approach for construction of the rbf network architecture.	[27]	Evolución diferencial	Área bajo la curva ROC, cantidad de nodos ocultos, porcentaje de atributos seleccionados	bootstrap.632+	Clasificación

Tabla 2.2: Principales avances en la construcción de la arquitectura de la RBFN empleando algoritmos bioinspirados considerando los enfoques orientados al diseño de las capas de entrada y oculta.

2.4 Sumario

En la discución mostrada se observa que son pocos los trabajos publicados en la literatura que tratan simultáneamente con los problemas de selección de características y diseño del clasificador; y aún son menos los que emplean el algoritmo de evolución diferencial para hacerlo.

En relación a los trabajos que abordan únicamente el problema de diseño del clasificador, se observa que en general no consideran un criterio para medir la complejidad de la arquitectura de la red, sino que solamente emplean una métrica para evaluar el desempeño de clasificación, la cual en muchos casos es el MSE.

Con respecto a los trabajos publicados que consideran ambos problemas de forma simultánea, en general, en la función objetivo no se emplean criterios para medir la complejidad de la arquitectura de la red, ni tampoco para medir la relevancia de las características seleccionadas, en algunos casos solamente se utilizan factores de penalización para reducir el número de características.

Por lo expuesto anteriormente, se considera que existe un área de oportunidad para tratar con ambos problemas en la construcción de la arquitectura de la red de función de base radial, empleando evolución diferencial, y considerando en la función objetivo métricas que puedan evaluar la relevancia de las características seleccionadas, la complejidad de la arquitectura de red, y el desempeño de clasificación.

3

Marco teórico

En el presente capítulo se explican las técnicas empleadas en este trabajo de tesis divididas en cuatro secciones: *red de función de base radial*, *medidas de dependencia de características*, *técnicas coadyuvantes a la clasificación* y *algoritmos de evolución diferencial*. En la primera parte se explican las particularidades y generalidades de una RBFN; en la segunda sección se describen las métricas utilizadas relacionadas al problema de la selección de características; en la tercera parte se realiza una reseña de las técnicas empleadas en el manejo de los datos y el análisis de resultados, en relación a la tarea de clasificación. Por último, se describe el funcionamiento del algoritmo de evolución diferencial clásico y sus variantes más importantes.

3.1 Red de función de base radial

La red de función de base radial (RBFN) es un tipo particular de red neuronal que se ha empleado en problemas de clasificación, identificación de sistemas no lineales y predicción de series de tiempo. La construcción de la arquitectura óptima y la obtención de los pesos en el proceso de entrenamiento

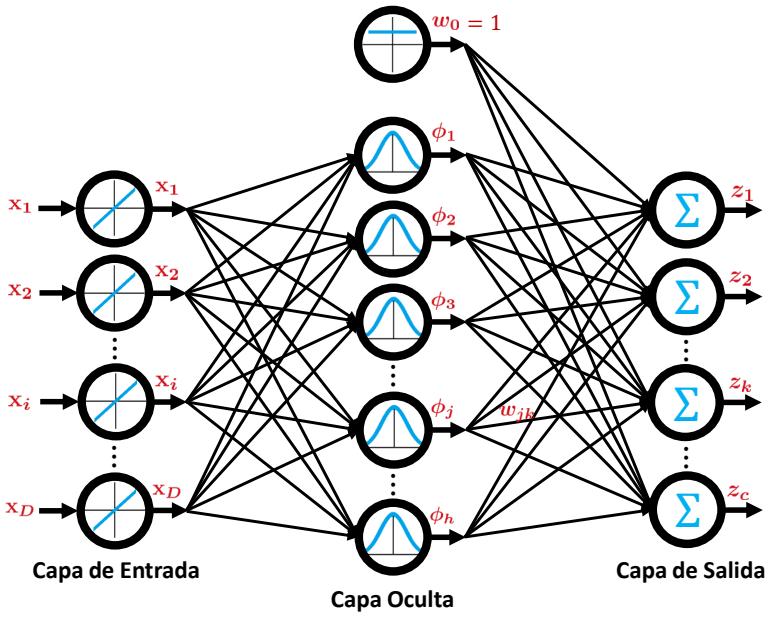


Figura 3.1: Arquitectura de una RBFN.

son los principales problemas que se abordan en el estudio de este tipo de red.

3.1.1 Topología de la red de función de base radial

La RBFN posee una topología invariante de tres capas, como se muestra en la Figura 3.1. La capa de entrada se encarga de recibir y distribuir los datos desde el exterior. La capa oculta es activada por funciones de base radial (RBF, del Inglés *Radial Basis Functions*), normalmente funciones Gaussianas, generando así respuestas no lineales de las variables provenientes de la capa de entrada. La capa de salida es activada por funciones lineales continuas; implementa una suma ponderada de las respuestas no lineales de los nodos de la capa oculta y los pesos de la red. Una de las ventajas importantes de este tipo de red es que puede generar hiperesferas de decisión al utilizar RBF en los nodos de la capa oculta, por lo que es capaz de clasificar conjuntos de datos no linealmente separables. Además, puede realizar mapeos muy complejos en donde una red neuronal perceptrón multicapa necesitaría múltiples capas ocultas [5].

3.1.2 Funciones de activación

La RBFN activa los nodos de la capa oculta al emplear RBF, las cuales se caracterizan por su respuesta monótonamente creciente o decreciente en relación a un punto central. Una función radial típica es la *función Gaussiana*, que en el caso de una entrada escalar x se tiene:

$$\phi(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right) \tag{3.1}$$

donde c y r son los parámetros del centro y radio, respectivamente. La RBF Gaussiana decrece monótonamente a medida que aumenta la distancia desde el centro, contrariamente a la RBF multicuadrática que se expresa como:

$$\phi(x) = (r^2 + (x-c)^2)^{1/2} \tag{3.2}$$

la cual aumenta monótonamente con la distancia a partir de su centro. Considerando a la RBF Gaussiana, la activación del $j-$ésimo nodo oculto en función de un patrón de entrada $\mathbf{x} \in \mathbb{R}^D$ es:

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|^2}{2\sigma_j^2}\right), \quad j = 1, \ldots, h \tag{3.3}$$

donde $\|\cdot\|$ denota distancia Euclidiana y σ_j es el radio del $j-$ésimo nodo oculto.

En general, las RBF exhiben un comportamiento local, ya que un nodo oculto se activa cuando un patrón $\mathbf{x}$ está dentro de su localidad en el espacio de características, de modo que se cumple lo siguiente:

$$\phi(\mathbf{x}) \to 1 \quad \text{cuando} \quad \|\mathbf{x}-\mathbf{c}_j\| \to 0 \tag{3.4}$$

3.1.3 Aprendizaje de la red de función de base radial

La etapa de entrenamiento de una RBFN se desarrolla en dos procesos independientes mediante un algoritmo de aprendizaje híbrido:

1. *Aprendizaje no supervisado:* Se determinan los centros de las RBF a partir de un algoritmo de agrupamiento y, posteriormente, se calculan los radios con base en los centros obtenidos. La cantidad de nodos ocultos determina los grupos que se desean obtener con un algoritmo de agrupamiento.

2. *Aprendizaje supervisado:* Dadas las respuestas de la capa oculta, los pesos se ajustan mediante un algoritmo supervisado que realiza el mapeo entrada-salida de los nodos de salida. La cantidad de nodos en la capa de salida corresponde al número de clases.

El proceso de aprendizaje se describe mediante los siguientes pasos:

1. Definir el número de nodos ocultos.

2. Calcular los centros de cada RBF con un algoritmo de agrupamiento (e.g., algoritmo $k-$means) aplicado a los datos de entrenamiento, de modo que los centros se distribuyan adecuadamente en el espacio de características.

3. Determinar los radios de cada RBF con base en los centros calculados, de tal forma que exista poco traslape entre las RBF. Existen dos maneras típicas de calcular los radios:

   - Media aritmética de las distancias Euclidianas a los p centros más cercanos:

$$\sigma_i = \frac{1}{p}\sum_{j=1}^{p} \|\mathbf{c}_i - \mathbf{c}_j\|, \quad i \neq j \tag{3.5}$$

   - Media geométrica de las distancias Euclidianas a los dos centros más cercanos:

$$\sigma_i = (\|\mathbf{c}_i - \mathbf{c}_j\|\|\mathbf{c}_i - \mathbf{c}_k\|)^{\frac{1}{2}}, \quad i \neq j \neq k \tag{3.6}$$

4. El cálculo de los pesos se hace mediante la resolución de un problema lineal de mínimos cuadrados, el cual se resuelve de manera directa empleando la matriz pseudoinversa:

$$\mathbf{W} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{Y} \tag{3.7}$$

donde $\mathbf{W}$, $\mathbf{\Phi}$ e $\mathbf{Y}$ son las matrices de pesos, respuestas de las RBF y salidas deseadas o referencias, respectivamente. Los valores de los pesos en la matriz $\mathbf{W}$ son óptimos en relación a los valores de los centros y radios de las RBF [6].

La referencia $\mathbf{Y} = \{y_n(\mathbf{x}_n)|n = 1, \ldots, N\}$, donde $y_n \in \{1, \ldots, c\}$, se binariza de acuerdo a la cantidad de clases existentes, por ejemplo, para un problema con $c = 3$ clases se tiene:

- Si $y = 1$ se convierte en $y = [1, 0, 0]$
- Si $y = 2$ se convierte en $y = [0, 1, 0]$
- Si $y = 3$ se convierte en $y = [0, 0, 1]$

En el aprendizaje supervisado cada nodo de salida genera una combinación lineal de las respuestas de la capa oculta y los pesos de la red:

$$z_k = w_{0k} + \sum_{j=1}^{h} w_{jk}\phi_j(\mathbf{x}) \tag{3.8}$$

El objetivo es minimizar las diferencias cuadradas entre la salida de la red z_k y las referencias y_k:

$$J(\mathbf{W}) = \frac{1}{2}\sum_{k=1}^{c}(y_k - z_k)^2 \tag{3.9}$$

La matriz de pesos $\mathbf{W}$ es de orden $h \times c$, donde h es la cantidad de nodos ocultos, c es el número de clases y tiene la forma:

$$\mathbf{W} = \begin{pmatrix} w_{01} & w_{02} & \cdots & w_{0k} & \cdots & w_{0c} \\ w_{11} & w_{12} & \cdots & w_{1k} & \cdots & w_{1c} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{j1} & w_{j2} & \cdots & w_{jk} & \cdots & w_{jc} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{h1} & w_{h2} & \cdots & w_{hk} & \cdots & w_{hc} \end{pmatrix} \tag{3.10}$$

Las respuestas de la capa oculta componen la matriz $\mathbf{\Phi}$, la cual es de orden $(h+1) \times n$, donde n es la cantidad de patrones de entrenamiento. La primera fila tiene valores unitarios empleados en el cálculo del *bias* y tiene la forma:

$$\mathbf{\Phi} = \begin{pmatrix} 1 & 1 & \cdots & 1 & \cdots & 1 \\ \phi_1(\mathbf{x}_1) & \phi_1(\mathbf{x}_2) & \cdots & \phi_1(\mathbf{x}_i) & \cdots & \phi_1(\mathbf{x}_n) \\ \phi_2(\mathbf{x}_1) & \phi_2(\mathbf{x}_2) & \cdots & \phi_2(\mathbf{x}_i) & \cdots & \phi_2(\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_j(\mathbf{x}_1) & \phi_j(\mathbf{x}_2) & \cdots & \phi_j(\mathbf{x}_i) & \cdots & \phi_j(\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_h(\mathbf{x}_1) & \phi_h(\mathbf{x}_2) & \cdots & \phi_h(\mathbf{x}_i) & \cdots & \phi_h(\mathbf{x}_n) \end{pmatrix} \tag{3.11}$$

La matriz de etiquetas binarias $\mathbf{Y}$ es de orden $n \times c$ y tiene la forma:

$$\mathbf{Y} = \begin{pmatrix} y_1(\mathbf{x}_1) & y_2(\mathbf{x}_1) & \cdots & y_k(\mathbf{x}_1) & \cdots & y_c(\mathbf{x}_1) \\ y_1(\mathbf{x}_2) & y_2(\mathbf{x}_2) & \cdots & y_k(\mathbf{x}_2) & \cdots & y_c(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ y_1(\mathbf{x}_i) & y_2(\mathbf{x}_i) & \cdots & y_k(\mathbf{x}_i) & \cdots & y_c(\mathbf{x}_i) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ y_1(\mathbf{x}_n) & y_2(\mathbf{x}_n) & \cdots & y_k(\mathbf{x}_n) & \cdots & y_c(\mathbf{x}_n) \end{pmatrix} \tag{3.12}$$

En el pseudocódigo del Algoritmo 3.1 se muestra el método de entrenamiento clásico de una RBFN. Con el objetivo de encontrar el número óptimo de nodos de la capa oculta, se suele emplear una estrategia de incremento gradual de los nodos ocultos hasta alcanzar el mínimo error de entrenamiento. Nótese que debe considerarse un conjunto de validación para evitar el sobreentrenamiento de la red. En el pseudocódigo del Algoritmo 3.2 se muestra la estrategia de incremento gradual para obtener la cantidad de nodos ocultos que minimiza el error de entrenamiento.

Algoritmo 3.1 Entrenamiento clásico de una RBFN

Entrada: Cantidad de nodos ocultos (h), datos de entrenamiento $(\mathbf{X})$, etiquetas de referencia $(\mathbf{Y})$
Salida: Centros $(\mathbf{C})$, radios $(\boldsymbol{\sigma})$ y pesos $(\mathbf{W})$

1: Encontrar los h centros $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_h\}$ de las RBF usando $\mathbf{X}$
2: Calcular los h radios $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_h\}$ de las RBF usando $\mathbf{C}$
3: Obtener la matriz de respuestas de la capa oculta $\boldsymbol{\Phi}$ usando $\mathbf{X}$
4: Calcular la matriz de pesos $\mathbf{W}$ usando $\boldsymbol{\Phi}$ y $\mathbf{Y}$

Algoritmo 3.2 Cálculo de la cantidad óptima de nodos ocultos mediante validación cruzada

Entrada: Mínimo de nodos ocultos (h_{min}), máximo de nodos ocultos (h_{max}),
datos de entrenamiento $(\mathcal{Z}_{train})$ y validación $(\mathcal{Z}_{val})$ divididos en K-dobleces
Salida: Cantidad de nodos ocultos $h^\star$

1: **para** $h = h_{min}$ hasta h_{max} **hacer**
2: **para** $i = 1$ hasta K **hacer**
3: Entrenar la RBFN con h nodos ocultos usando $\mathcal{Z}_{train,i}$
4: Validar la RBFN empleando $\mathcal{Z}_{val,i}$
5: Calcular el error de validación e_i generado por $\mathcal{Z}_{val,i}$
6: **fin para**
7: Promediar los errores de validación de los K-dobleces: e_h
8: **fin para**
9: Obtener la cantidad óptima de nodos ocultos: $h^\star = \operatorname{argmin}_h(e_h)$

3.1.4 Regla de decisión

Para la clasificación de un patrón desconocido se tienen c nodos en la capa de salida (uno por cada clase) y la señal del $k-$ésimo nodo de salida está dada por la función discriminante:

$$g_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{\Phi}(\mathbf{x}), \quad k = 1, \ldots, c \tag{3.13}$$

donde $\mathbf{w}_k^T$ es el vector de pesos del $k-$ésimo nodo de salida (i.e., la $k-$ésima columna de $\mathbf{W}$) y $\mathbf{\Phi}(\mathbf{x})$ es el vector de respuestas de las RBF para el vector de entrada $\mathbf{x}$.

Las funciones discriminantes se emplean para definir la regla de decisión que se utiliza en la clasificación de los patrones desconocidos. Por lo tanto, el patrón $\mathbf{x}$ es clasificado en ω_p dentro del conjunto $\Omega = \{\omega_1, \omega_2, ..., \omega_c\}$ con c clases si:

$$g_p(\mathbf{x}) > g_q(\mathbf{x}), \quad \forall p \neq q \tag{3.14}$$

3.2 Medidas de dependencia de características

En el proceso de selección de características es importante contar con métricas que permitan cuantificar la correlación que tienen los diferentes atributos de un conjunto de datos. Estas medidas de dependencia permiten conocer la relevancia que tienen diferentes características en relación a la variable de etiquetas de clase, así como la redundancia existente entre los atributos.

3.2.1 Correlación biserial puntual

El coeficiente de correlación biserial puntual es una medida de asociación entre una variable aleatoria discreta, que toma valores enteros (e.g., $1, 2, \ldots, c$), y una variable aleatoria continua [41]. Se emplea para cuantificar la relevancia que tiene el $i-$ésimo atributo x_i (con $i = 1, ..., D$) en relación a su respectiva variable de clase discreta y [10]. Este coeficiente está definido en el intervalo $[-1, 1]$, donde un valor de 0 indica que las variables no están correlacionadas, si el valor es cercano a 1 significa que las variables tienen una alta dependencia lineal, y un valor igual a -1 indica que la

relación de dependencia entre las variables es negativa. Este coeficiente se define como:

$$r(x_i, y) = \frac{1}{c}\sum_{k=1}^{c}\left[\frac{(\mu_k - \mu_l)}{S_{x_i}}\sqrt{\frac{n_k n_l}{(n_k + n_l)^2}}\right], \quad l = \{1, \ldots, c\}\backslash k \tag{3.15}$$

donde r es el grado de asociación entre x_i e y, el parámetro c denota el número de clases y S_{x_i} representa la desviación estándar de x_i. Las variables μ_k y n_k denotan media y cantidad de datos de la variable continua, respectivamente, y se calculan con los valores de x_i que pertenecen a la k-ésima clase. Por su parte, los valores μ_l y n_l se obtienen considerando a los datos de la variable continua que no forman parte de la k-ésima clase.

3.2.2 Correlación de Pearson

El coeficiente de correlación de Pearson es una medida estadística que permite cuantificar la relación lineal entre dos variables aleatorias continuas y está definido en el intervalo $[-1, 1]$. Al igual que el coeficiente de correlación biserial puntual, un valor cercano a 0 muestra baja correlación entre las variables, mientras que valores cercanos a los extremos del intervalo muestran alta dependencia lineal, tanto positiva como negativa, de acuerdo al signo del valor obtenido. La correlación de Pearson entre dos variables aleatorias cuantitativas x_i y x_j, con n valores cada una, se define como:

$$\rho(x_i, x_j) = \frac{\sum_{k=1}^{n}(x_{i,k} - \bar{x}_i)(x_{j,k} - \bar{x}_j)}{\sqrt{\sum_{k=1}^{n}(x_{i,k} - \bar{x}_i)^2}\sqrt{\sum_{k=1}^{n}(x_{j,k} - \bar{x}_j)^2}} \tag{3.16}$$

donde $\bar{x}_i$ representa la media de la variable x_i y análogamente para $\bar{x}_j$ [48].

3.2.3 Mínima-redundancia-máxima-relevancia

El criterio de mínima-redundancia-máxima-relevancia ($MRMR$) brinda información acerca de la dependencia entre atributos así como de los atributos con la variable de clases. El criterio $MRMR$ se emplea para evaluar diferentes subconjuntos de características, obteniendo así atributos que

minimicen la redundancia entre ellos y maximicen la relevancia con respecto a las etiquetas de clase.

La relevancia de atributos hace referencia a las características que causan un mayor impacto en la variación de las etiquetas de clase de un determinado conjunto de datos. La correlación biserial puntual se emplea para medir la relevancia que tiene un atributo x_i con respecto a su variable de etiquetas de clase y:

$$V = \frac{1}{n_s} \sum_{i \in S} |\rho_b(x_i, y)| \tag{3.17}$$

donde n_s es la cantidad de atributos seleccionados en el subconjunto S, ρ_b es el coeficiente de correlación biserial puntual, y V es la media de los valores de correlación de cada atributo con respecto a su etiqueta de clase.

La redundancia de los atributos se relaciona con aquellas características que presentan alta correlación entre ellas, en consecuencia, se emplea el coeficiente de correlación de Pearson para cuantificar este valor para cada par de atributos x_i y x_j seleccionados en el subconjunto S:

$$W = \frac{1}{n_s^2} \sum_{(i,j) \in S} |\rho_p(x_i, x_j)| \tag{3.18}$$

donde ρ_p es el coeficiente de correlación de Pearson y W es la media de los valores de correlación de cada par de atributos.

En las Ecuaciones 3.17 y 3.18 se considera solamente el valor absoluto de cada coeficiente de correlación, ya que sin importar su signo o su sentido de dependencia lineal, los valores de los coeficientes indican una medida de redundancia (correlación de Pearson) y relevancia (correlación biserial puntual), respectivamente.

Para el cálculo del criterio $MRMR$ se utiliza la diferencia entre la media de la correlación biserial puntual (V) y la media de la correlación de Pearson (W) como:

$$MRMR = \frac{1 + (V - W)}{2} \tag{3.19}$$

Este índice está en el rango $[0, 1]$, donde un valor cercano a 0 indica que el conjunto de características seleccionadas tiene una alta redundancia entre cada par de atributos y una baja relevancia de los mismos en relación a las etiquetas de clase, mientras que un valor cercano a 1 muestra que los atributos tienen poca información redundante entre sí y contienen información relevante con respecto a la variable de clase.

3.2.4 Distancia de Hamming

La distancia de Hamming entre dos vectores binarios de igual longitud es el número de posiciones en las que sus dígitos correspondientes difieren, y se emplea para determinar qué tan parecidos son los valores de dichos vectores. Entonces, para dos cadenas binarias $\mathbf{a}$ y $\mathbf{b}$ de longitud l, la distancia de Hamming se define como:

$$d_H(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{l} a_i \oplus b_i \tag{3.20}$$

donde $\oplus$ denota el operador *xor* [15].

En el presente trabajo se emplea la distancia de Hamming para comparar los resultados de dos algoritmos diferentes, cuantificando las diferencias entre los vectores binarios que representan las características seleccionadas por cada algoritmo.

3.3 Técnicas coadyuvantes a la clasificación

Existen diferentes técnicas relacionadas al problema de clasificación que se utilizan para el manejo de los datos y el análisis estadístico de los resultados. Estas técnicas consisten en diferentes tipos de normalización, estrategias para dividir los datos en subconjuntos de entrenamiento y prueba, y pruebas de significación estadística para validar hipótesis utilizando los resultados de clasificación obtenidos.

3.3.1 Normalización

La normalización es una técnica de procesamiento de información que se emplea para mejorar la exactitud y eficiencia de los algoritmos de clasificación. Este procedimiento escala los atributos de un conjunto de datos para que tengan un rango de valores específico, como por ejemplo $[-1, 1]$ ó $[0, 1]$. Es particularmente útil en algoritmos de clasificación que involucran redes neuronales, como la RBFN, y también en aquellos que involucran medidas de distancia, tal como la clasificación por vecinos más cercanos y algoritmos de agrupamiento. Existen varios métodos de normalización, entre los cuales destacan la *normalización min-max*, *normalización z-score*, *normalización por escala decimal* y la *normalización soft-max* [34].

En esta tesis se utiliza la normalización soft-max para reducir la influencia de valores atípicos sin removerlos del conjunto de datos, preservando la significancia de la media y desviación estándar de los atributos. Los datos son transformados de forma no lineal usando una función sigmoidal. Considérese que la variable x se transforma a una variable z con media cero y varianza unitaria como:

$$z = \frac{x - \overline{x}}{s} \tag{3.21}$$

donde $\overline{x}$ y s son la media y la desviación estandar de la variable x, respectivamente. Entonces, la

variable x es normalizada con la función tangente hiperbólica como:

$$x' = \frac{1 - \exp(-z)}{1 + \exp(-z)} \tag{3.22}$$

cuyo rango está definido entre $[-1, 1]$.

La normalización soft-max es casi lineal cerca de su valor medio y tiene una no linealidad suave en ambos extremos para asegurar que todos los valores están dentro de los rangos establecidos según la función sigmoidal.

3.3.2 Validación cruzada de K-dobleces

En el proceso de entrenamiento de la RBFN se requiere un conjunto de datos de validación para medir el desempeño de clasificación dados los parámetros de los centros, radios y pesos obtenidos con el conjunto de entrenamiento. Esto comúnmente se realiza mediante un esquema de *validación cruzada de* $K-$dobleces (del Inglés $K-$*fold cross validation*). De esta manera se mejora la generalización de la red al seleccionar un modelo que maximice el desempeño de clasificación en el conjunto de validación [19, 39].

Esta técnica divide aleatoriamente y de manera estratificada el conjunto de datos de entrenamiento en K subconjuntos, $S_1, S_2, \ldots, S_K$, manteniendo una proporción equivalente de muestras de cada clase del conjunto original. Posteriormente, se emplean $K - 1$ subconjuntos para entrenar la RBFN y obtener sus respectivos parámetros, esto es $\cup_{j=1}^{K} S_j$, $j \neq i$, donde el subconjunto S_i es empleado para evaluar el desempeño del clasificador. Este proceso se realiza K veces para dar variedad al proceso de entrenamiento, como se muestra en la Figura 3.2.

El promedio de los resultados de las K ejecuciones se toma como el resultado de la validación

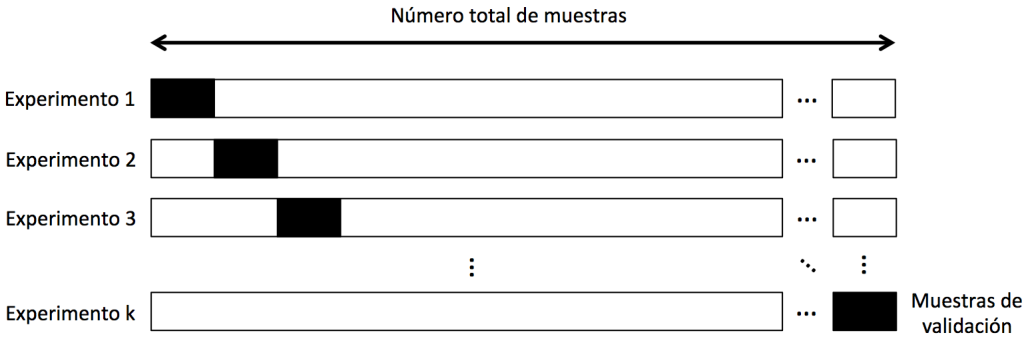


Figura 3.2: Ejemplo gráfico de un proceso de entrenamiento y validación con el método de validación cruzada con $k-$dobleces.

cruzada y una vez seleccionado el modelo de red todos los datos se emplean para entrenar el clasificador final [52].

3.3.3 Métrica de desempeño

Con el objetivo de cuantificar el desempeño de clasificación de la RBFN, se emplea el coeficiente de correlación de Matthews (MCC, del Inglés *Matthews Correlation Coefficient*), el cual mide la correlación entre las etiquetas predichas y las observadas empleando la matriz de confusión multiclase. Entonces, dado un problema de clasificación con c clases $\{1, \ldots, c\}$, la correspondiente matriz de confusión $\mathbf{C}$ es una matriz cuadrada de tamaño $c \times c$ cuya $ij-$ésima posición $\mathbf{C}_{i,j}$ es el número de elementos de clase verdadera i que han sido asignados por el clasificador a la clase j. El índice MCC es adecuado para problemas con clases desbalanceadas y se calcula como [13]:

$$MCC = \frac{N \cdot Tr(\mathbf{C}) - \sum_{kl} \mathbf{C}_k \mathbf{C}_l}{\sqrt{N^2 - \sum_{kl} \mathbf{C}_k (\mathbf{C}^T)_l} \sqrt{N^2 - \sum_{kl} (\mathbf{C}^T)_k \mathbf{C}_l}} \tag{3.23}$$

donde N es el número total de observaciones en el conjunto de datos, $Tr(\mathbf{C})$ denota la traza de $\mathbf{C}$, $\mathbf{C}_k$ es la $k-$ésima fila de $\mathbf{C}$ y $\mathbf{C}_l$ es la $l-$ésima columna de $\mathbf{C}$. El MCC está en el rango $[-1, 1]$ donde la unidad significa clasificación perfecta.

3.3.4 Pruebas de significación estadística

Las pruebas de significación estadística son utilizadas para verificar o rechazar una hipótesis, y se dividen en paramétricas y no paramétricas. En las pruebas paramétricas se asume que la distribución de los datos es normal. Las pruebas no paramétricas se emplean cuando no se asume ningún tipo de distribución particular de los datos y son robustas a valores atípicos [22].

Una prueba paramétrica empleada para evaluar la significación estadística entre dos o más grupos de muestras mutuamente independientes es el análisis de varianza (ANOVA, del Inglés *Analysis of Variance*). En esta prueba la hipótesis nula establece que todas las medias de la población son iguales mientras que la hipótesis alternativa dicta que al menos una es diferente.

En las pruebas de hipótesis se emplea el valor-*p* para determinar si los resultados son estadísticamente significativos, y de esta forma rechazar o no la hipótesis nula (H_0). Un valor-*p* toma una magnitud en el rango $[0, 1]$ que corresponde a la probabilidad que mide la evidencia en contra de la hipótesis nula. Las probabilidades más bajas proporcionan una evidencia más fuerte en contra de la hipótesis nula. El valor-*p* se compara con un nivel de significación (α) para decidir si se debe rechazar la hipótesis nula según la siguiente regla:

- Si $p \leq \alpha$, rechazar H_0
- Si $p > \alpha$, no rechazar H_0

en donde α suele tener un valor de 0.05, es decir, un intervalo de confianza del $95\,\%$.

3.4 Algoritmos de evolución diferencial

La evolución diferencial (DE, del Inglés *differential evolution*) es un algoritmo de optimización global del área de cómputo evolutivo propuesto por Storn y Price [40] que utiliza una diferencia

ponderada de vectores para perturbar a la población. Como todo algoritmo evolutivo, DE inicializa aleatoriamente una población de individuos definidos matemáticamente como vectores $M-$dimensionales con valores reales.

3.4.1 Evolución diferencial clásico

En el algoritmo DE clásico cada miembro de la población se inicializa con una distribución uniforme en los rangos definidos por el espacio de búsqueda. Posteriormente para cada generación g, el algoritmo aplica tres operadores: *mutación*, *cruza* y *selección*.

1. **Mutación**: Para cada vector objetivo $\mathbf{x}_{i,g}$ se crea un vector mutante $\mathbf{v}_{i,g}$, el cual involucra tres miembros de la población: $\mathbf{x}_{r1,g}$, $\mathbf{x}_{r2,g}$ y $\mathbf{x}_{r3,g}$ los cuales se seleccionan aleatoriamente:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F[\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}] \tag{3.24}$$

   donde $i \neq r1 \neq r2 \neq r3$ y F es un factor de escala comúnmente utilizado en el intervalo $[0.5, 1]$ que controla la amplificación de la diferencia de vectores.

2. **Cruza**: Se genera un vector de prueba $\mathbf{u}_{i,g}$ combinando la información del vector objetivo $\mathbf{x}_{i,g}$ y el vector mutante $\mathbf{v}_{i,g}$ bajo la siguiente condición:

$$\mathbf{u}_{i,g} = u_{j,i,g} \begin{cases} v_{j,i,g} & \text{si } \mathcal{U}(0,1) < CR \vee j = j_{rand} \\ x_{j,i,g} & \text{otro caso} \end{cases} \tag{3.25}$$

   donde $j = 1, \ldots, M$, j_{rand} es un valor tomado aleatoriamente del conjunto $\{1, \ldots, M\}$, $\mathcal{U}(0,1)$ es un número aleatorio unformemente distribuido en el rango $[0, 1]$, y $CR \in (0, 1)$ representa una probabilidad de cruza que controla la cantidad de información que se copia a partir del vector mutante.

3. **Selección**: Se determina el vector que sobrevive para la siguiente generación entre el vector objetivo $\mathbf{x}_{i,g}$ y el vector de prueba $\mathbf{u}_{i,g}$:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{si } f(\mathbf{u}_{i,g}) \leq f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otro caso} \end{cases} \tag{3.26}$$

donde $f(\cdot)$ es la función objetivo minimizada sin pérdida de generalidad.

Existe un paso de penalización que se realiza después del operador de cruza para evitar que se generen soluciones fuera del espacio de búsqueda. Esta operación es conocida como estrategia restrictiva *bounce-back*. Consiste en restablecer el valor de las variables del vector de prueba ($\mathbf{u}_{i,g}$) que estén fuera de los límites, y así seleccionar un nuevo valor aleatorio que se encuentre entre el valor de la variable del vector objetivo y el límite excedido [33].

En el Algoritmo 3.3 se muestra el pseudocódigo de evolución diferencial propuesto originalmente por Storn y Price.

3.4.2 Evolución diferencial con parámetros dinámicos

En esta tesis se ha implementado una variante del algoritmo de evolución diferencial clásico, donde la diferencia principal consiste en que el factor de escala (F) y la probabilidad de cruza (CR) son dinámicos y cambian en cada generación. Esto permite mejorar el proceso de exploración en las primeras generaciones y aprovechar de mejor forma la etapa de explotación en las últimas iteraciones [12]. El cálculo de los parámetros F y CR se realiza como se muestra en las Ecuaciones 3.27 y 3.28, respectivamente:

$$F = 0.5 * (1 + \mathcal{U}(0, 1)) \tag{3.27}$$

Algoritmo 3.3 Evolución diferencial clásico

Entrada: Factor de escala (F), probabilidad de cruza (CR), tamaño de la población (NP), número máximo de generaciones (G_{max})
Salida: Mejor individuo $(\mathbf{x}_{best,g})$, valor de aptitud del mejor individuo $f(\mathbf{x}_{best,g})$

```
Inicializar población de individuos aleatoriamente X_0 = {x_{1,0}, x_{2,0}, ... x_{NP,0}}
para i = 1 hasta NP hacer
    Evaluar aptitud del individuo: f(x_{i,0})
fin para
para g = 1 hasta G_max hacer
    para i = 1 hasta NP hacer
        Aplicar estrategia de mutación: v_{i,g}
        Aplicar cruza binomial: u_{i,g}
        Aplicar estrategia restrictiva bounce-back a u_{i,g}
        Evaluar aptitud del individuo de prueba: f(u_{i,g})
        si f(u_{i,g}) < f(x_{i,g}) entonces
            Reemplazar el vector objetivo con el vector de prueba: x_{i,g+1} ← u_{i,g}
        si no
            Mantener el vector objetivo en la población actual: x_{i,g+1} ← x_{i,g}
        fin si
    fin para
fin para
```

$$CR = 0.5 + (1 - 0.5) * \left(\frac{G_{max} - g}{G_{max}} \right) \tag{3.28}$$

donde G_{max} es el número máximo de generaciones y g es la generación actual. En esta versión del algoritmo también se emplean las estrategias de mutación, cruza, penalización y selección como se muestra en el Algoritmo 3.3.

Para la experimentación con el algoritmo de evolución diferencial con parámetros dinámicos se consideran las siguientes estrategias de mutación más comunes reportadas en la literatura [35]:

- **rand/1/bin**: Es la estrategia más básica, mantiene diversidad en la población y tiene un costo computacional bajo. Los individuos $\mathbf{x}_{r1,g}$, $\mathbf{x}_{r2,g}$ y $\mathbf{x}_{r3,g}$ son seleccionados de forma aleatoria de

la población actual g y son diferentes entre sí:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F[\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}] \tag{3.29}$$

- **best/1**: Tiene bajo costo computacional aunque en ocasiones compromete la convergencia de la población pues no permite una mayor diversidad entre individuos. El elemento $\mathbf{x}_{best,g}$ es el mejor elemento de la población actual g:

$$\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F[\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}] \tag{3.30}$$

- **current-to-best**: Mantiene cierto grado de diversidad; sin embargo, la diferencia producida entre el mejor individuo y el actual asegura la convergencia sobre el mejor:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F[\mathbf{x}_{best,g} - \mathbf{x}_{i,g}] + F[\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}] \tag{3.31}$$

3.4.3 Evolución diferencial con parámetros autoadaptativos

El algoritmo DE con parámetros autoadaptativos JADE [49] mantiene la estructura del algoritmo DE clásico. La diferencia principal consiste en que los parámetros F_i y CR_i están asociados a cada individuo de la población. También, esta variante del algoritmo DE puede emplear un archivo externo A de tamaño NP que guarda datos históricos del proceso de búsqueda con el objetivo principal de diversificar la población y mejorar la velocidad de convergencia.

La estrategia de mutación del algoritmo JADE se llama *current-to-$p$best*, la cual es una variante de la estrategia *current-to-best* que emplea el archivo externo A para generar individuos mutantes y se expresa como:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i[\mathbf{x}_{pbest,g} - \mathbf{x}_{i,g}] + F_i[\mathbf{x}_{r1,g} - \tilde{\mathbf{x}}_{r2,g}] \tag{3.32}$$

donde $\mathbf{x}_{pbest,g}$ es elegido aleatoriamente de los $100p\,\%$ mejores individuos en la población actual con $p \in [0, 1)$, F_i es el factor de escala asociado a $\mathbf{x}_{i,g}$, $\mathbf{x}_{r1,g}$ es tomado aleatoriamente de la población actual y $\tilde{x}_{r2,g}$ es elegido aleatoriamente de la unión de del archivo externo A con la población actual.

JADE emplea los operadores de cruza y selección descritos en las Ecuaciones 3.25 y 3.26, respectivamente. En el caso de que un individuo de prueba sustituya a un individuo de la población actual, este último es agregado al archivo externo A.

Los parámetros F_i y CR_i asociados a los individuos exitosos que trascendieron a la siguiente generación se guardan en los conjuntos S_F y S_{CR}, respectivamente, los cuales se emplean para recalcular los valores de los parámetros F_i y CR_i relacionados a los individuos de la siguiente generación.

El factor de cruza CR_i es recalculado en cada generación de acuerdo a una distribución normal con media μCR y desviación estándar de 0.1 en el intervalo $[0, 1]$ como:

$$CR_i = \text{randn}_i(\mu CR, 0.1) \tag{3.33}$$

El parámetro μCR se inicializa en 0.5 y es calculado en cada iteración como:

$$\mu CR = (1 - c) \cdot \mu CR + c \cdot \text{mean}_A(S_{CR}) \tag{3.34}$$

donde c es una constante positiva en el rango $[0, 1]$, mean_A es la media aritmética y S_{CR} es el conjunto de los parámetros de probabilidad de cruza asociados con los individuos exitosos de la

generación anterior.

Por su parte, el factor de mutación F_i es generado aleatoriamente siguiendo la distribución de Cauchy:

$$F_i = \text{randc}_i(\mu F, 0.1) \tag{3.35}$$

donde el valor de F_i es truncado a 1 si $F_i > 1$ o recalculado si $F_i \leq 0$.

El parámetro μF es inicializado con un valor de 0.5 y se actualiza en cada iteración como:

$$\mu F = (1 - c) \cdot \mu F + c \cdot \text{mean}_L(S_F) \tag{3.36}$$

donde el conjunto S_F se compone con los factores de mutación exitosos de la generación anterior y mean_L es la media de Lehmer, la cual se calcula como:

$$\text{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{3.37}$$

En el Algoritmo 3.4 se muestra el pseudocódigo de la evolución diferencial con parámetros autoadaptativos, en el cual se pueden emplear cualquiera de las siguientes estrategias de mutación:

- **current-to-$p$best** (con archivo externo):

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i[\mathbf{x}_{pbest,g} - \mathbf{x}_{i,g}] + F_i[\mathbf{x}_{r1,g} - \tilde{\mathbf{x}}_{r2,g}] \tag{3.38}$$

- **current-to-$p$best** (sin archivo externo):

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i[\mathbf{x}_{pbest,g} - \mathbf{x}_{i,g}] + F_i[\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}] \tag{3.39}$$

donde $\mathbf{x}_{r2,g}$ es tomado de la población actual y es diferente de $\mathbf{x}_{r1,g}$.

Algoritmo 3.4 Evolución diferencial con parámetros autoadaptativos (JADE)

Entrada: Tamaño de la población (NP), número máximo de generaciones (G_{max})
Salida: Mejor individuo $(\mathbf{x}_{best,g})$, valor de aptitud del mejor individuo $f(\mathbf{x}_{best,g})$

$\mu CR = 0.5$, $\mu F = 0.5$, $A = \emptyset$, $p = 0.2 * NP$
Inicializar población de individuos aleatoriamente $\mathbf{X}_0 = \{\mathbf{x}_{1,0}, \mathbf{x}_{2,0}, \ldots \mathbf{x}_{NP,0}\}$
para $i = 1$ hasta NP **hacer**
  Evaluar aptitud del individuo: $f(\mathbf{x}_{i,0})$
fin para
para $g = 1$ hasta G_{max} **hacer**
  $S_F = \emptyset$, $S_{CR} = \emptyset$
  para $i = 1$ hasta NP **hacer**
    Generar CR_i y F_i a partir de μCR y μF
    Aplicar estrategia de mutación (current-to-$p$best): $\mathbf{v}_{i,g}$
    Aplicar cruza binomial: $\mathbf{u}_{i,g}$
    Aplicar estrategia restrictiva a $\mathbf{u}_{i,g}$
    Evaluar aptitud del individuo de prueba: $f(\mathbf{u}_{i,g})$
    si $f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g})$ **entonces**
      Reemplazar el vector objetivo con el vector de prueba: $\mathbf{x}_{i,g+1} \leftarrow \mathbf{u}_{i,g}$
      Actulizar $A \leftarrow \mathbf{x}_{i,g}$; $S_{CR} \leftarrow CR_i$, $S_F \leftarrow F_i$
    si no
      Mantener el vector objetivo en la población actual: $\mathbf{x}_{i,g+1} \leftarrow \mathbf{x}_{i,g}$
    fin si
  fin para
  Remover soluciones aleatoriamente de A de modo que $|A| \leq NP$
  Recalcular μCR y μF a partir de S_{CR} y S_F
fin para

3.5 Sumario

En este capítulo se describieron las principales técnicas empleadas en el desarrollo del trabajo de tesis. Primero se hizo una descripción de la RBFN y su algoritmo de entrenamiento clásico. Después se explicaron las métricas utilizadas para la resolución del problema de selección de características. Posteriormente, se mostraron las herramientas para el manejo de los datos y el análisis estadístico

de los resultados, esto en relación a la tarea de clasificación. Finalmente, se detalló el algoritmo de evolución diferencial y sus variantes utilizadas en el presente trabajo de investigación.

4

Diseño y desarrollo del método propuesto

4.1 Introducción

En el presente capítulo se explica el método propuesto en este trabajo de tesis, el cual consiste en el entrenamiento de una RBFN empleando el algoritmo DE, considerando simultáneamente el diseño de la capa de entrada y la capa oculta, así como también el desempeño de clasificación relacionado a la capa de salida. De igual manera, se detalla un enfoque de entrenamiento exhaustivo de la RBFN contra el que se compara el desempeño del algoritmo propuesto.

4.2 Entrenamiento evolutivo

Generalmente el entrenamiento clásico de una RBFN se realiza mediante un proceso híbrido. Primero se utiliza un algoritmo de aprendizaje no supervisado para encontrar los centros y radios de la capa oculta y después, en la etapa de aprendizaje supervisado, se calculan los pesos de la red empleando el método de la pseudoinversa, el cual minimiza el error de clasificación de

manera analítica. Un enfoque alternativo para realizar el entrenamiento consiste en emplear una metaheurística como el algoritmo DE para obtener un modelo de RBFN que tenga un buen desempeño en la tarea de clasificación.

Para implementar el algoritmo DE se requiere de una representación de la solución, la cual consiste en la codificación de los parámetros de la RBFN en un vector que será perturbado durante el proceso de entrenamiento evolutivo. Además de lo anterior, es fundamental diseñar una función objetivo que evalúe el desempeño de las soluciones generadas por el algoritmo, y que considere los criterios pertinentes para dicha evaluación, los cuales son: complejidad de la arquitectura de la red y desempeño de clasificación.

4.2.1 Representación de la solución

La RBFN tiene tres capas en su arquitectura, donde los nodos de la capa de entrada corresponden a las características de los patrones del conjunto de datos. La cantidad de nodos en la capa oculta es definida *a priori* por el usuario; sin embargo, una regla comúnmente utilizada en la literatura consiste en utilizar un número entero igual a $\sqrt{N}$, ya que en la práctica éste es el número de grupos que puede representar adecuadamente a un conjunto de datos de tamaño N [31]. Cada nodo oculto se asocia a una RBF que agrupa a un conjunto de patrones, por lo que tiene un centro y radio definidos. La cantidad de nodos en la capa de salida es igual al número de clases del problema. Entonces, considerando la arquitectura de una RBFN, los parámetros que se codifican en la representación de una solución son:

- **Nodos de entrada (o características)**: Codifica una combinación específica de atributos seleccionados, lo que representa diferentes nodos activos en la capa de entrada. Contrariamente, los nodos inactivos representan características que no fueron seleccionadas.
- **Nodos ocultos**: Codifica los nodos activos en la capa oculta. Contrariamente, los nodos

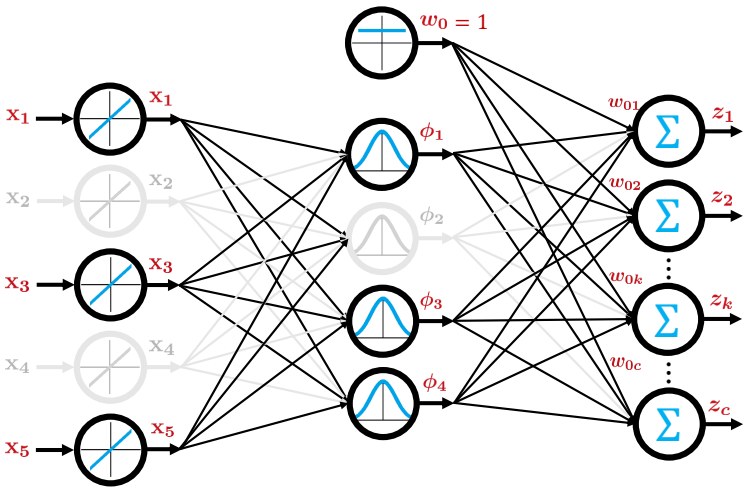
inactivos son aquellos que no participan en el proceso de clasificación.

- **Centroides**: Codifica las posiciones de los centros de las RBF en el espacio de características.

Los parámetros que no se codifican debido a que su cálculo es directo son:

- **Radios**: Es el ancho de banda de las RBF y se calculan como la distancia Euclidiana promedio a los p-centros más cercanos (Ecuación 3.5). Normalmente se utiliza $p = 2$ [4, 28].

- **Pesos**: Consisten en los pesos que se encuentran entre la capa oculta y la capa de salida, se obtienen empleando el método de mínimos cuadrados utilizando la matriz pseudoinversa para el cálculo de la solución (Ecuación 3.7) [25, 32].



Características (D)

x_1	x_2	x_3	x_4	x_5
0.6	0.2	0.9	0.3	0.7
1	0	1	0	1

Nodos ocultos (h_{max})

ϕ_1	ϕ_2	ϕ_3	ϕ_4
0.7	0.1	0.8	0.6
1	0	1	1

Centros ($D \times h_{max}$)

1.2	0	0.8	0	0.3	0	0	0	0	0	0.7	0	1.4	0	0.1	0.9	0	1.3	0	0.6
1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1
1					0					1					1				

Figura 4.1: Ejemplo de inicialización de un individuo de RBFN codificado con $D = 5$ características y $h_{max} = 4$ nodos ocultos. Los nodos de color gris representan aquellos que fueron inicializados con un valor binario igual a cero por las reglas de decisión de las capas de entrada y oculta, de modo que no participan en el entrenamiento de la red. En la parte de abajo se muestran los valores binarios que corresponden a cada sección del individuo.

Considérese un conjunto de datos de entrenamiento con N muestras y D características, la representación de la solución es de longitud fija de orden $D + h_{max} + D \times h_{max}$, donde h_{max} es el máximo número de nodos ocultos permitido.

Las secciones que codifican los nodos de entrada y nodos ocultos se inicializan aleatoriamente con distribución uniforme en el rango $[0, 1]$. Para determinar cuáles nodos de entrada y nodos ocultos participan en el proceso de entrenamiento, se utiliza una regla de decisión que activa un nodo de entrada o un nodo oculto si y sólo si su respectivo valor codificado es mayor a 0.5, en otro caso, el nodo está inactivo. Por otro lado, los centroides se inicializan con una variante del algoritmo de agrupamiento *k*-means, la cual evita nodos vacíos [30]. Además, sólo se inicializan los centroides relacionados a los nodos ocultos activos y sólo utilizando aquellos atributos seleccionados en la capa de entrada. Los valores de los centroides relacionados a posiciones inactivas se inicializan en cero.

Nótese que la cantidad de nodos de entrada activos está en el rango $[1, D]$, mientras que la cantidad de nodos ocultos se encuentra entre $h_{min} = 3$ y $h_{max} = 3\sqrt{N}$. El valor de h_{min} se establece debido a que los radios de las RBF se calculan como la distancia promedio a los dos centros más cercanos, mientras que h_{max} se utiliza para proveer un rango de búsqueda mayor en relación al número de grupos comúnmente utilizado en la literatura, es decir, $\sqrt{N}$. En la Figura 4.1 se muestra un ejemplo de la inicialización de un individuo que codifica los parámetros de una RBFN.

4.2.2 Diseño de la función objetivo

Cuando se busca minimizar el esfuerzo requerido o maximizar el beneficio deseado, se vuelve indispensable modelar estos objetivos como una función dependiente de ciertas variables de decisión. La selección de una función objetivo adecuada es una de las etapas más importantes en el proceso de optimización. En ocasiones se requiere evaluar más de un criterio para optimizar múltiples objetivos, y una forma de tratar con este problema es construir una función de costo ponderada como una

combinación lineal de las n funciones objetivo $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})$ [36]:

$$f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \alpha_2 f_2(\mathbf{x}) + \ldots + \alpha_n f_n(\mathbf{x}) \tag{4.1}$$

donde las constantes $\alpha_1, \alpha_2, \ldots, \alpha_n$ indican la importancia relativa de cada función tal que:

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{4.2}$$

Con respecto al problema del entrenamiento evolutivo de la RBFN, se plantea optimizar el desempeño de clasificación y la complejidad de la arquitectura de la red, a partir de esto se establecen tres criterios para el diseño de la función objetivo:

1. Maximizar la relación de relevancia y redundancia de las características de entrada (selección de características).

2. Maximizar la cobertura de las RBF de los nodos ocultos en el espacio de características y minimizar su cantidad (diseño del clasificador).

3. Maximizar el desempeño de clasificación.

Una vez establecidos los criterios de evaluación, se plantea el diseño de una función objetivo que considere la combinación lineal de tres funciones de costo:

$$f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \alpha_2 f_2(\mathbf{x}) + \alpha_3 f_3(\mathbf{x}) \tag{4.3}$$

donde $f_1(\mathbf{x})$, $f_2(\mathbf{x})$ y $f_3(\mathbf{x})$ representan las funciones objetivo de cada uno de los criterios mencionados y $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$, ya que en este trabajo se asume que las tres funciones poseen la misma importancia.

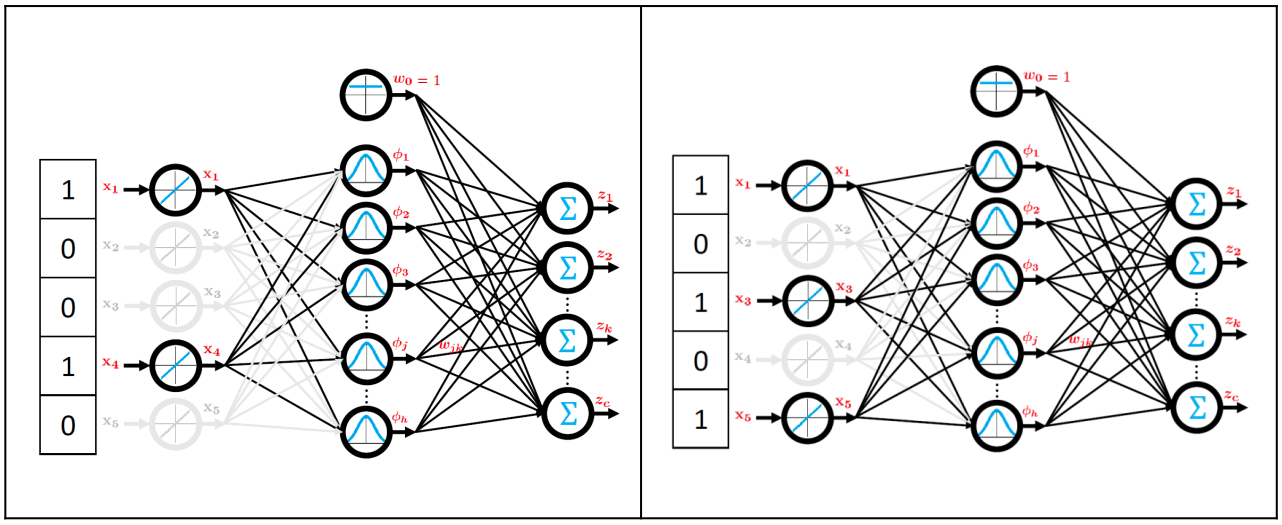


Figura 4.2: Ejemplo de selección de características: RBFN con dos atributos seleccionados (izquierda), RBFN con tres atributos seleccionados (derecha). Los vectores binarios están definidos por la regla de activación de la capa de entrada.

4.2.2.1. Selección de características

El problema de selección de características consiste en encontrar los atributos que maximicen la relación entre relevancia y redundancia de un conjunto de datos $D-$dimensional. Para cuantificar este criterio se emplea la métrica $MRMR$ (Ecuación 3.19), con la cual se evalúan distintas combinaciones de características en el rango $[1, D]$ para obtener la combinación de atributos que provean la información más relevante en relación a la variable de etiquetas de clase y que tengan la mínima redundancia de información entre ellos. Se busca maximizar el valor de la métrica $MRMR$ en el rango $[0, 1]$.

Con respecto a la arquitectura de la RBFN, la selección de características se relaciona directamente con la capa de entrada, ya que cada atributo seleccionado representa un nodo de entrada. En la Figura 4.2 se muestra un ejemplo de selección de características para un conjunto de datos de cinco dimensiones.

4.2.2.2. Tasa de cobertura

La *tasa de cobertura* (TC) es un índice que mide la correcta distribución de las RBF en el espacio de características y se encuentra en el rango $[0, 1]$. Una buena medida de cobertura muestra que los patrones se distribuyen de manera equiprobable en todos los nodos ocultos y se calcula como:

$$TC = 1 - \sqrt{\frac{\sum_{i=1}^{h}(p_i - p_r)^2}{(1 - p_r)^2 + (h - 1)p_r^2}} \tag{4.4}$$

donde h es el número de nodos ocultos activos, p_i es la probabilidad de cobertura del $i-$ésimo nodo, que a su vez se calcula como:

$$p_i = \frac{n_i}{N} \tag{4.5}$$

donde n_i es la cantidad de patrones cubiertos por el $i-$ésimo nodo oculto, N es el número de patrones de entrenamiento, y la probabilidad de referencia p_r se calcula:

$$p_r = \frac{1}{h} \tag{4.6}$$

El índice TC debe tender a la unidad para indicar una buena distribución de las RBF en el espacio de características.

4.2.2.3. Razón de nodos ocultos

La *razón de nodos ocultos* (NO) se emplea para minimizar la cantidad de nodos ocultos de la RBFN. Si el número de nodos ocultos se aproxima a $h_{max} = 3\sqrt{N}$, entonces se dice que la red tiene una complejidad alta, mientras que si se aproxima a $h_{min} = 3$ se tiene una red con baja complejidad. Por tanto, este índice calcula la fracción de nodos ocultos utilizados en el rango $[h_{min}, h_{max}]$ y se obtiene de la siguiente manera:

$$NO = 1 - \frac{h - h_{min}}{h_{max} - h_{min}} \tag{4.7}$$

donde NO está en el rango $[0, 1]$, si la red cuenta con una cantidad de nodos ocultos cercana al límite superior, NO tiene un valor cercano a cero, mientras que si el número de nodos ocultos es cercano al límite inferior, el valor de este parámetro se acerca a uno.

4.2.2.4. Eficiencia de la capa oculta

La *eficiencia* de la capa oculta (E) es una relación entre TC y NO. Una buena medida de eficiencia implica que las RBF están distribuidas de manera uniforme en el espacio de caracteríscas con un número reducido de nodos ocultos. La eficiencia de la capa oculta se puede calcular al combinar los criterios TC y NO como:

$$E = \frac{TC + NO}{2} \tag{4.8}$$

El índice E está en el rango $[0, 1]$, donde un valor cercano a la unidad indica una buena eficiencia de la capa oculta. En la Figura 4.3 se muestran cuatro posibles distribuciones de RBF en un espacio de características bidimensional con su respectivo valor de eficiencia.

4.2.2.5. Desempeño de clasificación

El desempeño de clasificación se relaciona con la capa de salida, donde el índice utilizado en esta tesis es el MCC multiclase (Ecuación 3.23), el cual mide la correlación entre las etiquetas predichas y las verdaderas. Este coeficiente se calcula a partir de la matriz de confusión multiclase y es adecuado para la clasificación de conjuntos de datos que tienen clases desbalanceadas. El valor de MCC se encuentra en el rango $[-1, 1]$, donde la unidad significa clasificación perfecta, por lo cual se desea maximizar. Cabe señalar que valores negativos de MCC indican un pobre desempeño de clasificación y pueden truncarse a un valor de cero; por tanto, el valor de MCC se redefine en el rango $[0, 1]$.

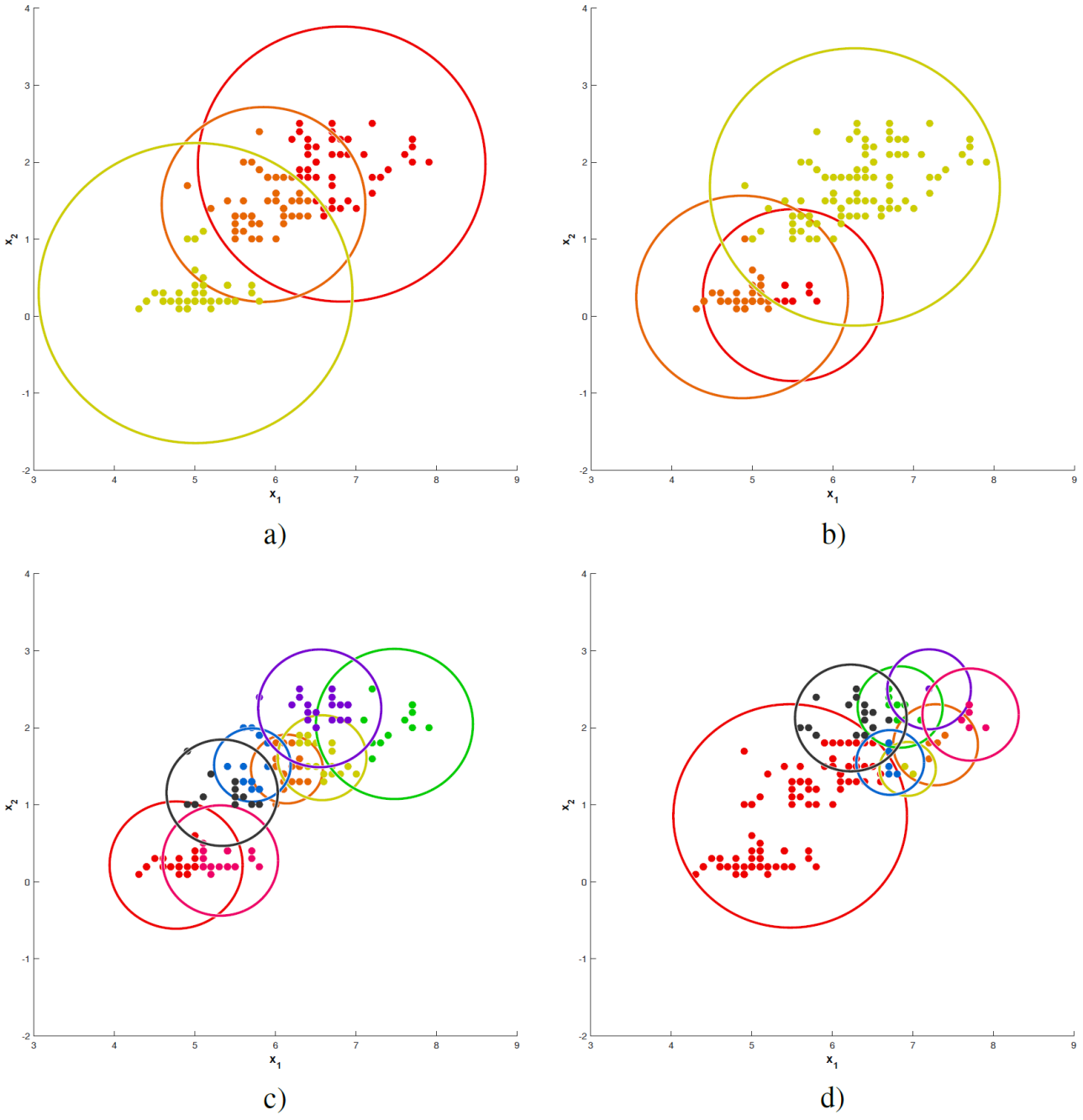


Figura 4.3: Ejemplo de distintas distribuciones de RBF, donde un valor alto del índice E corresponde a pocos nodos ocultos (h) colocados de manera uniforme en el espacio de características: a) $h = 3$ con $E = 0.976$, b) $h = 3$ con $E = 0.786$, c) $h = 8$ con $E = 0.880$ y d) $h = 8$ con $E = 0.689$.

4.2.2.6. Combinación de criterios de evaluación

De acuerdo a la Ecuación 4.1, combinando los índices de evaluación se obtiene la función objetivo ponderada:

$$f(\mathbf{x}) = \frac{MCC + MRMR + E}{3} \tag{4.9}$$

donde $f(\mathbf{x})$ está en el rango $[0, 1]$. Un valor de aptitud cercano a la unidad indica que la solución $\mathbf{x}$ (i.e., arquitectura de una RBFN codificada) cuenta con un conjunto de atributos en la capa de entrada que maximizan la relación relevancia/redundancia, una alta eficiencia en la capa oculta y un buen desempeño de clasificación de la capa de salida.

4.2.3 Algoritmo de entrenamiento evolutivo

El pseudocódigo del método de entrenamiento evolutivo propuesto se muestra en el Algoritmo 4.1, en el cual cada individuo de la población codifica los parámetros de la arquitectura de una RBFN y es inicializado dentro de los límites del espacio de búsqueda, esto es, dentro de los rangos de las características normalizadas y nodos ocultos. En esta propuesta, el desempeño de cada individuo de red es evaluado con un esquema de validación cruzada de $5-$dobleces (del Inglés *folds*), lo cual se describe en el Algoritmo 4.2. El Algortimo 4.3 se utiliza para obtener los centros, radios y pesos del modelo definitivo de RBFN, al entrenar la mejor arquitectura de red conseguida con el entrenamiento evolutivo y considerando el conjunto de datos de entrenamiento completo.

4.3 Entrenamiento exhaustivo

Una estrategia alternativa para obtener los parámetros de la mejor arquitectura de RBFN es evaluar todo el conjunto de soluciones posibles para obtener un paisaje de aptitud (del Inglés, *landscape*). Este paisaje de aptitud se obtiene al evaluar el desempeño de la arquitectura de red

Algoritmo 4.1 Entrenamiento evolutivo de una RBFN basado en DE

Entrada: Datos de entrenamiento ($\mathcal{Z}_{train}$) y validación ($\mathcal{Z}_{val}$),
rango de nodos ocultos ($[h_{min}, h_{max}]$), tamaño de la población (NP),
número máximo de generaciones (G_{max})
Salida: Parámetros del mejor individuo de RBFN ($\mathbf{x}_{best,g}$) y su valor de aptitud $f(\mathbf{x}_{best,g})$

Inicializar población de individuos de RBFN aleatoriamente: $\mathbf{X}_0 = \{\mathbf{x}_{1,0}, \mathbf{x}_{2,0}, \ldots \mathbf{x}_{NP,0}\}$
para $i = 1$ hasta NP **hacer**
  Evaluar individuo con validación cruzada: $f(\mathbf{x}_{i,0})$ usando $\mathcal{Z}_{train}$ y $\mathcal{Z}_{val}$
fin para
para $g = 1$ hasta G_{max} **hacer**
  para $i = 1$ hasta NP **hacer**
    Aplicar estrategia de mutación: $\mathbf{v}_{i,g}$
    Aplicar cruza binomial: $\mathbf{u}_{i,g}$
    Aplicar estrategia restrictiva *bounce back* a $\mathbf{u}_{i,g}$
    Evaluar individuo con validación cruzada: $f(\mathbf{u}_{i,g})$ usando $\mathcal{Z}_{train}$ y $\mathcal{Z}_{val}$
    si $f(\mathbf{u}_{i,g}) > f(\mathbf{x}_{i,g})$ **entonces**
      Reemplazar el vector objetivo con el vector de prueba: $\mathbf{x}_{i,g+1} \leftarrow \mathbf{u}_{i,g}$
    si no
      Mantener el vector objetivo en la población actual: $\mathbf{x}_{i,g+1} \leftarrow \mathbf{x}_{i,g}$
    fin si
  fin para
fin para
Obtener los parámetros del mejor individuo de RBFN ($\mathbf{x}_{best,g}$) y su valor de aptitud $f(\mathbf{x}_{best,g})$

en cada uno de los índices que conforman la función objetivo, los cuales son: $MRMR$, E y MCC.

Por cada índice se evalúan todas las posibles combinaciones de subconjuntos de características, las cuales son un total de $2^D - 1$, y todos los posibles valores de nodos ocultos que están en el rango $[3, 3\sqrt{N}]$, donde D y N son las dimensiones y el número de patrones en el conjunto de entrenamiento, respectivamente. Por tanto, se evalúan $(3\sqrt{N} - 3 + 1) \cdot (2^D - 1)$ posibles configuraciones de RBFN. Una configuración se puede definir como una solución $\mathbf{x}_{i,j}$, con $i = 1, ..., 2^D - 1$ y $j = 3, ..., 3\sqrt{N}$, para la cual $f(\mathbf{x}_{i,j})$ es la aptitud de la solución de acuerdo a la función objetivo en la Ecuación 4.9, lo cual representa un punto de altitud en el paisaje de aptitud. De este modo, el punto más alto del paisaje de aptitud se considera como la mejor solución de arquitectura de RBFN. El método de entrenamiento exhaustivo se muestra en el Algoritmo 4.4, en donde se emplea un esquema de

Algoritmo 4.2 Evaluación del individuo con un esquema de validación cruzada de 5-dobleces

Entrada: Datos de entrenamiento ($\mathcal{Z}_{train}$) y validación ($\mathcal{Z}_{val}$) divididos en 5-dobleces, individuo a evaluar ($\mathbf{x}$)

Salida: Aptitud del individuo: $f(\mathbf{x})$

1: A partir del individuo $\mathbf{x}$ obtener: características seleccionadas ($\mathcal{F}_{sel}$) y centros activos ($\mathbf{C}$) considerando $\mathcal{F}_{sel}$
2: **para** $k = 1$ hasta 5 **hacer**
3: Seleccionar $\mathcal{F}_{sel}$ en $\mathcal{Z}_{train,k}$ para generar $\mathbf{X}_{train,k}$
4: Seleccionar $\mathcal{F}_{sel}$ en $\mathcal{Z}_{val,k}$ para generar $\mathbf{X}_{val,k}$
5: Obtener etiquetas verdaderas de $\mathcal{Z}_{train,k}$ para generar $\mathbf{Y}_{train,k}$
6: Obtener etiquetas verdaderas de $\mathcal{Z}_{val,k}$ para generar $\mathbf{Y}_{val,k}$
7: Calcular los radios $\boldsymbol{\sigma}_k$ de las RBF usando $\mathbf{C}$
8: Calcular la matriz $\mathbf{\Phi}_k$ de respuestas RBF usando $\mathbf{X}_{train,k}$, $\boldsymbol{\sigma}_k$, y $\mathbf{C}$
9: Calcular la matriz de pesos $\mathbf{W}_k$ con $\mathbf{\Phi}_k$ y $\mathbf{Y}_{train,k}$
10: Computar etiquetas predichas $\mathbf{Y}_{pred,k}$ usando $\mathbf{X}_{val,k}$, $\boldsymbol{\sigma}_k$, $\mathbf{C}$ y $\mathbf{W}_k$
11: Calcular $MRMR(k)$ usando $\mathbf{X}_{train,k}$
12: Calcular $E(k)$ usando $\mathbf{X}_{val,k}$ y $\mathbf{C}$ en el rango de nodos ocultos $[3, 3\sqrt{N}]$
13: Calcular $MCC(k)$ usando $\mathbf{Y}_{pred,k}$ y $\mathbf{Y}_{val,k}$
14: **fin para**
15: Obtener aptitud del individuo $f(\mathbf{x}) \leftarrow \frac{1}{15}\sum_{k=1}^{5} MRMR(k) + E(k) + MCC(k)$

Algoritmo 4.3 Entrenamiento del mejor individuo de RBFN

Entrada: Parámetros del mejor individuo de RBFN ($\mathbf{x}_{best,g}$), conjunto de entrenamiento completo ($\mathcal{Z}_{train} \cup \mathcal{Z}_{val}$) y etiquetas de referencia ($\mathbf{Y}$)

Salida: Modelo de RBFN definitivo: centros ($\mathbf{C}$), radios ($\boldsymbol{\sigma}$) y pesos ($\mathbf{W}$)

1: A partir de los parámetros de $\mathbf{x}_{best,g}$ obtener: características seleccionadas ($\mathcal{F}_{sel}$), cantidad de nodos ocultos (h) y centros activos ($\mathbf{C}$) considerando $\mathcal{F}_{sel}$
2: Calcular los h radios $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_h\}$ de las RBF usando $\mathbf{C}$
3: Obtener la matriz de respuestas de la capa oculta $\mathbf{\Phi}$ usando $\mathcal{Z}_{train} \cup \mathcal{Z}_{val}$
4: Calcular la matriz de pesos $\mathbf{W}$ usando $\mathbf{\Phi}$ y $\mathbf{Y}$
5: Obtener el modelo de RBFN definitivo: centros ($\mathbf{C}$), radios ($\boldsymbol{\sigma}$) y pesos ($\mathbf{W}$)

validación cruzada de $5-$dobleces para evaluar cada arquitectura de red del paisaje de aptitud.

En la Figura 4.4 se muestra un ejemplo de un paisaje de aptitud generado al evaluar cada una de las soluciones o arquitecturas de RBFN en la función objetivo (Ecuación 4.9) utilizando el conjunto de datos *Seeds* con siete atributos, tres clases y 199 instancias; el punto rojo representa la mejor arquitectura de RBFN con tres características y tres nodos ocultos.

Algoritmo 4.4 Entrenamiento exhaustivo de una RBFN

Entrada: Datos de entrenamiento ($\mathcal{Z}_{train}$) y validación ($\mathcal{Z}_{val}$) divididos en 5-dobleces, tamaño del conjunto de datos (N), número de características (D)
Salida: Parámetros de la mejor arquitectura de RBFN ($\mathbf{x}^*$) y sus valores en el paisaje de aptitud $f(\mathbf{x}^*)$

Definir todas las posibles combinaciones de atributos: $A = \{a_1, a_2, ..., a_{2^D-1}\}$
para $i = 1$ hasta $2^D - 1$ **hacer**
  para $j = 3$ hasta $3\sqrt{N}$ **hacer**
    para $k = 1$ hasta 5 **hacer**
      Obtener una combinación de atributos: a_i
      Obtener número de nodos ocultos: $h_j = j$
      Generar una solución de RBFN: $\mathbf{x}_{i,j} = (a_i, h_j)$
      Seleccionar a_i en $\mathcal{Z}_{train,k}$ para generar $\mathbf{X}_{train,k}$
      Seleccionar a_i en $\mathcal{Z}_{val,k}$ para generar $\mathbf{X}_{val,k}$
      Obtener etiquetas verdaderas de $\mathcal{Z}_{train,k}$ para generar $\mathbf{Y}_{train,k}$
      Obtener etiquetas verdaderas de $\mathcal{Z}_{val,k}$ para generar $\mathbf{Y}_{val,k}$
      Calcular los centroides $\mathbf{C}_k$ de las h_j RBF usando el algoritmo k-means
      Calcular los radios $\boldsymbol{\sigma}_k$ de las RBF usando $\mathbf{C}$
      Calcular la matriz $\mathbf{\Phi_k}$ de respuestas RBF usando $\mathbf{X}_{train,k}$, $\boldsymbol{\sigma}_k$, y $\mathbf{C}$
      Calcular la matriz de pesos $\mathbf{W}_k$ con $\mathbf{\Phi}_k$ y $\mathbf{Y}_{train,k}$
      Computar etiquetas predichas $\mathbf{Y}_{pred,k}$ usando $\mathbf{X}_{val,k}$, $\boldsymbol{\sigma}_k$, $\mathbf{C}$ y $\mathbf{W}_k$
      Calcular $MRMR(k)$ usando $\mathbf{X}_{train,k}$
      Calcular $E(k)$ usando $\mathbf{X}_{val,k}$ y $\mathbf{C}$ en el rango de nodos ocultos $[3, 3\sqrt{N}]$
      Calcular $MCC(k)$ usando $\mathbf{Y}_{pred,k}$ y $\mathbf{Y}_{val,k}$
    fin para
    $f(\mathbf{x}) \leftarrow \frac{1}{15}\sum_{k=1}^{5} MRMR(k) + E(k) + MCC(k)$
  fin para
fin para
Obtener la mejor arquitectura de RBFN: $\mathbf{x}^\star = \mathrm{argmax}_{(i,j)}\ f(\mathbf{x})$

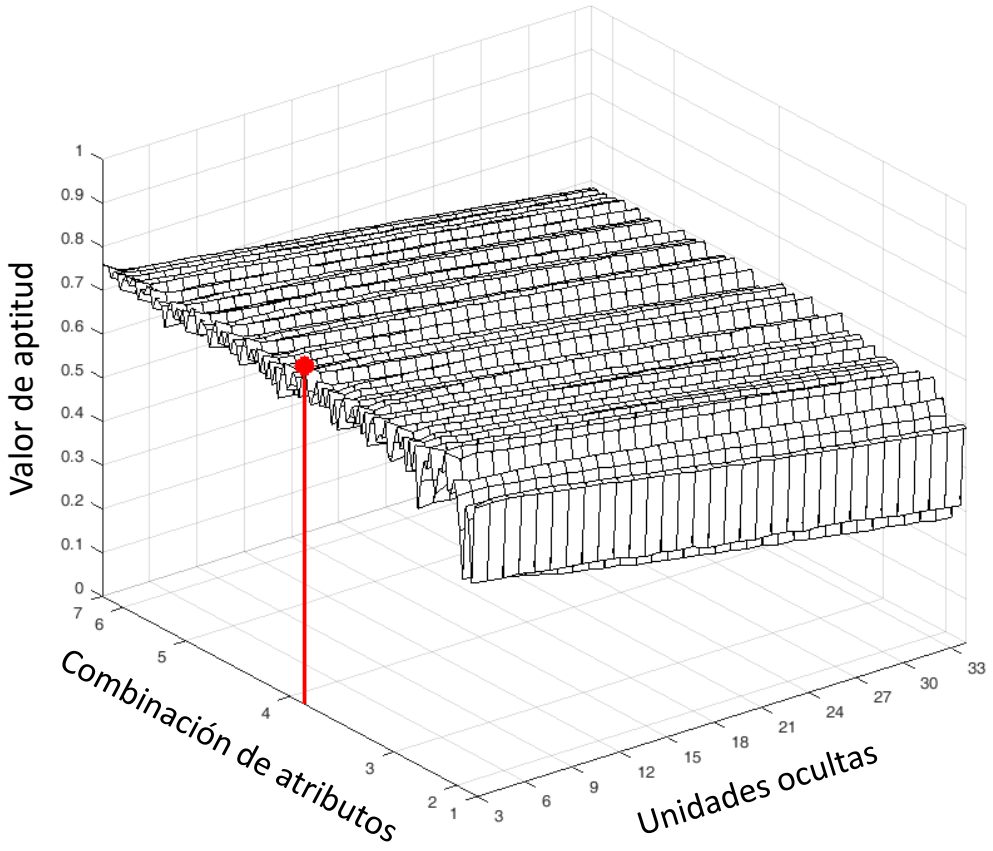


Figura 4.4: Paisaje de aptitud de la función objetivo del conjunto de datos Seeds. El punto rojo representa la mejor arquitectura de RBFN con un valor de aptitud de 0.775.

4.4 Sumario

En este capítulo se describió el diseño y desarrollo del método propuesto para el entrenamiento evolutivo de una RBFN, el cual está basado en el algoritmo de evolución diferencial. También se mostraron los parámetros de la arquitectura de red que se codifican en la representación de la solución. De igual forma se detalló el cálculo de los parámetros que no se codifican en el individuo. Se explicaron los criterios e índices utilizados en el diseño de la función objetivo. Finalmente se

describió el método de entrenamiento exhaustivo contra el cual se comparan los resultados obtenidos con el método propuesto.

5

Experimentación y resultados

5.1 Introducción

En este capítulo se muestran los conjuntos de datos utilizados, se detalla la metodología de desarrollo experimental y se describe el análisis estadístico empleado para comparar los resultados obtenidos con los métodos de entrenamiento exhaustivo y evolutivo. Además, se describe la selección de la variante del algoritmo de evolución diferencial y el análisis de convergencia empleado para reducir el número de generaciones del algoritmo de entrenamiento evolutivo. Finalmente, se lleva a cabo una comparativa del desempeño de ambos algoritmos de entrenamiento considerando únicamente el problema de la selección de características.

5.2 Conjuntos de datos utilizados

Los conjuntos de datos empleados en la experimentación poseen variables continuas y fueron obtenidos del repositorio de aprendizaje de máquinas que provee de forma gratuita la Universidad de

Conjunto de datos	Atributos	Clases	Instancias
Echocardiogram data	9	2	61
Glass identification	9	6	214
Heart disease cleveland	13	2	297
Iris data	4	3	150
BUPA liver disorders	6	2	341
Pima indians diabetes	8	2	768
Seeds	7	3	199
Thyroid gland data	5	3	215
Vowel recognition data	11	11	990
Wine recognition data	13	3	178

Tabla 5.1: Descripción de los conjuntos de datos.

California en Irvine [23] y son mostrados en la Tabla 5.1.

5.3 Detalles de la experimentación

En el proceso de experimentación se lleva a cabo el entrenamiento de la arquitectura de una RBFN considerando los algoritmos exhaustivo y evolutivo, los cuales tienen como entrada un conjunto de datos dividido en dos secciones: entrenamiento y prueba, que componen el $80\,\%$ y $20\,\%$ del total de los datos, respectivamente. En ambos algoritmos se utiliza un esquema de validación cruzada de *5*-dobleces, ya que con esta división eventualmente todos los datos se emplean para entrenar y validar el clasificador, esta cantidad de dobleces se usa típicamente en la literatura [9, 20]. De esta forma se emplean los datos de entrenamiento para generar cada modelo de arquitectura de red, la cual consiste en la cantidad de nodos de entrada, correspondiente al conjunto de características seleccionadas, y el número de nodos ocultos que maximice el desempeño de clasificación y minimice la complejidad de la arquitectura de la red. De esta forma, se selecciona la mejor arquitectura de red obtenida por el algoritmo y se utiliza el conjunto completo de datos de entrenamiento para generar los centros, radios y pesos del modelo definitivo de RBFN. El esquema de la metodología de desarrollo experimental se muestra en la Figura 5.1.

Se realizaron 31 experimentos independientes con ambos algoritmos de entrenamiento. En relación a los experimentos con el algoritmo de entrenamiento evolutivo se emplearon 100 individuos en la población y 1000 generaciones. En la variante del algoritmo de evolución diferencial con parámetros dinámicos se utilizaron las estrategias de mutación *rand/1/bin*, *best/1* y *current-to-best*. En los experimentos con el algoritmo JADE se utilizó la estrategia de mutación *current-to-$p$best* con archivo externo y sin archivo externo. Además, se realizó un análisis de varianza de un factor (del Inglés, *one-way ANOVA*) empleando un intervalo de confianza del $95\,\%$ ($\alpha = 0.05$) y el método Tukey-Kramer para determinar la significación estadística entre los resultados de los algoritmos de entrenamiento evolutivo y entrenamiento exhaustivo. En el análisis estadístico es de interés comparar los resultados relacionados al desempeño de clasificación, en términos del índice MCC y la complejidad de la red, en relación al porcentaje de características seleccionadas y la cantidad de nodos ocultos elegidos por cada algoritmo. Además, se compara la consistencia de las soluciones respecto a la selección de atributos de cada algoritmo de entrenamiento, esto se hace empleando la distancia de Hamming entre los vectores binarios de características seleccionadas (generados por la regla de activación de nodos) en cada experimento.

La plataforma de experimentación constó de una computadora con cuatro núcleos a 3.5 GHz (Intel i7 4770k) y 32 GB de RAM. Todos los algoritmos fueron desarrollados en el lenguaje MATLAB 2014a (The MathWorks).

5.4 Resultados experimentales

En esta sección se comparan los resultados del método de entrenamiento exhaustivo contra las variantes del algoritmo de entrenamiento basado en evolución diferencial.

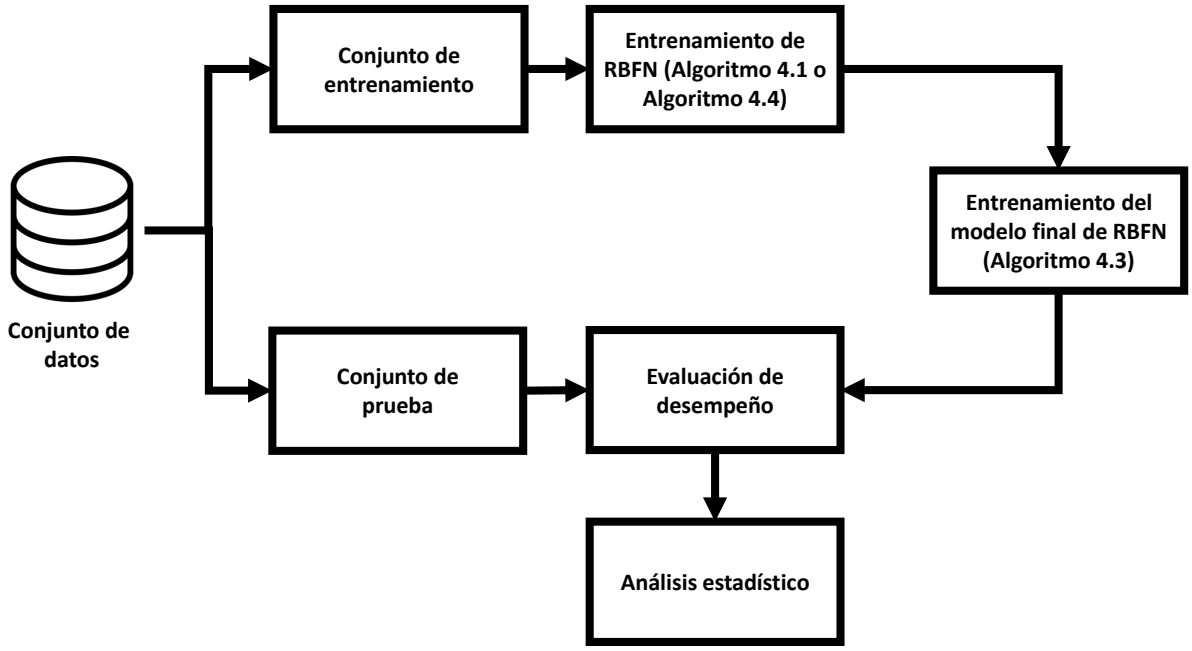


Figura 5.1: Esquema de la metodología de desarrollo experimental.

5.4.1 Análisis de desempeño

En las Tablas 5.2, 5.3 y 5.4 se resumen los valores de la media y la desviación estándar de los resultados experimentales correspondientes al desempeño de clasificación, la cantidad de nodos ocultos y el porcentaje de características seleccionadas, respectivamente.

	1	2	3	4	5	⋯	29	30	31
1	$dH_{1,1}$	$dH_{1,2}$	$dH_{1,3}$	$dH_{1,4}$	$dH_{1,5}$		$dH_{1,29}$	$dH_{1,30}$	$dH_{1,31}$
2	$dH_{2,1}$	$dH_{2,2}$	$dH_{2,3}$	$dH_{2,4}$	$dH_{2,5}$		$dH_{2,29}$	$dH_{2,30}$	$dH_{2,31}$
3	$dH_{3,1}$	$dH_{3,2}$	$dH_{3,3}$	$dH_{3,4}$	$dH_{3,5}$		$dH_{3,29}$	$dH_{3,30}$	$dH_{3,31}$
4	$dH_{4,1}$	$dH_{4,2}$	$dH_{4,3}$	$dH_{4,4}$	$dH_{4,5}$		$dH_{4,29}$	$dH_{4,30}$	$dH_{4,31}$
5	$dH_{5,1}$	$dH_{5,2}$	$dH_{5,3}$	$dH_{5,4}$	$dH_{5,5}$		$dH_{5,29}$	$dH_{5,30}$	$dH_{5,31}$
⋮									
29	$dH_{29,1}$	$dH_{29,2}$	$dH_{29,3}$	$dH_{29,4}$	$dH_{29,5}$		$dH_{29,29}$	$dH_{29,30}$	$dH_{29,31}$
30	$dH_{30,1}$	$dH_{30,2}$	$dH_{30,3}$	$dH_{30,4}$	$dH_{30,5}$		$dH_{30,29}$	$dH_{30,30}$	$dH_{30,31}$
31	$dH_{31,1}$	$dH_{31,2}$	$dH_{31,3}$	$dH_{31,4}$	$dH_{31,5}$		$dH_{31,29}$	$dH_{31,30}$	$dH_{31,31}$

Figura 5.2: Arreglo de elementos de la distancia de Hamming entre los vectores binarios de características seleccionadas por cada experimento, donde $dH_{i,j}$ es la distancia de Hamming entre los vectores resultantes del $i-$ésimo y $j-$ésimo experimentos. Los elementos de la diagonal principal tienen un valor igual a cero, ya que corresponden a la distancia de Hamming entre los vectores de los mismos experimentos.

Conjunto de Datos	Método Exhaustivo	DE rand/1/bin	DE best/1	DE current-to-best	JADE con archivo	JADE sin archivo
Echocardiogram data	0.87 ±0.12	**0.93 ±0.13** (=)	0.91 ±0.15 (=)	0.91 ±0.15 (=)	0.88 ±0.16 (=)	0.88 ±0.17 (=)
Glass identification	0.52 ±0.10	0.52 ±0.09 (=)	0.53 ±0.09 (=)	**0.54 ±0.10** (=)	0.50 ±0.09 (=)	0.48 ±0.09 (=)
Heart disease cleveland	**0.64 ±0.08**	0.61 ±0.07 (=)	0.61 ±0.07 (=)	0.60 ±0.07 (=)	0.59 ±0.07 (=)	0.60 ±0.08 (=)
Iris data	0.90 ±0.07	0.92 ±0.07 (=)	0.92 ±0.06 (=)	**0.93 ±0.08** (=)	0.90 ±0.07 (=)	0.91 ±0.07 (=)
BUPA liver disorders	0.17 ±0.11	0.38 ±0.11 (+)	0.34 ±0.12 (+)	**0.39 ±0.12** (+)	0.35 ±0.15 (+)	0.33 ±0.11 (+)
Pima indians diabetes	**0.47 ±0.06**	0.45 ±0.07 (=)	0.45 ±0.06 (=)	0.46 ±0.06 (=)	0.42 ±0.06 (−)	0.42 ±0.09 (=)
Seeds	0.86 ±0.05	0.90 ±0.05 (+)	0.91 ±0.05 (+)	0.91 ±0.05 (+)	**0.93 ±0.05** (+)	0.92 ±0.06 (+)
Thyroid gland data	**0.90 ±0.06**	0.83 ±0.09 (−)	0.85 ±0.07 (=)	0.87 ±0.07 (=)	0.84 ±0.11 (−)	0.87 ±0.07 (=)
Vowel recognition data	0.54 ±0.04	0.60 ±0.04 (+)	0.65 ±0.05 (+)	**0.67 ±0.04** (+)	0.57 ±0.05 (=)	0.60 ±0.04 (+)
Wine recognition data	**0.95 ±0.05**	0.94 ±0.04 (=)	0.93 ±0.05 (=)	0.94 ±0.05 (=)	0.92 ±0.06 (=)	0.91 ±0.05 (−)

Tabla 5.2: Desempeño de clasificación en términos de MCC. Los símbolos denotan: (+) estadísticamente superior ($p < 0.05$), (=) estadísticamente igual ($p \geq 0.05$) y (−) estadísticamente inferior ($p < 0.05$), en relación al método exhaustivo. En negritas se resaltan los mejores resultados para cada conjunto de datos.

Conjunto de Datos	Método Exhaustivo	DE rand/1/bin	DE best/1	DE current-to-best	JADE con archivo	JADE sin archivo
Echocardiogram data	3.13 ±0.34	**3.00 ±0.00** (=)	3.06 ±0.36 (=)	3.03 ±0.18 (=)	**3.00 ±0.00** (=)	**3.00 ±0.00** (=)
Glass identification	9.74 ±1.50	7.03 ±2.02 (+)	8.03 ±2.37 (+)	7.00 ±1.67 (+)	6.42 ±1.18 (+)	**6.16 ±1.29** (+)
Heart disease cleveland	**3.13 ±0.34**	3.58 ±0.72 (−)	3.61 ±0.84 (−)	3.29 ±0.46 (=)	3.23 ±0.43 (=)	3.23 ±0.50 (=)
Iris data	3.19 ±0.60	**3.00 ±0.00** (=)	3.19 ±0.40 (=)	**3.00 ±0.00** (=)	**3.00 ±0.00** (=)	**3.00 ±0.00** (=)
BUPA liver disorders	9.94 ±3.42	7.81 ±1.85 (+)	7.45 ±1.86 (+)	5.90 ±2.20 (+)	5.35 ±1.47 (+)	**5.23 ±1.54** (+)
Pima indians diabetes	3.84 ±0.82	5.32 ±1.85 (−)	5.52 ±2.32 (−)	3.74 ±1.00 (=)	**3.26 ±0.51** (=)	3.58 ±0.76 (=)
Seeds	**3.00 ±0.00**	3.19 ±0.48 (=)	3.10 ±0.30 (=)	3.06 ±0.36 (=)	3.03 ±0.18 (=)	**3.00 ±0.00** (=)
Thyroid gland data	6.87 ±0.99	3.68 ±0.94 (+)	4.19 ±0.87 (+)	**3.55 ±0.57** (+)	3.71 ±0.46 (+)	3.84 ±0.37 (+)
Vowel recognition data	20.13 ±4.14	22.61 ±3.42 (−)	23.48 ±3.19 (−)	20.58 ±2.90 (=)	**19.90 ±3.22** (=)	21.03 ±2.97 (=)
Wine recognition data	**3.00 ±0.00**	3.16 ±0.45 (−)	3.03 ±0.18 (=)	3.03 ±0.18 (=)	**3.00 ±0.00** (=)	**3.00 ±0.00** (=)

Tabla 5.3: Cantidad de nodos ocultos. Los símbolos denotan: (+) estadísticamente superior ($p < 0.05$), (=) estadísticamente igual ($p \geq 0.05$) y (−) estadísticamente inferior ($p < 0.05$), en relación al método exhaustivo. En negritas se resaltan los mejores resultados para cada conjunto de datos.

Conjunto de Datos	Método Exhaustivo	DE rand/1/bin	DE best/1	DE current-to-best	JADE con archivo	JADE sin archivo
Echocardiogram data	55.91 ±15.04	**49.10 ±5.57** (=)	57.71 ±11.97 (=)	59.86 ±11.36 (=)	55.91 ±6.07 (=)	54.48 ±5.25 (=)
Glass identification	**73.12 ±12.43**	73.84 ±12.68 (=)	80.65 ±12.49 (=)	79.93 ±10.89 (=)	81.36 ±9.25 (=)	81.36 ±10.50 (=)
Heart disease cleveland	**62.78 ±10.53**	68.49 ±8.27 (=)	79.90 ±12.03 (−)	78.66 ±11.36 (−)	80.65 ±11.04 (−)	78.41 ±10.40 (−)
Iris data	**61.29 ±19.19**	79.84 ±10.04 (−)	79.03 ±19.47 (−)	80.65 ±10.63 (−)	84.68 ±12.38 (−)	87.10 ±12.70 (−)
BUPA liver disorders	**66.67 ±12.17**	95.70 ±8.57 (−)	90.86 ±11.25 (−)	95.16 ±7.69 (−)	96.77 ±6.69 (−)	95.70 ±7.41 (−)
Pima indians diabetes	**55.65 ±9.04**	60.48 ±11.68 (=)	73.79 ±14.56 (−)	74.19 ±13.28 (−)	66.94 ±12.31 (−)	65.32 ±13.19 (−)
Seeds	**51.15 ±11.53**	56.68 ±6.88 (=)	62.21 ±14.07 (−)	61.75 ±10.68 (−)	58.06 ±3.57 (−)	56.22 ±3.57 (=)
Thyroid gland data	**92.90 ±12.16**	97.42 ±6.82 (=)	98.06 ±6.01 (−)	98.71 ±4.99 (−)	99.35 ±3.59 (−)	100.00 ±0.00 (−)
Vowel recognition data	**72.43 ±10.36**	79.18 ±7.12 (−)	86.51 ±7.37 (−)	86.51 ±8.09 (−)	81.23 ±7.40 (−)	79.47 ±9.67 (−)
Wine recognition data	73.45 ±8.39	**71.22 ±10.87** (=)	78.66 ±9.46 (=)	76.18 ±10.76 (=)	73.70 ±7.10 (=)	74.94 ±8.88 (=)

Tabla 5.4: Porcentaje de características seleccionadas. Los símbolos denotan: (+) estadísticamente superior ($p < 0.05$), (=) estadísticamente igual ($p \geq 0.05$) y (−) estadísticamente inferior ($p < 0.05$), en relación al método exhaustivo. En negritas se resaltan los mejores resultados para cada conjunto de datos.

Conjunto de Datos	Método Exhaustivo	DE rand/1/bin	DE best/1	DE current-to-best	JADE con archivo	JADE sin archivo
Echocardiogram data	3.45 ±1.35	**0.98 ±0.94** (+)	2.48 ±1.20 (+)	2.40 ±1.16 (+)	1.22 ±1.02 (+)	1.32 ±0.84 (+)
Glass identification	3.00 ±1.93	2.89 ±1.31 (=)	2.62 ±1.28 (+)	2.52 ±1.23 (+)	2.10 ±1.18 (+)	**2.06 ±1.17** (+)
Heart disease cleveland	3.30 ±1.78	3.70 ±1.45 (−)	3.39 ±1.55 (=)	3.72 ±1.48 (−)	**3.20 ±1.44** (=)	3.55 ±1.53 (=)
Iris data	0.92 ±0.79	**0.43 ±0.59** (+)	0.90 ±0.79 (=)	0.46 ±0.59 (+)	0.79 ±0.68 (+)	0.79 ±0.66 (+)
BUPA liver disorders	2.21 ±1.10	0.45 ±0.62 (+)	0.96 ±0.83 (+)	0.50 ±0.57 (+)	**0.32 ±0.47** (+)	0.45 ±0.55 (+)
Pima indians diabetes	1.28 ±1.13	**1.17 ±0.96** (=)	2.33 ±1.17 (−)	2.04 ±1.12 (−)	1.75 ±0.97 (−)	1.62 ±1.04 (−)
Seeds	1.00 ±0.94	1.65 ±1.16 (−)	2.46 ±1.19 (−)	2.24 ±1.10 (−)	1.26 ±0.94 (−)	**0.98 ±1.06** (=)
Thyroid gland data	0.62 ±0.73	0.25 ±0.45 (+)	0.18 ±0.39 (+)	0.12 ±0.33 (+)	0.06 ±0.25 (+)	**0.00 ±0.00** (+)
Vowel recognition data	2.41 ±1.40	2.24 ±1.16 (=)	1.57 ±0.97 (+)	**1.36 ±0.89** (+)	1.75 ±1.03 (+)	1.88 ±1.07 (+)
Wine recognition data	3.31 ±1.82	3.71 ±1.60 (−)	3.66 ±1.42 (−)	3.56 ±1.39 (=)	**2.70 ±1.36** (+)	3.23 ±1.44 (=)

Tabla 5.5: Distancia Hamming entre los vectores binarios de características seleccionadas. Los símbolos denotan: (+) estadísticamente superior ($p < 0.05$), (=) estadísticamente igual ($p \geq 0.05$) y (−) estadísticamente inferior ($p < 0.05$), en relación al método exhaustivo. En negritas se resaltan los mejores resultados para cada conjunto de datos.

Conjunto de datos ($D\|N$)	Método Exhaustivo	DE rand/1/bin	DE best/1	DE current-to-best	JADE con archivo	JADE sin archivo
Echocardiogram data ($9\|61$)	189.95	**165.69**	201.74	208.43	291.79	290.24
Glass identification ($9\|214$)	897.82	**247.17**	475.49	476.48	564.34	551.88
Heart disease cleveland ($13\|247$)	11454.69	**359.73**	364.96	364.49	487.06	483.77
Iris data ($4\|150$)	**20.34**	188.24	240.66	243.99	347.43	342.48
BUPA liver disorders ($6\|341$)	**175.69**	239.59	309.59	312.95	425.97	426.97
Pima indians diabetes ($8\|768$)	1784.85	**371.12**	368.49	370.53	500.81	494.82
Seeds ($7\|199$)	**148.59**	269.38	331.10	341.09	446.13	446.38
Thyroid gland data ($5\|215$)	**52.84**	271.19	353.34	358.78	467.55	452.87
Vowel recognition data ($11\|990$)	21745.78	**576.15**	1320.49	1266.92	1332.83	1326.24
Wine recognition data ($13\|178$)	6150.36	**493.91**	550.21	552.12	703.56	686.25

Tabla 5.6: Tiempo de cómputo promedio de los 31 experimentos medido en segundos. En negritas se resaltan los menores valores de tiempo de cómputo.

Los resultados del desempeño de clasificación de la Tabla 5.2 muestran que el enfoque propuesto es estadísticamente igual y en algunos casos superior al enfoque de entrenamiento exhaustivo. La variante del algoritmo de evolución diferencial *current-to-best* es la que presenta el mejor desempeño de clasificación en más ocasiones que los otros algoritmos de entrenamiento evolutivo. Además, los resultados del análisis ANOVA muestran que esta estrategia de mutación es significativamente superior o igual al método de entrenamiento exhaustivo.

Respecto a la complejidad de la capa oculta, los resultados de la Tabla 5.3 muestran que los algoritmos de entrenamiento evolutivo son competitivos con respeto al método exhaustivo y en varios casos mejoran las soluciones, esto es, seleccionan una menor cantidad de nodos ocultos. Las variantes *current-to-best* y *JADE con y sin archivo externo* presentan resultados estadísticamente iguales y en algunos casos superiores al método exhaustivo.

En general, el método exhaustivo produce soluciones con un menor porcentaje de características de entrada, como se muestra en la Tabla 5.4, por ello todas las variantes de evolución diferencial implementadas son estadísticamente inferiores en la mayoría de los experimentos con los diferentes conjuntos de datos.

En la Figura 5.2 se muestra una matriz en donde la primera fila tiene los valores de la distancia de Hamming entre el vector binario de características resultante del experimento uno y los vectores binarios correspondientes a los otros experimentos, y así sucesivamente para cada una de las 31 filas, por lo que este arreglo constituye una matriz cuadrada y simétrica. Los elementos que constituyen la diagonal principal tienen valores iguales a cero, ya que corresponden a la distancia de Hamming entre vectores del mismo experimento.

En la Tabla 5.5 se muestran los resultados de la media y la desviación estándar de los valores de la matriz superior en la Figura 5.2. Las variantes JADE muestran las soluciones más consistentes, ya que la distancia Hamming es más cercana a cero. Esto indica que el conjunto de atributos seleccionado por ambas variantes son similares, además, la estrategia *JADE con archivo externo* tiene más resultados estadísticamente superiores al método exhautivo en comparación con las otras variantes de entrenamiento evolutivo. Por otro lado, las variantes *JADE sin archivo externo* y *current-to-best* son las que siguen con mayor cantidad de soluciones significativamente superiores con respecto al método exhaustivo.

En relación al tiempo de procesamiento, los resultados de la Tabla 5.6 muestran que los algoritmos de entrenamiento evolutivo son más rápidos que el algoritmo de entrenamiento exhaustivo cuando se consideran conjuntos de datos con alta dimensionalidad y grandes cantidades de datos. Un ejemplo de lo anterior se muestra en los resultados experimentales correspondientes al conjunto de datos *Vowel recognition data* con 11 características y 990 patrones, en donde el tiempo de procesamiento

promedio entre los algoritmos de entrenamiento evolutivo es de $1,164$ segundos, mientras que el tiempo promedio del método exhaustivo es de $21,745$ segundos, es decir, el entrenamiento evolutivo fue 18 veces más rápido que el método exhaustivo.

5.4.2 Selección de la variante del algoritmo de evolución diferencial

Como parte de la metodología de desarrollo del proyecto se debe de seleccionar una variante del algoritmo de evolución diferencial de entre las cinco que se utilizaron en la etapa de experimentación. La selección de esta variante se realiza con base en los resultados de la sección anterior mostrados en las Tablas 5.2 a 5.6, donde se observa el desempeño de cada algoritmo de evolución diferencial en relación a los distintos criterios de evaluación, así como la significación estadística de los resultados de cada algoritmo de entrenamiento evolutivo con respecto al algoritmo de entrenamiento exhaustivo. A partir del análisis estadístico de los resultados se concluye que la variante de entrenamiento evolutivo que tuvo un mejor desempeño en la experimentación es la estrategia de mutación *current-to-best*. Esta estrategia muestra los mejores resultados en relación al desempeño de clasificación, por otra parte, respecto a la selección del número de nodos ocultos, los resultados muestran que esta variante es estadísticamente igual y en algunos casos superior al método de entrenamiento exhaustivo. En relación al porcentaje de características seleccionadas, esta estrategia de mutación obtuvo un desempeño similar a las otras cuatro variantes implementadas.

5.5 Análisis de convergencia

Se realiza un análisis de convergencia para definir un criterio de paro y reducir el total de generaciones sin disminuir significativamente el desempeño del algoritmo de entrenamiento evolutivo. En esta tarea se emplea la estrategia de mutación *current-to-best*. Además, se utiliza una condición de paro basada en un umbral, que le indica al algoritmo terminar la ejecución si la diferencia de la función objetivo entre la generación actual g y un determinado número de generaciones anteriores β

es menor que un valor de umbral θ, lo cual se expresa como [7]:

$$\text{Detener si } f(g) - f(g - \beta) < \theta \tag{5.1}$$

5.5.1 Resultados del análisis de convergencia

En el análisis de convergencia se emplean distintos valores de umbral $\theta = \{1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$, y por cada valor de θ se usan los valores de $\beta = \{20, 50, 100\}$. Se realizó una prueba estadística *t-test* con un intervalo de confianza del $95\,\%$ ($\alpha = 0.05$) para corroborar si los resultados del análisis de convergencia mostrados en las Tablas 5.7, 5.8 y 5.9 difieren significativamente de los resultados originales de desempeño de clasificación que consideraron 1000 generaciones. En la primera y segunda columnas se indican los nombres del conjunto de datos y los tres valores de umbral, respectivamente. En la tercera columna se presenta el valor de la hipótesis nula (h_0), donde un valor igual a cero indica que los resultados no son significativamente diferentes y un valor igual a la unidad muestra lo contrario. En la cuarta columna se indica el valor p de significación estadística y en la quinta columna se muestra el promedio de las generaciones de paro de cada experimento.

En la Tabla 5.7 se observa que todos los resultados obtenidos con un valor de $\beta = 20$ son estadísticamente diferentes a los de las pruebas originales. Con respecto a los resultados con un valor de $\beta = 50$, la Tabla 5.8 muestra que la mayoría de los valores obtenidos también son estadísticamente diferentes a los resultados de las pruebas sin condición de paro, solamente en dos conjuntos de datos los valores de la variable de hipótesis son igual a cero para los tres umbrales. En la Tabla 5.9 se observa que la gran mayoría de los resultados de la variable de hipótesis tienen un valor igual a cero por lo que estos resultados obtenidos con un valor de umbral $\beta = 100$ no tienen significación estadística con respecto a los resultados de las pruebas originales.

Con base en los resultados obtenidos con el análisis de convergencia se selecciona un criterio

Conjunto de datos	Umbral (θ)	Hipótesis (h_0)	$p-$value	Generación de paro
Echocardiogram data	1×10^{-3}	1	0.0004	55.64
	1×10^{-4}	1	0.0005	69.83
	1×10^{-5}	1	0.0006	63.48
Glass identification	1×10^{-3}	1	0.0000	73.83
	1×10^{-4}	1	0.0000	69.38
	1×10^{-5}	1	0.0000	103.61
Heart disease cleveland	1×10^{-3}	1	0.0000	74.58
	1×10^{-4}	1	0.0000	89.54
	1×10^{-5}	1	0.0000	99.41
Iris data	1×10^{-3}	1	0.0000	52.96
	1×10^{-4}	1	0.0002	79.51
	1×10^{-5}	1	0.0000	80.51
BUPA liver disorders	1×10^{-3}	1	0.0000	60.87
	1×10^{-4}	1	0.0000	76.35
	1×10^{-5}	1	0.0000	65.90
Pima indians diabetes	1×10^{-3}	1	0.0000	59.74
	1×10^{-4}	1	0.0000	72.48
	1×10^{-5}	1	0.0000	80.61
Seeds	1×10^{-3}	1	0.0000	67.93
	1×10^{-4}	1	0.0000	65.48
	1×10^{-5}	1	0.0003	80.25
Thyroid gland data	1×10^{-3}	1	0.0000	58.35
	1×10^{-4}	1	0.0000	56.35
	1×10^{-5}	1	0.0000	70.51
Vowel recognition data	1×10^{-3}	1	0.0000	52.96
	1×10^{-4}	1	0.0000	68.83
	1×10^{-5}	1	0.0000	52.96
Wine recognition data	1×10^{-3}	1	0.0000	50.03
	1×10^{-4}	1	0.0000	63.03
	1×10^{-5}	1	0.0002	69.06

Tabla 5.7: Resultados del análisis de convergencia para un valor de $\beta = 20$.

Conjunto de datos	Umbral (θ)	Hipótesis (h_0)	$p-$value	Generación de paro
Echocardiogram data	1×10^{-3}	0	0.2472	139.12
	1×10^{-4}	0	0.1053	136.77
	1×10^{-5}	0	0.9980	152.67
Glass identification	1×10^{-3}	1	0.0001	230.19
	1×10^{-4}	1	0.0003	322.48
	1×10^{-5}	1	0.0022	359.19
Heart disease cleveland	1×10^{-3}	1	0.0008	178.67
	1×10^{-4}	1	0.0425	268.90
	1×10^{-5}	1	0.0269	270.61
Iris data	1×10^{-3}	1	0.0358	132.29
	1×10^{-4}	1	0.0300	162.77
	1×10^{-5}	1	0.0018	174.03
BUPA liver disorders	1×10^{-3}	1	0.0000	235.45
	1×10^{-4}	1	0.0007	313.38
	1×10^{-5}	1	0.0005	319.32
Pima indians diabetes	1×10^{-3}	1	0.0005	213.25
	1×10^{-4}	1	0.0091	279.96
	1×10^{-5}	1	0.0003	285.12
Seeds	1×10^{-3}	0	0.4506	146.93
	1×10^{-4}	0	0.1021	172.41
	1×10^{-5}	0	0.2427	194.74
Thyroid gland data	1×10^{-3}	1	0.0005	158.64
	1×10^{-4}	1	0.0005	229.00
	1×10^{-5}	0	0.2495	308.45
Vowel recognition data	1×10^{-3}	1	0.0000	304.32
	1×10^{-4}	1	0.0000	348.45
	1×10^{-5}	1	0.0000	419.90
Wine recognition data	1×10^{-3}	0	0.0501	132.87
	1×10^{-4}	1	0.0187	177.58
	1×10^{-5}	0	0.0622	198.06

Tabla 5.8: Resultados del análisis de convergencia para un valor de $\beta = 50$.

Conjunto de datos	Umbral (θ)	Hipótesis (h_0)	$p-$value	Generación de paro
Echocardiogram data	1×10^{-3}	0	0.5409	206.74
	1×10^{-4}	0	0.5964	212.38
	1×10^{-5}	0	0.3132	200.51
Glass identification	1×10^{-3}	1	0.0163	414.54
	1×10^{-4}	0	0.6442	630.80
	1×10^{-5}	0	0.6207	705.35
Heart disease cleveland	1×10^{-3}	0	0.4279	307.74
	1×10^{-4}	0	0.2907	433.70
	1×10^{-5}	0	0.1446	458.58
Iris data	1×10^{-3}	0	0.8582	231.90
	1×10^{-4}	0	0.3702	260.77
	1×10^{-5}	0	0.1913	264.77
BUPA liver disorders	1×10^{-3}	0	0.2299	457.70
	1×10^{-4}	0	0.3319	582.03
	1×10^{-5}	0	0.2804	648.03
Pima indians diabetes	1×10^{-3}	0	0.0646	359.00
	1×10^{-4}	0	0.7452	529.38
	1×10^{-5}	0	0.8281	587.67
Seeds	1×10^{-3}	0	0.1899	210.22
	1×10^{-4}	0	0.9516	258.61
	1×10^{-5}	0	0.1465	308.19
Thyroid gland data	1×10^{-3}	0	0.1512	287.41
	1×10^{-4}	0	0.1997	392.54
	1×10^{-5}	0	0.4847	447.54
Vowel recognition data	1×10^{-3}	1	0.0053	559.25
	1×10^{-4}	0	0.1473	824.25
	1×10^{-5}	0	0.2833	827.38
Wine recognition data	1×10^{-3}	0	0.5631	198.16
	1×10^{-4}	0	0.6230	259.00
	1×10^{-5}	0	0.3823	271.83

Tabla 5.9: Resultados del análisis de convergencia para un valor de $\beta = 100$.

de paro en donde se consideran valores de umbral (θ) y de generaciones anteriores (β) adecuados para reducir el número de generaciones del entrenamiento evolutivo sin afectar significativamente el desempeño de clasificación del algoritmo. Los resultados obtenidos con $\beta = 100$, correspondientes a los mostrados en la Tabla 5.9, son los que mostraron no tener significación estadística con respecto a los resultados originales en la gran mayoría de los casos. En relación a los valores de umbral se observa que aunque un valor de $\theta = 1 \times 10^{-3}$ reduce notablemente el número de generaciones, los resultados muestran que en dos ocasiones los valores fueron significativamente diferentes; sin embargo, también se observa que no existe significación estadística con respecto a los valores de las pruebas originales para los umbrales $\theta = 1 \times 10^{-4}$ y $\theta = 1 \times 10^{-5}$, por lo que se selecciona el valor de umbral que muestra una mayor disminución en las generaciones, esto es $\theta = 1 \times 10^{-4}$.

Finalmente el criterio de paro queda definido de la siguiente manera:

$$\text{Detener si } f(g) - f(g - 100) < 1 \times 10^{-4} \tag{5.2}$$

donde g es la generación actual.

5.6 Comparativa de selección de características entre algoritmos de entrenamiento

Debido a que el algoritmo propuesto utiliza el criterio $MRMR$ para realizar la selección de atributos, es importante determinar la cercanía al máximo valor posible de $MRMR$ para conjuntos de datos con alta dimensionalidad. Por lo anterior, se lleva a cabo una medición exhaustiva del índice $MRMR$ de los conjuntos de datos mostrados en la Tabla 5.10, cuya dimensionalidad es mayor a los conjuntos de datos mostrados en la Tabla 5.1.

Conjunto de datos	Atributos	Clases	Instancias
Climate model simulation crashes	18	2	540
Hepatitis domain	18	2	112
Statlog (image segmentation)	18	7	2310
Vehicle silhouettes	18	4	846
Diabetic retinopathy debrecen	19	2	1151
Parkinsons data set	22	2	195

Tabla 5.10: Descripción de los conjuntos de datos.

El algoritmo de entrenamiento evolutivo utilizó la estrategia de mutación *current-to-best* y se realizaron 31 ejecuciones independientes. Por otra parte, la medición exhaustiva del índice $MRMR$ se realizó una vez debido a que se trata de un experimento determinista. En las Figuras 5.3 a 5.8 se muestran las curvas de los valores del índice $MRMR$ correspondientes a las diferentes combinaciones de atributos ordenadas según su relación de relevancia/redundancia, y la línea punteada indica la posición de la combinación de atributos obtenida con el entrenamiento evolutivo y su respectivo valor del índice $MRMR$ en la curva. Nótese que la última posición de la curva en cada gráfica indica el máximo valor posible de $MRMR$ obtenido para una combinación específica de características. Por tanto, se busca que la solución generada por el algoritmo propuesto se encuentre cercana a la combinación que obtiene el máximo valor de $MRMR$.

Los resultados muestran que en general el algoritmo de entrenamiento evolutivo selecciona conjuntos de características similares al mejor conjunto obtenido con la medición exhaustiva, esto considerando la cantidad de atributos seleccionados por ambos enfoques y la distancia Hamming entre los vectores binarios de características.

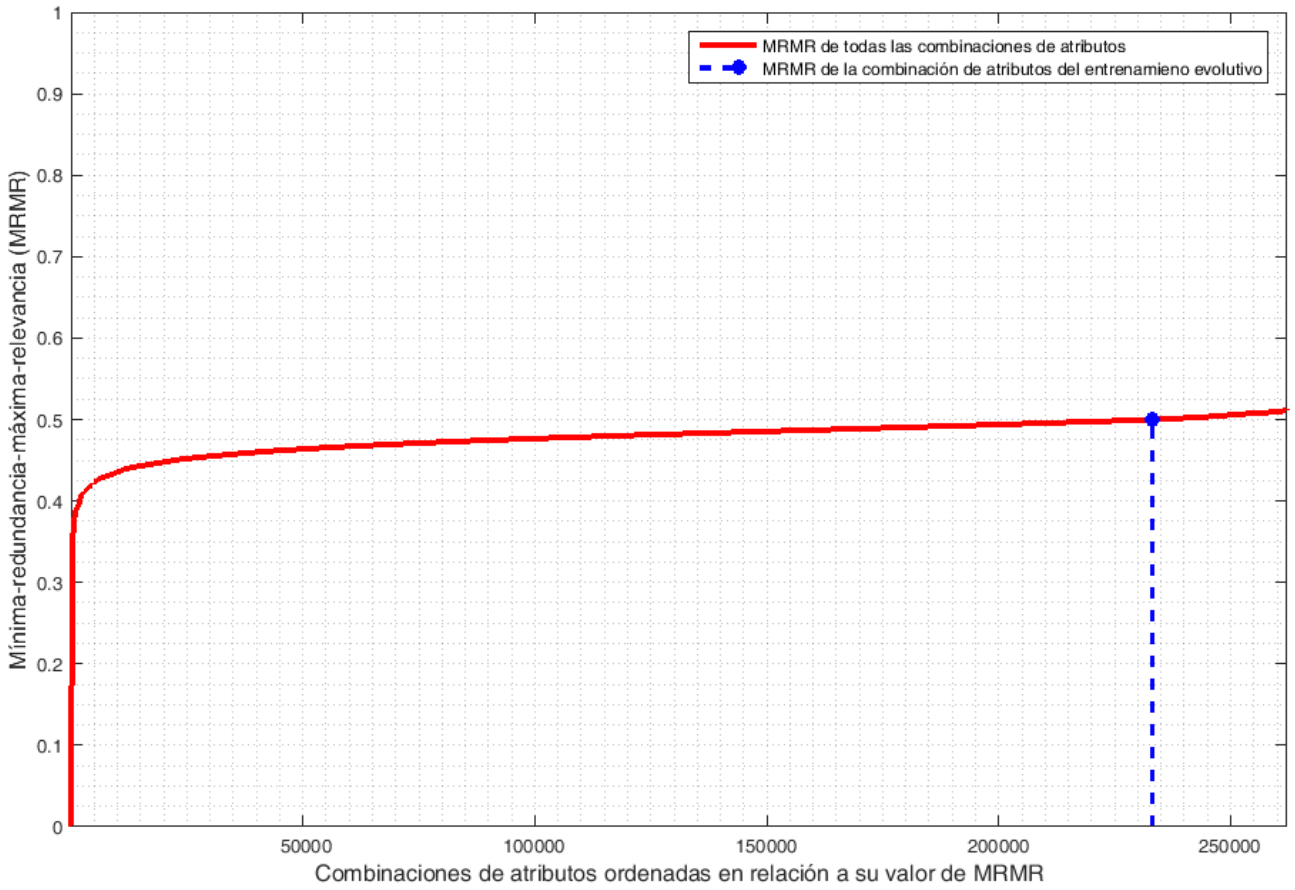


Figura 5.3: Curva de valores del índice $MRMR$ correspondiente al conjunto de datos *Climate model simulation crashes* con 18 características $(262,143$ combinaciones). El conjunto de atributos que maximiza la relación de relevancia y redundancia de acuerdo a la medición exhaustiva es $\{1,2,4,5,6,9,12,13,14,16,17\}$, mientras que el conjunto de atributos seleccionado por el entrenamiento evolutivo indicado en color azul es $\{1,2,3,4,6,7,9,12,13,15,16,17\}$, y la distancia Hamming entre sus respectivos vectores binarios de características es 5.

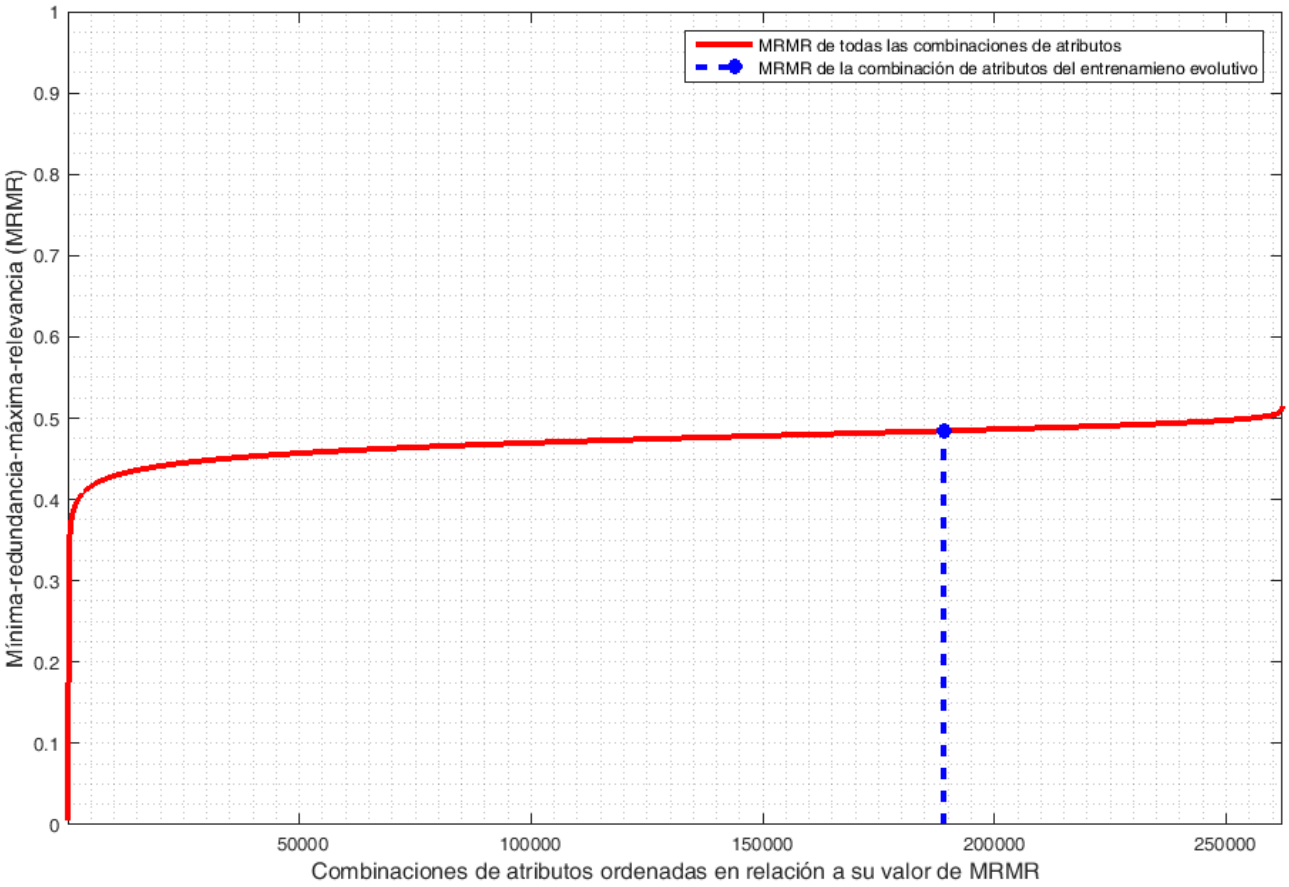


Figura 5.4: Curva de valores del índice $MRMR$ correspondiente al conjunto de datos *Hepatitis domain* con 18 características ($262,143$ combinaciones). El conjunto de atributos que maximiza la relación de relevancia y redundancia de acuerdo a la medición exhaustiva es $\{1, 2, 6, 8, 10, 12, 13, 14, 17, 18\}$, mientras que el conjunto de atributos seleccionados por el entrenamiento evolutivo indicado en color azul es $\{2, 4, 8, 10, 12, 13, 14, 15, 17, 18\}$, y la distancia Hamming entre sus respectivos vectores binarios de características es 4.

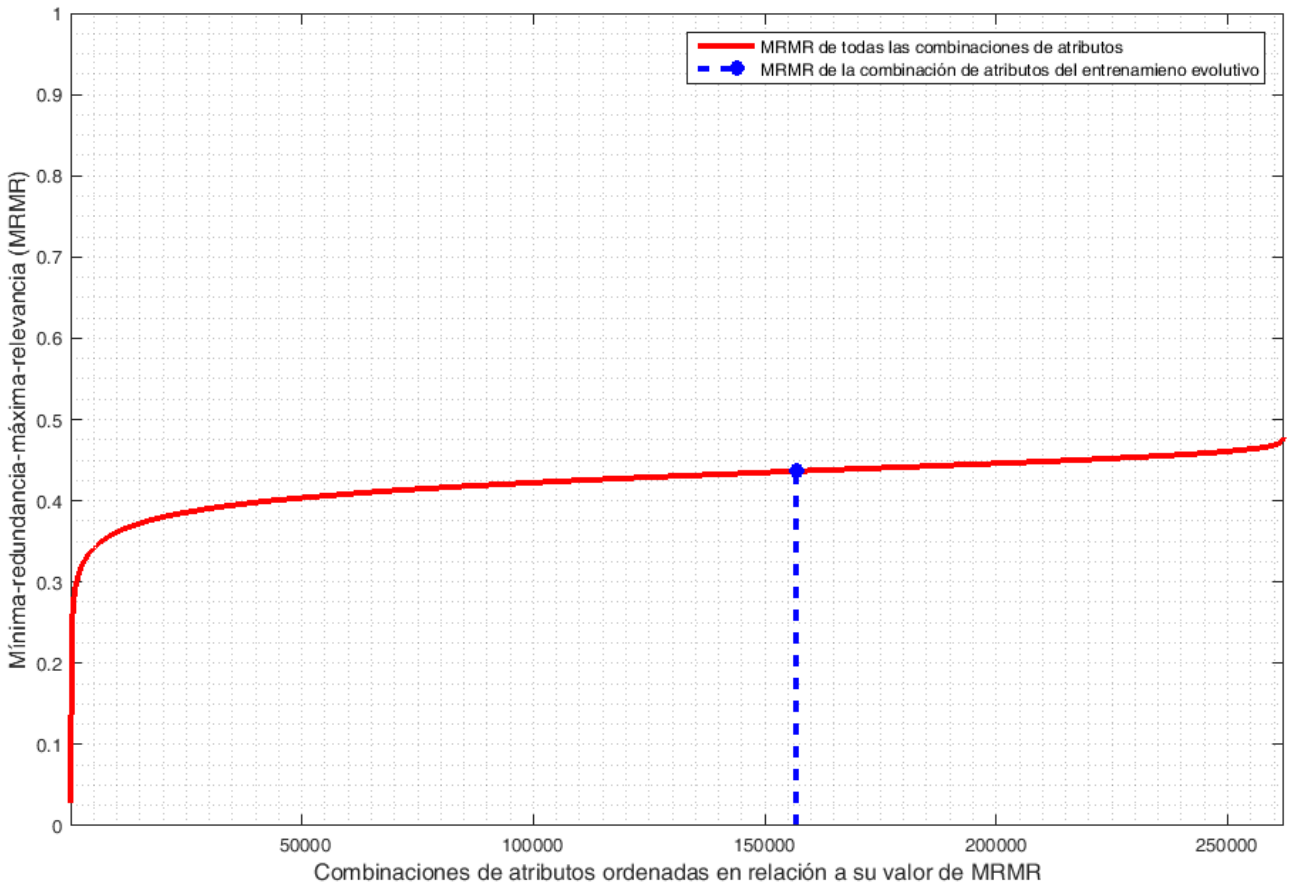


Figura 5.5: Curva de valores del índice $MRMR$ correspondiente al conjunto de datos *Statlog (image segmentation)* con 18 características $(262,143$ combinaciones). El conjunto de atributos que maximiza la relación de relevancia y redundancia de acuerdo a la medición exhaustiva es $\{1,2,3,4,5,8,12,15,17\}$, mientras que el conjunto de atributos seleccionados por el entrenamiento evolutivo indicado en color azul es $\{1,2,3,4,6,7,9,10,12,15\}$, y la distancia Hamming entre sus respectivos vectores binarios de características es 7.

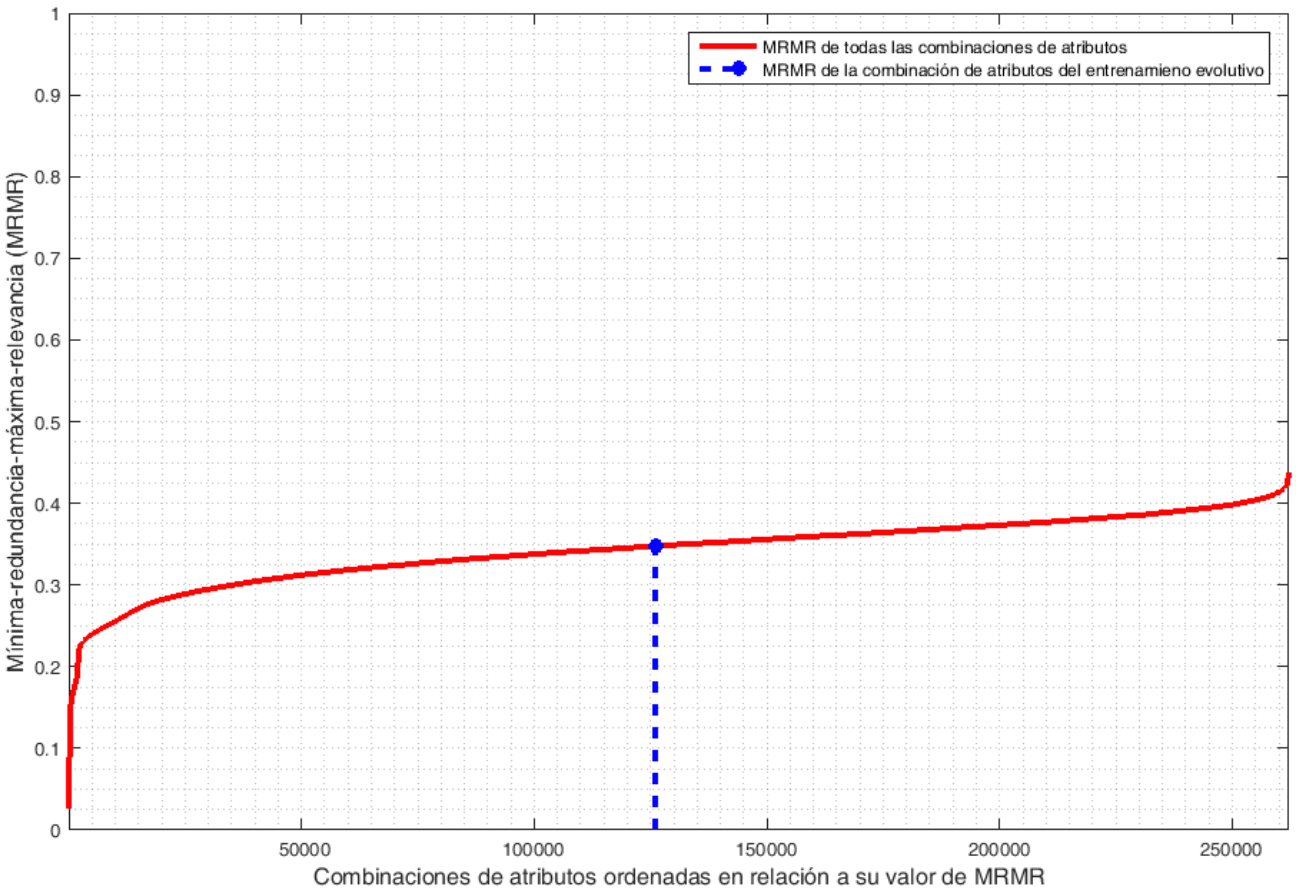


Figura 5.6: Curva de valores del índice $MRMR$ correspondiente al conjunto de datos *Vehicle silhouettes* con 18 características $(262,143$ combinaciones). El conjunto de atributos que maximiza la relación de relevancia y redundancia de acuerdo a la medición exhaustiva es $\{5, 12, 14, 15, 16\}$, mientras que el conjunto de atributos seleccionados por el entrenamiento evolutivo indicado en color azul es $\{1, 4, 5, 8, 10, 14, 15, 16, 17\}$, y la distancia Hamming entre sus respectivos vectores binarios de características es 6.

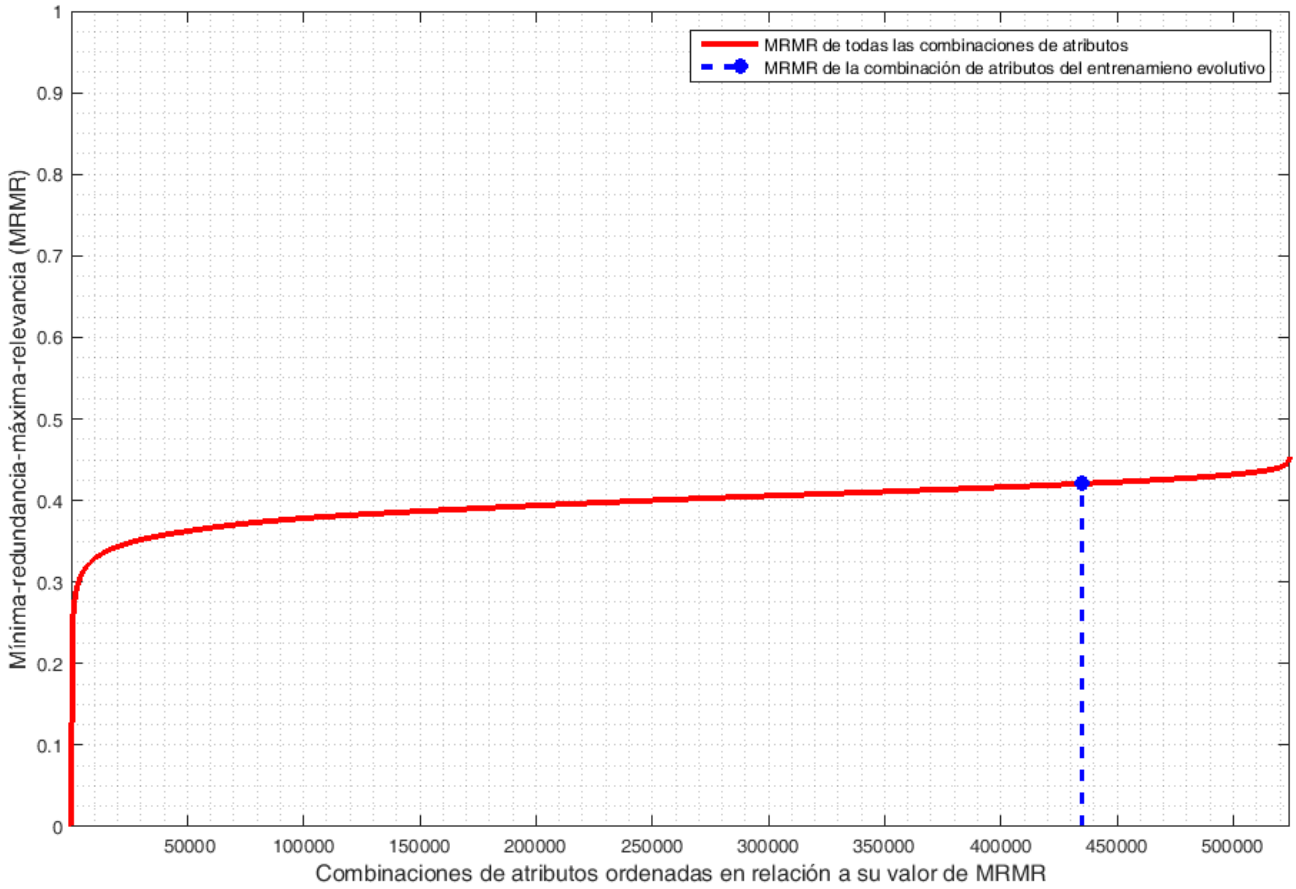


Figura 5.7: Curva de valores del índice $MRMR$ correspondiente al conjunto de datos *Diabetic retinopathy debrecen* con 19 características ($524,287$ combinaciones). El conjunto de atributos que maximiza la relación de relevancia y redundancia de acuerdo a la medición exhaustiva es $\{1, 2, 3, 4, 11, 16, 17, 18\}$, mientras que el conjunto de atributos seleccionados por el entrenamiento evolutivo indicado en color azul es $\{1, 2, 3, 4, 6, 8, 10, 11, 13, 14, 16, 18\}$, y la distancia Hamming entre sus respectivos vectores binarios de características es 6.

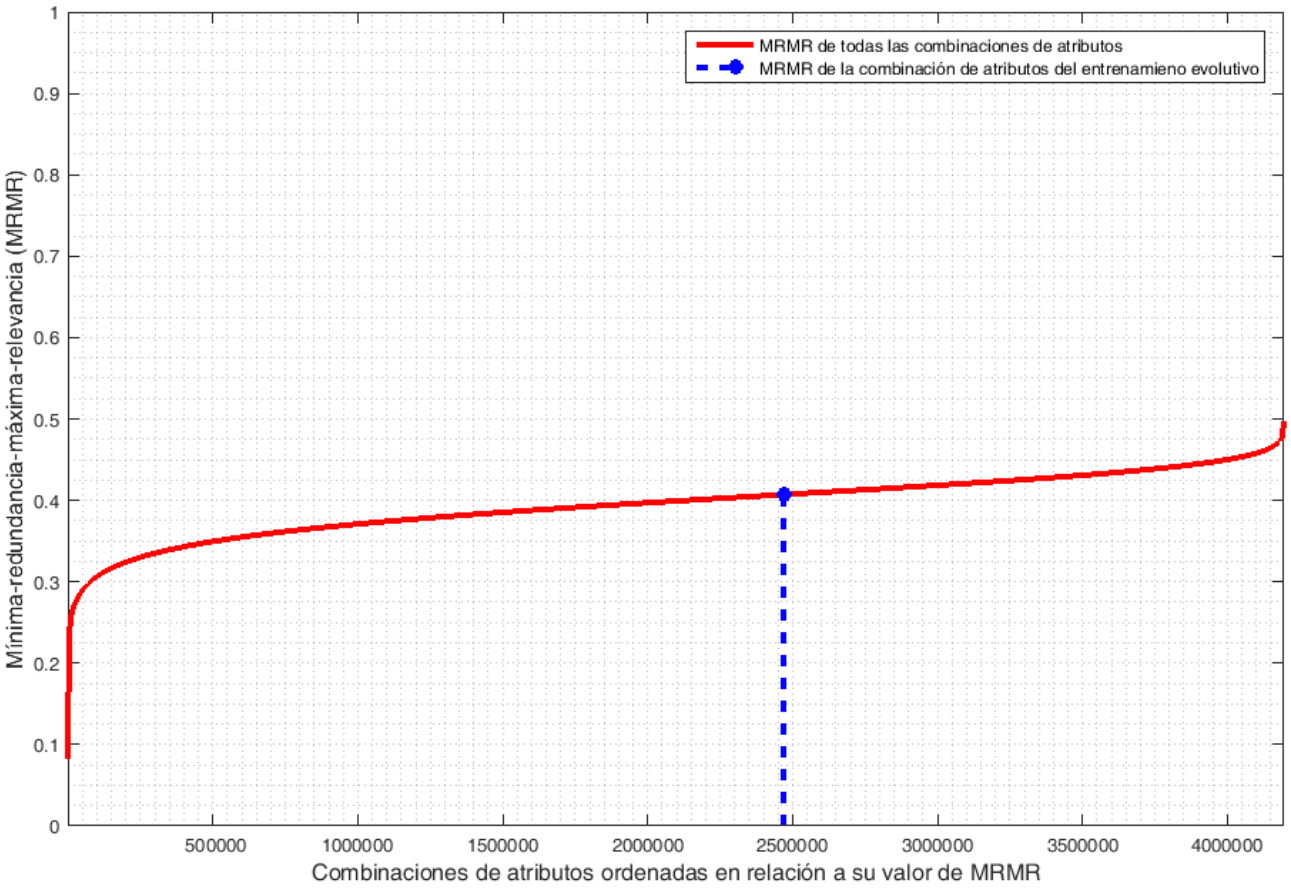


Figura 5.8: Curva de valores del índice $MRMR$ correspondiente al conjunto de datos *Parkinsons data set* con 22 características ($4,194,303$ combinaciones). El conjunto de atributos que maximiza la relación de relevancia y redundancia de acuerdo a la medición exhaustiva es $\{2,3,16,18,19,20\}$, mientras que el conjunto de atributos seleccionados por el entrenamiento evolutivo indicado en color azul es $\{1,2,3,4,5,11,14,16,17,18,19,22\}$, y la distancia Hamming entre sus respectivos vectores binarios de características es 8.

5.7 Sumario

En este capítulo se mostraron los detalles de la experimentación y los resultados obtenidos por los algoritmos de entrenamiento evolutivo y exhaustivo. Se realizó la comparación del desempeño de ambos algoritmos de entrenamiento empleando un análisis estadístico ANOVA de un factor con un intervalo de confianza del $95\,\%$. Los resultados mostraron que el desempeño de la arquitectura de RBFN obtenida por el método de entrenamiento evolutivo es competitiva y en algunos casos mejora estadísticamente al desempeño de las arquitecturas conseguidas por el método de entrenamiento exhaustivo. También se realizó un análisis de convergencia y se definió un criterio de paro con base en dicho análisis. Finalmente se hizo una comparativa del desempeño de ambos enfoques considerando solamente los resultados de la selección de características empleando conjuntos de datos con un mayor número de atributos en relación a los utilizados en el análisis estadístico, en esta comparativa el algoritmo de entrenamiento evolutivo tuvo resultados competitivos al seleccionar conjuntos de atributos similares a la medición exhaustiva.

6

Conclusiones y trabajo futuro

En el presente capítulo se muestran las conclusiones de este trabajo de tesis, se discuten las principales contribuciones realizadas y se describe el trabajo futuro para complementar esta investigación.

6.1 Conclusiones

En el diseño de sistemas de clasificación, las etapas de selección de características y diseño del clasificador comúnmente se tratan por separado. En la construcción de la arquitectura de una RBFN, la selección de características se relaciona con la capa de entrada, el diseño del clasificador se vincula con la capa oculta, y el desempeño de clasificación está relacionado con la capa de salida.

La mayoría de los trabajos publicados en la literatura relacionados al entrenamiento de la arquitectura de una RBFN empleando algoritmos evolutivos consideran solamente la etapa de diseño del clasificador. En estos enfoques normalmente se emplea el error cuadrático medio para medir el

error de clasificación, y para evaluar el desempeño de la capa oculta únicamente consideran métricas relacionadas a la cantidad de nodos ocultos utilizados.

En este trabajo de tesis se propuso un algoritmo de entrenamiento basado en evolución diferencial para construir la arquitectura completa de una RBFN considerando simultáneamente los problemas de selección de características y diseño del clasificador. El desempeño del algoritmo propuesto fue comparado con un algoritmo de entrenamiento exhaustivo que evalúa todas las posibles combinaciones de características y nodos ocultos de una RBFN con el propósito de obtener la mejor arquitectura de red.

El desempeño de las soluciones de los algoritmos se han medido en relación a tres criterios principales, los cuales son: relación de relevancia y redundancia de las características de la capa de entrada, eficiencia de la capa oculta en relación a la cantidad de RBF y su respectiva distribución en el espacio de características, y desempeño de clasificación de la capa de salida.

En la etapa de experimentación con el algoritmo de entrenamiento evolutivo se utilizaron cinco estrategias de mutación: *rand/1/bin*, *best/1*, *current-to-best*, *current-to-$p$best* con archivo externo y sin archivo externo. Los resultados experimentales muestran que el algoritmo propuesto es competitivo y en algunos casos mejora las soluciones del algoritmo de entrenamiento exhaustivo. En términos de desempeño de clasificación, la variante *current-to-best* presenta un mejor rendimiento en relación a las otras estrategias de mutación. Con respecto a la cantidad de nodos ocultos elegidos, las variantes *current-to-best*, *current-to-$p$best* con archivo externo y sin archivo externo muestran resultados estadísticamente similares entre sí y además presentan un mejor rendimiento con respecto a las otras dos estrategias de mutación, esto es, seleccionan una cantidad menor de nodos ocultos. Finalmente, en términos de porcentaje de características seleccionadas, el algoritmo de entrenamiento exhaustivo genera mejores soluciones que las variantes del algoritmo de entrenamiento evolutivo, esto significa

que el método exhaustivo reduce el número de atributos seleccionados para formar parte de la tarea de clasificación. No obstante, esto no demerita el desempeño del enfoque propuesto, ya que la métrica $MRMR$, empleada para la evaluación de la selección de características maximiza la relación de relevancia y redundancia de las mismas y no considera disminuir la cantidad de atributos. De esta manera se evita reducir características de manera obligatoria, ya que puede haber casos en los que el conjunto completo de atributos es indispensable para mantener el desempeño de clasificación, de modo que si se eliminan atributos el desempeño de clasificación disminuiría. Un ejemplo de este comportamiento se observó con el conjunto *Thyroid gland data*, en el cual prácticamente se necesitan todas las características para maximizar el desempeño de clasificación.

6.2 Contribuciones

La principal contribución del presente trabajo de tesis es un algoritmo de entrenamiento de la arquitectura de una RBFN empleando evolución diferencial (Algoritmo 4.1), el cual considera simultáneamente los problemas de selección de características y diseño del clasificador. Para esto se propuso un algoritmo de entrenamiento evolutivo en donde se emplea una representación de la solución que codifica los nodos de entrada y ocultos de la red y una función objetivo ponderada que evalúa el desempeño de cada una de las capas de la RBFN.

Otra contribución al estado del arte es el índice de eficiencia (E, Ecuación 4.8), el cual es propuesto para evaluar el desempeño de la capa oculta no sólo en relación a la cantidad de nodos ocultos, sino que también permite cuantificar la calidad de la distribución de las RBF en el espacio de características.

Finalmente, como parte de las contribuciones realizadas, se realizó un artículo de conferencia intitulado *"Automatic Construction of the Complete Architecture of a Radial Basis Function Network*

Using Differential Evolution", que presenta la metodología propuesta y los resultados obtenidos, el cual fue enviado al congreso *"2017 14th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE 2017)*".

6.3 Trabajo futuro

En este trabajo se consideró una función de costo ponderada con coeficientes iguales para transformar el problema de optimización de múltiples objetivos en un problema de optimización de un sólo objetivo. El uso de coeficientes iguales para todas las funciones objetivo indica que poseen la misma importancia; sin embargo, no se exploró el comportamiento del algoritmo al darle diferente importancia relativa a cada objetivo. Por tanto, en el trabajo futuro se considera la multi-objetivización del algoritmo propuesto, optimizando simultáneamente los tres criterios considerados en la función objetivo propuesta para obtener soluciones óptimas de Pareto.

Bibliografía

[1] Bajer, D., Zorić, B., and Martinović, G. (2016). Effectiveness of differential evolution in training radial basis function networks for classification. In *2016 International Conference on Smart Systems and Technologies (SST)*, pages 179–184.

[2] Bauer, M., Buchtala, O., Horeis, T., Kern, R., Sick, B., and Wagner, R. (2005). Process identification and quality control with evolutionary optimized rbf classifiers. In *Proceedings of the 2005 IEEE Midnight-Summer Workshop on Soft Computing in Industrial Applications, 2005. SMCia/05.*, pages 178–183.

[3] Bebis, G. and Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31.

[4] Benoudjit, N. and Verleysen, M. (2003). On the kernel widths in radial-basis function networks. *Neural Processing Letters*, 18(2):139–154.

[5] Broomhead, D. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems Publications*.

[6] Buchtala, O., Klimek, M., and Sick, B. (2005). Evolutionary optimization of radial basis function classifiers for data mining applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(5):928–947.

[7] Chakraborty, U. (2008). *Advances in differential evolution*. Springer Verlag, Berlin.

[8] Dhahri, H. and Alimi, A. M. (2006). The modified differential evolution and the rbf (mde-rbf) neural network for time series prediction. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 2938–2943.

[9] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10(7):1895–1923.

[10] Ding, C. and Peng, H. (2005). Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 3(2):185–205.

[11] Ding, S., Xu, L., Su, C., and Jin, F. (2012). An optimizing method of rbf neural network based on genetic algorithm. *Neural Computing and Applications*, 21(2):333–336.

[12] Engelbrecht, A. (2007). *Computational intelligence : an introduction*. John Wiley & Sons, Chichester, England Hoboken, NJ.

[13] Gorodkin, J. (2004). Comparing two k-category assignments by a k-category correlation coefficient. *Comput. Biol. Chem.*, 28(5-6):367–374.

[14] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*.

[15] Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160.

[16] Huang, Y. C., Chen, S. J., and Huang, C. M. (2011). Evolving radial basic function neural network for fast restoration of distribution systems. In *2011 6th IEEE Conference on Industrial Electronics and Applications*, pages 401–406.

[17] Izenman, A. J. (2013). Linear discriminant analysis. In *Springer Texts in Statistics*, pages 237–280. Springer New York.

[18] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *1995 IEEE International Conference on Neural Networks Proceedings*, volume 4, pages 1942–1948 vol.4.

[19] Kilmek, M. and Sick, B. (2003). Architecture optimization of radial basis function networks with a combination of hard- and soft-computing techniques. In *2003 IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 4664–4671 vol.5.

[20] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[21] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324.

[22] Kruskal, W. H. and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621.

[23] Lichman, M. (2013). UCI machine learning repository.

[24] Marhon, S. A., Cameron, C. J. F., and Kremer, S. C. (2013). Recurrent neural networks. In *Intelligent Systems Reference Library*, pages 29–65. Springer Berlin Heidelberg.

[25] Meyer, C. D. and Painter, R. J. (1970). Note on a least squares inverse for a matrix. *J. ACM*, 17(1):110–112.

[26] Mezura-Montes, E. n., Velázquez-Reyes, J., and Coello Coello, C. A. (2006). A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 485–492, New York, NY, USA. ACM.

[27] Montero-Hernandez, S. and Gomez-Flores, W. (2014). Evolutionary approach for construction of the rbf network architecture. In *2014 13th Mexican International Conference on Artificial Intelligence (MICAI)*, pages 121–127.

[28] Moody, J. and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Comput.*, 1(2):281–294.

[29] O'Hora, B., Perera, J., and Brabazon, A. (2006). Designing radial basis function networks for classification using differential evolution. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 2932–2937.

[30] Pakhira, M. K. (2009). A modified k-means algorithm to avoid empty clusters. *International Journal of Recent Trends in Engineering*, pages 220–226.

[31] Pal, N. R. and Bezdek, J. C. (1995). On cluster validity for the fuzzy c-means model. *Trans. Fuz Sys.*, 3(3):370–379.

[32] Penrose, R. (1955). A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406 413.

[33] Price, K., Storn, R. M., and Lampinen, J. A. (2005). *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[34] Priddy, K. L. and Keller, P. E. (2005). *Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering, Vol. TT68)*. SPIE- International Society for Optical Engineering.

[35] Qing, A. (2010). *Basics of Differential Evolution*, pages 19–42. Springer Berlin Heidelberg, Berlin, Heidelberg.

[36] Rao, S. S. (2009). *Engineering Optimization: Theory and Practice*. John Wiley & Sons, fourth edition edition.

[37] Rich, E. and Knight, K. (1990). *Artificial Intelligence*. McGraw-Hill Higher Education, 2nd edition.

[38] Rish, I. (2001). An empirical study of the naive bayes classifier. Technical report.

[39] Schwenker, F., Kestler, H. A., and Palm, G. (2001). Three learning phases for radial-basis-function networks. *Neural Networks*, 14(4):439 – 458.

[40] Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

[41] Tate, R. F. (1954). Correlation between a discrete and a continuous variable. point-biserial correlation. *Ann. Math. Statist.*, 25(3):603–607.

[42] Theodoridis, S. and Koutroumbas, K. (2008). *Pattern Recognition*. Elsevier Science.

[43] Vesterstrom, J. and Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Congress on Evolutionary Computation, 2004. CEC2004.*, volume 2, pages 1980–1987 Vol.2.

[44] Weiss, G. (1994). Neural networks and evolutionary computation. part ii: hybrid approaches in the neurosciences. In *1994 IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 273–277 vol.1.

[45] Xue, X., Lu, J., and Xiang, W. (2012). Nonlinear system identification with modified differential evolution and rbf networks. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*, pages 332–335.

[46] Yu, B. and He, X. (2006). Training radial basis function networks with differential evolution. In *2006 IEEE International Conference on Granular Computing*, pages 369–372.

[47] Yu, S. and Zhu, K. (2009). A hybrid mga-bp algorithm for rbfns self-generate. In *2009 IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009.*, pages 4304–4308.

[48] Zhang, A. (2006). *Advanced Analysis of Gene Expression Microarray Data*. World Scientific Publishing Company.

[49] Zhang, J. and Sanderson, A. C. (2009). Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958.

[50] Zhang, Y., Feng, Y., Wu, D., and Hou, C. (2010). The optimization of radial basis function network based on chaos immune genetic algorithm. In *2010 International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 506–511.

[51] Zhou, J., Wen, Z., and Qu, J. (2010). Rbf neural network using improved differential evolution for groundwater table prediction. In *2010 International Conference on Computational Intelligence and Software Engineering (CiSE)*, pages 1–4.

[52] Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition.