Universidad de las Ciencias Informáticas

FACULTAD 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

ALGORITMO KORA PARA LA SOLUCIÓN DE PROBLEMAS DE CLASIFICACIÓN SUPERVISADA COMO EXTENSIÓN DE LA HERRAMIENTA CEPAR

Autor:

Eliani Cabrera García

Tutores:

M. Sc. Maidelis Milanés Luque

Ing. Tania García González

La Habana, junio de 2018

"Año 60 de la Revolución"

"Es que, cuando los hombres llevan en la mente un mismo ideal, nada puede incomunicarlos, ni las paredes de una cárcel, ni la tierra de los cementerios, porque un mismo recuerdo, una misma alma, una misma idea, una misma conciencia y dignidad los alienta a todos."

DECLARACIÓN DE AUTORÍA

Se declara que la estudiante Eliani Cabrera García es la única autora de la presente tesis titulada: “Algoritmo KORA para la solución de problemas de clasificación supervisada como extensión de la herramienta CEPAR” y se le concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales, con carácter exclusivo.

Para que así conste se firma a los 12 días del mes de junio del año 2018.

_____

Eliani Cabrera García
Firma del Autor

_____ _____

M. Sc. Maidelis Milanés Luque
Firma del Tutor

Ing. Tania García González
Firma del Tutor

DATOS DE CONTACTO

Datos del Autor:

Eliani Cabrera García

Universidad de las Ciencias Informáticas, La Habana, Cuba.

e-mail: egarcia@estudiantes.uci.cu

Datos del Tutor:

M. Sc. Maidelis Milanés Luque: graduada de Ingeniería en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas en el año 2007. Pertenece al claustro de profesores de la Facultad 2 y ocupa el cargo de Jefa de Departamento de Técnicas de Programación y Sistemas Digitales. Obtiene el grado científico de Máster en Ciencias en el año 2017.

e-mail: mmilanes@uci.cu

Datos del Tutor:

Ing. Tania Garcia González: graduada de Ingeniería en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas en el año 2015. Pertenece al Centro de Informática Médica (CESIM) y ocupa el rol de desarrolladora.

e-mail: tgonzalez@uci.cu

AGRADECIMIENTOS

En primer lugar, agradezco a Fidel y a esta obra tan grande que nos legó que es la Revolución, que permitió que pudiera recorrer el camino de la educación hasta llegar aquí, a este momento en que me gradúo como Ingeniera en Ciencias Informáticas en esta casa de altos estudios.

A mi familia por el tiempo que no he pasado con ellos y aun así me han comprendido, por las horas de desvelo y las palabras de aliento, por apoyarme en todas las decisiones que he tomado y por ser guía y ejemplo en mi vida.

A mis mejores amigos gracias por su preocupación y por cuidar de mí, aún y cuando no me lo merecía, gracias por abrirme las puertas de su corazón y albergar a este ser imperfecto que ha sido mucho mejor gracias a sus enseñanzas, a sus alegrías y tristezas, a sus locuras y sinceridad, a todo el amor que me han brindado.

A mi amor por aceptarme y comprender como soy, por toda la preocupación y el amor demostrado, por darme la satisfacción de amar y ser amada.

A mis profesores, a los que me enseñaron todo lo que sabían, a los que me educaron, a los que me aconsejaron, a los que de una forma u otra han aportado su granito de arena en mi formación personal y profesional.

A mis compañeros de aula por cubrirme las espaldas en los momentos que lo necesitaba, porque con ellos tenía el reto de ser mejor estudiante y mejor persona cada día en el aula.

A la FEU por ponerme situaciones difíciles y retos a diario, que hicieron que diera lo mejor de mí y que fuera más fuerte, por ser otra gran escuela que sin lugar a dudas me preparó para ser una persona más íntegra y justa.

A mis tutores por el tiempo que me dedicaron, incluso hasta en las noches, en momentos en que tenían grandes compromisos de trabajo siempre tuvieron para mi ese tiempo para ayudarme a realizar esta tesis.

A todos no me alcanzará la vida para agradecerles lo que han hecho por mi… Gracias!

DEDICATORIA

A mi mamita y mi papito por enseñarme que en la vida no se puede dejar de estudiar y aprender,

por inculcarme los deseos de la superación constante y

por demostrarme que el campeón siempre es parte de la solución.

RESUMEN

El Reconocimiento Lógico Combinatorio de Patrones, es uno de los enfoques de Reconocimiento de Patrones que permite el trabajo simultáneo con variables cuantitativas y cualitativas. En la actualidad existe el Entorno Cubano para el Reconocimiento Lógico Combinatorio de Patrones (por sus siglas en inglés, CEPAR) desarrollado por el grupo de investigación de Inteligencia Artificial y Reconocimiento de Patrones de la Universidad de las Ciencias Informáticas. La herramienta CEPAR para la solución de problemas de clasificación supervisada sólo cuenta con un algoritmo desarrollado, lo que restringe las formas y vías de solucionar este tipo de problemas; limitando la utilización y extensión de la herramienta en investigaciones y en la docencia.

El objetivo de la presente investigación fue implementar el algoritmo KORA para la solución de problemas de clasificación supervisada como extensión de la herramienta CEPAR. Para el desarrollo de la investigación se utilizaron los métodos de investigación analítico-sintético, histórico, la modelación, el sistémico, la entrevista, análisis documental y análisis comparativo; así como las herramientas NetBeans y Visual Paradigm.

Se obtuvo como resultado la adición a la herramienta CEPAR el algoritmo KORA-3 y su extensión el KORA-Ω. La incorporación de los algoritmos en la herramienta permite su reutilización en las labores cotidianas de los investigadores, estudiantes y docentes que trabajan con el enfoque Reconocimiento Lógico Combinatorio de Patrones, brindándoles un espectro más amplio de formas y posibilidades de resolver un problema específico.

Palabras clave: clasificación supervisada, herramienta CEPAR, KORA, Reconocimiento Lógico Combinatorio de Patrones.

ABSTRACT

The Logical Combinatorial Patterns Recognition, is one of the Patterns Recognition approaches that allows the simultaneous work with quantitative and qualitative variables. At present, there is the *Cuban Environment for Logical Combinatorial Pattern Recognition* (CEPAR) developed by the Artificial Intelligence and Pattern Recognition research group of the University of Informatics Sciences. The CEPAR tool for solving supervised classification problems only has a developed algorithm that restricts the ways and means of solving this type of problem; limiting the use and extension of the tool in research and teaching.

The objective of the present investigation was to implement the KORA algorithm for the solution of supervised classification problems as an extension of the CEPAR tool. For the development of the research methods were used analytical-synthetic, historical, modeling, systemic, interview, documentary analysis and comparative analysis; and the NetBeans and Visual Paradigm tools.

The result was the addition to the CEPAR tool of the KORA-3 algorithm and its KORA-Ω extension. The incorporation of the algorithms in the tool allows its reuse in the daily tasks of researchers, students and teachers who work with the Combinatorial Logical Recognition of Patterns approach, giving them a wider spectrum of ways and possibilities to solve a specific problem.

Keywords: CEPAR tool, KORA, Logical Combinatorial Patterns Recognition, supervised classification.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los seres humanos realizan diariamente tareas complejas en fracciones de segundos, sin detenerse a pensar siquiera en cómo lo hacen. El reconocimiento de un rostro familiar, la identificación de la voz ante una llamada telefónica, la comprensión de textos, la formación de grupos de clientes, estudiantes, pacientes, así como la detección de enfermedades, son sólo algunos ejemplos de estas tareas.

Una vez que un objeto es conocido, el cerebro tiene la capacidad de reconocerlo posteriormente. El proceso de reconocimiento ocurre cuando, ante la presencia de un nuevo elemento, el conjunto de atributos que se perciben en él, resulta notoriamente semejante al conjunto de atributos del elemento previamente conocido. Ante esas condiciones la mente asocia al nuevo elemento con él o los anteriormente conocidos y es entonces cuando se le reconoce (Guzmán 2009).

El continuo desarrollo de las Tecnologías de la Información y la Comunicación, la creciente evolución de las computadoras a nivel mundial y con ellas las ideas afines a la sustitución con estos dispositivos de algunas actividades humanas, dio paso al surgimiento del Reconocimiento de Patrones (RP).

El RP es una disciplina que tiene como bases fundamentales a la Matemática, la Computación y las Ingenierías; y se empieza a conformar como ciencia a partir de los estudios realizados a finales de los años 50 del siglo pasado por Frank Rosenblatt y con la publicación en 1965 del libro *Principios de la Neurodinámica* (Shulcloper 2009). En sus inicios el RP estuvo vinculado a la identificación, al reconocimiento y la clasificación de imágenes, por lo que mundialmente se identifica esta disciplina con todo lo relacionado exclusivamente al procesamiento y análisis de imágenes.

Las imágenes son, sin lugar a dudas, patrones y constituyen objetos de investigación del RP, pero son sólo uno de los patrones físicos que tienen esa misma virtud. Sin embargo, esta disciplina actualmente a nivel mundial ya abarca una cantidad mayor de problemas como los relacionados con las señales, los electrocardiogramas, los rasgos manuscritos e impresos, la descripción de un paciente en términos de sus síntomas y signos, una zona geológica en términos de sus magnitudes geofísicas y sus características geológicas y personas descritas por medio de sus peculiaridades socio-económicas (Shulcloper, Edurado et al. 1995).

Los problemas del RP son todos aquellos relacionados con la clasificación de objetos y fenómenos, y con la determinación de los factores que inciden en los mismos. Así, se consideran cuatro familias de problemas que se denominan respectivamente [4]:

- selección de variables y objetos;
- clasificación supervisada;
- clasificación no supervisada;
- clasificación parcialmente supervisada.

El problema de clasificación supervisada es el problema más conocido del Reconocimiento de Patrones y con el que más están relacionadas muchas de las actividades humanas, directa o indirectamente, con procesos de asociación por tipos y clases.

Los problemas que se abordan en el RP son tratados desde diferentes perspectivas. Dando origen a lo que autores denominan enfoques, entre los que se encuentran:

- Reconocimiento Estadístico.
- Reconocimiento Sintáctico Estructural.
- Reconocimiento Lógico Combinatorio de Patrones (RLCP).

Las ideas centrales del RLCP consisten en suponer que los objetos se describen por medio de una combinación de rasgos numéricos y no numéricos, y los distintos valores pueden ser procesados por funciones numéricas (González 2014). Se busca modelar los problemas prácticos de una manera más cercana al problema que se desea resolver. Este enfoque se caracteriza por lograr una mejor adaptación de sus modelos a las características de los problemas prácticos.

En el RLCP para solucionar los problemas de clasificación supervisada se utilizan modelos de algoritmos, agrupados según diferentes criterios como los ideales de las clases, los umbrales de exactitud, la tipicidad y el contraste y las precedencias parciales. Este último modelo de algoritmos es una descripción general de los algoritmos: ALVOT, CR y KORA.

En Cuba el RP tiene sus inicios a finales de los años setenta con dos vertientes fundamentales de trabajos (Shulcloper 2013):

1. el procesamiento digital de señales de audio, de voz principalmente, en la Universidad Central de las Villas, en el Ministerio del Interior y en el Instituto de Cibernética, Matemática y Física (ICIMAF)

2. las aplicaciones de herramientas matemáticas (Análisis Discriminante y Lógica Matemática, en particular, Teoría de Testores) a la solución de problemas en las Geociencias y las Ciencias Sociales en el Instituto de Geofísica y Astronomía, en la Universidad de la Habana y en el ICIMAF.

A partir de ese momento la disciplina ha tenido una evolución en el país destacándose el impacto alcanzado por la Asociación Cubana de Reconocimiento de Patrones y su congreso nacional, así como la creación y

participación en los Congresos Iberoamericanos de Reconocimiento de Patrones (CIARP). Ambos constituyen pilares importantes del desarrollo de esta ciencia en Cuba, tanto en los aspectos teóricos como en las aplicaciones de esta rama del conocimiento a los problemas prácticos propios del país.

Tributando a esta evolución se encuentran los estudios realizados en la Universidad de las Ciencias Informáticas (UCI). Esta institución cuenta con una la línea de investigación titulada *Inteligencia Artificial y Reconocimiento de Patrones*, en la cual se desarrollan modelos, herramientas y algoritmos para tratar los problemas concretos en aras de avanzar en la informatización de los procesos de la sociedad cubana.

El desarrollo y utilización de herramientas, framework[1] o librerías genéricas encaminadas a la solución de problemas, como los antes mencionados, específicamente los relacionados con el Reconocimiento Lógico Combinatorio de Patrones, se ha visto limitado por problemas metodológicos, que impiden su utilización en variados modelos de problemas de RLCP, sin que se afecte el modelado inicial con el que se trabaja (Oses, González et al. 2016).

En el año 2016 esta línea de investigación desarrolló la herramienta Entorno Cubano para el Reconocimiento Lógico Combinatorio de Patrones (CEPAR por sus siglas en inglés *Cuban Environment for Logical Combinatorial Pattern Recognition*). Es un Sistema Herramienta Universal (SHU) aún en desarrollo, que tiene como objetivo fundamental apoyar en las labores cotidianas de los investigadores, estudiantes y docentes que trabajan con el RLCP, además proveer de una herramienta que pueda ser embebida en desarrollos propios de una investigación (Oses, González et al. 2016).

La herramienta permite el trabajo con bases de datos; su edición, modelado y la selección de objetos y/o rasgos, aplicando la Teoría de Testores. Además, incluye algoritmos para la clasificación de los objetos de la base de datos, los cuales son:

- Componentes Conexas
- Componentes Compactas
- C-Means
- Holotipo
- KMSN

De ellos los cuatro primeros forman parte de los algoritmos de clasificación no supervisada y el último a los de clasificación supervisada.

Se considera que el hecho de que la herramienta CEPAR posea un solo algoritmo para la solución de problemas de clasificación supervisada restringe las formas y vías de modelar este tipo de problemas. Lo

[1] Marco de trabajo con artefactos o módulos concretos de *software.*

que limita la utilización y extensión de la herramienta en investigaciones y en la docencia. Además, reduce el espectro de formas y posibilidades de solucionar un problema específico.

La problemática antes descrita ha generado la necesidad de desarrollar una investigación que dé respuesta al siguiente **problema de la investigación:** ¿Cómo hacer más extensible la aplicación de la herramienta CEPAR en la solución de problemas de clasificación supervisada?

Teniendo en cuenta el problema antes descrito se define como **objeto de estudio:** Algoritmos de clasificación supervisada en el Reconocimiento Lógico Combinatorio de Patrones para hacer extensible la herramienta CEPAR, que contiene como **campo de acción:** el Algoritmo KORA

Se define como **objetivo general:** Implementar el algoritmo KORA para la solución de problemas de clasificación supervisada como extensión de la herramienta CEPAR.

Para dar cumplimiento al objetivo general se plantean como **objetivos específicos**:

1. Establecer el marco teórico-conceptual de los problemas y algoritmos de clasificación supervisada en el RLCP y del algoritmo KORA.
2. Analizar la herramienta CEPAR para su extensión con el algoritmo KORA.
3. Desarrollar el algoritmo KORA-3 y su extensión el KORA-Ω.
4. Validar mediante pruebas de software el efectivo funcionamiento del algoritmo KORA-3 y su extensión el KORA-Ω en la herramienta.

Durante el desarrollo de la investigación se emplearon los métodos de investigación siguientes:

Métodos Teóricos:

Permiten estudiar las características del objeto de investigación que no son observables directamente, facilitan la construcción de modelos e hipótesis de investigación y crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, contribuyendo al desarrollo de las teorías científicas (Coello and León 2002).

Analítico-Sintético: se utiliza en el análisis de documentos y la extracción de los elementos más importantes de los problemas y algoritmos de clasificación supervisada en el RLCP y del algoritmo KORA.

Histórico-Lógico: se maneja en el análisis de la trayectoria y condicionamiento a los diferentes periodos de la historia del algoritmo KORA, revelando las etapas principales de su desarrollo y las conexiones históricas fundamentales de su surgimiento.

Modelación: se emplea en el diseño de diagramas que permiten mostrar y comprender el algoritmo de clasificación supervisada KORA, así como su integración en la herramienta.

Sistémico: se utiliza en el estudio de la herramienta CEPAR mediante la determinación de sus componentes, así como la relación entre ellos. Esta relación determina por un lado la estructura y la jerarquía de cada componente en la herramienta y por otra parte su dinámica, siendo también la expresión del comportamiento de CEPAR como totalidad en el que un componente depende de otro u otros.

Métodos Empíricos:

Describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional (Coello and León 2002).

Entrevista: se desarrollaron entrevistas formales no estructuradas para obtener información rápida y precisa de los principales problemas, necesidades y posibles resultados esperados del algoritmo KORA.

En aras de estructurar el proceso de desarrollo del presente trabajo de una manera coherente y organizada, se dividió el mismo en 3 capítulos:

Capítulo 1. Fundamentos teóricos para la solución de problemas de clasificación supervisada en el Reconocimiento Lógico Combinatorio de Patrones: se abordan los elementos teóricos y conceptuales del Reconocimiento Lógico Combinatorio de Patrones, de los problemas de clasificación supervisada y de los modelos de algoritmos para resolver problemas de este tipo.

Capítulo 2. Descripción e implementación del algoritmo KORA: se describe el algoritmo a implementar KORA, además se modela la propuesta de la extensión de este algoritmo para el trabajo con problemas de clasificación supervisada desde el punto de vista teórico. También se muestra el proceso de integración de estos algoritmos en la herramienta CEPAR.

Capítulo 3. Validación del algoritmo KORA en la herramienta CEPAR: se realizan pruebas de software, específicamente de unidad al código de programación realizado y de integración para comprobar la correcta relación entre las clases del algoritmo implementado con las ya existentes en la herramienta.

CAPÍTULO 1 FUNDAMENTOS TEÓRICOS PARA LA SOLUCIÓN DE PROBLEMAS DE CLASIFICACIÓN SUPERVISADA EN EL RECONOCIMIENTO LÓGICO COMBINATORIO DE PATRONES

Introducción

En el presente capítulo se abordan los fundamentos teóricos para la solución de problemas de clasificación supervisada, lo que conlleva a enunciar conceptos importantes de la disciplina RP, de sus enfoques, del problema de clasificación supervisada y de los algoritmos para su solución. Además, se realizan valoraciones sobre los conocimientos esenciales de CEPAR como plataforma para el desarrollo de la solución, así como de las tecnologías y herramientas de desarrollo utilizadas en la implementación.

1.1 Reconocimiento de Patrones

El RP es una disciplina en el área de la ciencia que pretende reemplazar algunas de las actividades humanas con dispositivos electrónicos. Ha sido definida por varios autores como son:

Según José Ruíz Shulcloper: *es una ciencia con un fuerte carácter aplicado e interdisciplinario. Está relacionado con procesos (ingenieriles, físicos, matemáticos y computacionales) de datos que provienen de descripciones de objetos (fotos, hologramas, escrituras, jeroglíficos, símbolos, señales bioeléctricas, acústicas, pacientes, zonas geológicas, etc.) con el propósito de obtener (por medio de dispositivos computacionales y/o seres humanos) información que permita establecer las propiedades de ciertos subconjuntos de objetos y/o las relaciones entre ellos. También conocer de las propiedades que poseen las variables en términos de las cuales estos objetos son representados. Estas propiedades constituyen el soporte del posible conocimiento que de estos datos podemos extraer* (Shulcloper, Ochoa et al. 2013).

Anil K. Jain: *es el estudio de cómo las máquinas pueden, observando el ambiente, aprender a distinguir patrones de interés de un fondo y realizar decisiones razonables sobre las categorías de los mismos* (Jain, Duin et al. 2000).

Fukunaga: *el Reconocimiento de Patrones está relacionado con el proceso de toma de decisiones de un ser humano. El objetivo de este es aclarar los mecanismos complicados de los procesos de toma de decisiones y automatizar las funciones usando computadoras. Sin embargo, debido a la compleja naturaleza del problema, la mayoría de las investigaciones de Reconocimiento de Patrones se han concentrado en los problemas más realistas, como el reconocimiento de caracteres latinos y la clasificación de formas de onda* (Fukunaga 1990).

En (Razo Gil 2013), (Borroto, Villuendas et al. 2013) y demás autores coinciden con la primera de las definiciones abordadas, por lo que en la presente investigación se adopta el concepto de José Ruíz Shulcloper, porque es el autor que muestra una definición más cercana al enfoque centro de atención de la investigación.

En RP, en general, son básicos los conceptos de objeto, patrón, universo de objetos, rasgos, representación o descripción de objetos, espacio de representaciones, clases y reconocimiento. A continuación se abordan algunos de ellos tomados de (Shulcloper 2009).

Desde un punto de vista intuitivo, **objeto** ha sido un término empleado en diferentes disciplinas para referirse a los entes sujetos a estudio. Ejemplos de objetos pueden ser considerados un paciente de una enfermedad, una zona geográfica, un dispositivo eléctrico, un conjunto de personas, una bioseñal, una fotografía, y muchos otros.

Patrón es sinónimo de objeto. En algunas ocasiones es conveniente establecer diferencias entre los objetos acerca de los cuales se conocen ciertas propiedades (por ejemplo, la pertenencia a una clase, tipo, etc.) de otros, de los cuales no se sabe nada acerca de dichas propiedades. En esos casos suele usarle el término objeto para aquellos de los que se desconocen estas propiedades y a los otros se les llama patrones.

Por **universo de objetos** se entiende el conjunto de todos los posibles objetos admisibles para los propósitos del estudio en cada caso particular. Es por ello un concepto relativo a la investigación que se esté realizando. En general, en este trabajo este concepto se manejará de manera análoga a como se hace en la Teoría de Conjuntos.

Clase es un conjunto de objetos que constituyen un tipo, una categoría, en el contexto de los objetos del universo. Siempre será un subconjunto propio del universo de objetos. El término “conjunto” asume implícitamente el que todos los objetos de la clase satisfacen una cierta propiedad (lo que en Teoría de Conjuntos se denomina, la determinación intencional del conjunto).

Rasgo (atributo, variable, característica, propiedad, primitivas de un alfabeto) es un factor a tener en cuenta en el estudio de los objetos dados. De hecho, constituyen la vía real para poder estudiar, procesar, analizar a los objetos. Es decir, se tienen que estudiar descripciones de los objetos en términos de un conjunto de rasgos. En ciertos problemas de la realidad estos rasgos constituyen el objetivo central del estudio: su relevancia informacional, su influencia en un fenómeno dado, entre otros, son cuestiones que se requieren conocer por sí mismas y en ocasiones son necesarias para ser empleadas en el proceso de reconocimiento. Al conjunto de valores admisibles que tiene asociado un rasgo se le denomina **dominio**.

Reconocimiento es un proceso para captar, interpretar, clasificar a los objetos de un universo dado a partir de sus descripciones y además encontrar propiedades, regularidades entre dichas descripciones.

Es el procedimiento por el cual se determina la pertenencia de un objeto a un cierto conjunto de objetos (clase). En ocasiones estas clases son conocidas, se tienen muestras de objetos que pertenecen respectivamente a cada una de ellas. En ocasiones no se tienen muestras de todas estas clases, aunque se sabe que existen y en otras ocasiones no se tienen ejemplos de objetos que estén en clase alguna y de hecho en ciertos problemas, muy frecuentes en la práctica real, no se sabe ni cuántas clases pueden existir en el universo de objetos en cuestión. Es también el proceso para determinar la importancia informacional de un rasgo o un conjunto de rasgos, o las propiedades que en ellos pudieran estar implícitas.

1.2 Enfoques del Reconocimiento de Patrones

Esta variedad de criterios entorno al Reconocimiento de Patrones ha traído aparejado diversos enfoques para resolver sus distintos problemas. Atendiendo a los mecanismos utilizados para representar o modelar estos problemas se describen cuatro enfoques como se muestra en la Figura 1:

Figura 1 Enfoques del Reconocimiento de Patrones (Oses, González et al. 2016)

El *enfoque Reconocimiento Estadístico de Patrones* como se muestra en la imagen anterior se representa mediante descripciones de objetos en términos de mediciones, es decir, variables numéricas. A dichas variables se le presuponen propiedades tales como las de estar definidas sobre un espacio métrico o normado, e incluso en ocasiones se asume un tipo particular de métrica, por lo general de las muy conocidas y para las que se cuentan con una gran cantidad de herramientas matemáticas (Shulcloper, Arenas et al. 1999).

Este enfoque se basa en la teoría de probabilidad y estadística y se supone que tiene un conjunto de medidas numéricas con distribución de probabilidad conocida o estimable, y a partir de ella se hace el reconocimiento (Eduardo 2015).

En este enfoque es muy frecuente el uso de probabilidades, en particular cuando se considera la presencia de elementos de incertidumbre o subjetividad. Pero también en estos casos es frecuente el asumir un determinado comportamiento de dichas probabilidades y con ello aparece la suposición de ajustarse a distribuciones normales, por lo general asumiendo valores medios y desviaciones determinadas de los mismos (Shulcloper, Arenas et al. 1999).

El *Reconocimiento Sintáctico Estructural* se basa en las descripciones de los objetos en términos de sus partes constitutivas. Este enfoque se basa en los objetos que no están descritos como vectores de atributos, por ejemplo, objetos descritos como: grafos, cadenas de símbolos, etc. Normalmente el objetivo es encontrar las relaciones estructurales que guardan los objetos de estudios (Eduardo 2015).

Este enfoque se apoya en la Teoría de los Lenguajes Formales, la Teoría de Autómatas, las Funciones Recursivas; Teoría de Grafos, en particular el estudio relacional entre partes constitutivas de los objetos, con frecuencia emplea medidas de similaridad estructural. Se asume que la estructura de los objetos a ser reconocidos es cuantificable. De forma muy general se puede afirmar que este enfoque asocia a cada conjunto de objetos una gramática que genera sólo elementos de dicho conjunto y el problema consiste en averiguar, cuál de las gramáticas genera como palabra, la correspondiente al objeto que se desea clasificar; o también que a cada conjunto de objetos se le asocia un grafo que describe las relaciones entre las propiedades estructurales de un objeto representante del conjunto de objetos, aquí se compararían los grafos asociados a cada representante de las clases con el del objeto que se quiere clasificar (Shulcloper, Arenas et al. 1999).

En los casos en que las descripciones de los objetos (zonas geológicas, pacientes o enfermedades) son sólo elementos de un producto cartesiano sin ninguna suposición de tipo algebraico, lógico o topológico sobre el mismo, ¿cómo se seleccionan los rasgos más informativos y se clasifica un nuevo objeto dada una muestra de entrenamiento? Este es el problema central en el que se enfoca el Reconocimiento Lógico Combinatorio de Patrones y constituye una necesaria respuesta metodológica al problema de procesar los datos mezclados e incompletos (DMI) en problemas de RP. Bajo estas condiciones el enfoque del RLCP es más adecuado para trabajar que los enfoques tradicionales, dado que permite captar mejor la realidad del problema y al mismo tiempo, procesar los datos mediante una modelación matemática en un contexto metodológico en el que la realidad no es desvirtuada. Ello no implica la imposibilidad de ser empleado en casos en que las descripciones de los objetos sean exclusivamente numéricas o no numéricas (Shulcloper

2007). Por tales motivos es el eje central de la presente investigación y será abordado a fondo en el subepígrafe siguiente.

1.2.1 Reconocimiento Lógico Combinatorio de Patrones

Este enfoque se basa en la idea de que el modelado del problema debe ser lo más cercano posible a la realidad del mismo, sin hacer suposiciones que no estén fundamentadas. Uno de los aspectos esenciales del enfoque es que los atributos utilizados para describir a los objetos de estudio deben ser tratados cuidadosamente para no realizar operaciones que resulten antinaturales respecto al problema que están representando. Este tratamiento cuidadoso permite trabajar con atributos cualitativos y cuantitativos e incluso con ausencia de información (Ochoa and Trinidad 2011) (Eduardo 2015).

El enfoque lógico combinatorio facilita analizar las variables de manera diferenciada, lo que permite que las mismas puedan ser simultáneamente de naturalezas diferentes (cuantitativas, cualitativas) e incluso permite trabajar con descripciones incompletas de objetos, como es frecuente que ocurra en muchas ramas de la Ciencia, en particular en las zonas poco formalizadas del conocimiento (medicina, geología, sociología) y como aparece en muchos problemas prácticos.

El Reconocimiento Lógico Combinatorio de Patrones encuentra su basamento teórico matemático en la Lógica Matemática, la Teoría de Testores, la Teoría Clásica de Conjuntos, la Teoría de los Subconjuntos Difusos, la Teoría Combinatoria y la Matemática Discreta en general (Shulcloper, Arenas et al. 1999).

Es más adecuado para trabajar bajo las condiciones de descripciones mezcladas e incompletas, dado que permite captar mejor la realidad del problema y al mismo tiempo, procesar los datos mediante una modelación matemática en un marco metodológico, en el que la realidad no es desvirtuada. Esto no implica la imposibilidad de ser empleado, en casos en que las descripciones de los objetos sean exclusivamente numéricas o no numéricas, a diferencia de otros enfoques (Shulcloper 2009).

En este enfoque la clasificación se realiza formando agrupaciones de los elementos de la muestra, de acuerdo a la semejanza que existe entre ellos, dando un valor específico a cada uno de los rasgos que distingan a los elementos y de acuerdo con esos valores se estudia la semejanza entre ellos (Vázquez 2009).

Sin embargo, contrastando con su amplio espectro de investigación y enorme potencial de aplicación, este enfoque presenta también la característica de que los algoritmos tienen, por lo general complejidad alta y baja eficiencia computacional. Los resultados de estos algoritmos y sus interpretaciones son altamente sensibles a los detalles del modelado matemático, en particular, a las características del conjunto de

atributos que describen los patrones, así como a las propiedades de las funciones planteadas de semejanza entre patrones (Hernández 2009).

1.3 Problemas que resuelve el Reconocimiento de Patrones

Por problemas de reconocimiento de patrones se entienden todos aquellos relacionados con la clasificación de objetos y fenómenos y con la determinación de los factores que inciden en los mismos (Shulcloper 2009) (Peña 1995). Así, se consideran cuatro familias de problemas que se abordan a continuación:

El *problema de selección de rasgos y/u objetos* es uno de los pasos fundamentales en cualquier problema de clasificación debido a que la mayoría de los problemas de RP están basados en la descripción de los objetos en términos de un conjunto de rasgos (Shulcloper 2009). De forma general consiste en encontrar el subconjunto de rasgos que mejor describe los objetos del dominio; usualmente este subconjunto se encuentra maximizando o minimizando una función objetivo.

Existen diferentes enfoques y técnicas para seleccionar rasgos relevantes, tales como: técnicas de optimización de colonias de hormigas, las de computación evolutiva, las basadas en la teoría de los conjuntos aproximados y el enfoque Lógico Combinatorio, problemas como la determinación de síndromes, factores de riesgos, perfiles de usuarios, características discriminantes de conjuntos de objetos o fenómenos. Todos son instancias de este tipo de problemas en el mundo real (González 2014).

Los *problemas de la clasificación no supervisada* (problemas de agrupamiento – clustering, structuralization, en inglés) constituyen una de las etapas más importantes en la mayoría de las ciencias, en particular, las naturales y sociales. La Taxonomía es el primer período de trabajo científico en muchas de las áreas del conocimiento y es una de las herramientas necesarias en muchas de las investigaciones en la actualidad (Shulcloper 2009).

El objetivo principal a solucionar en estos problemas es encontrar las relaciones entre los objetos de un universo en términos de sus características (rasgos). Estas relaciones se establecen sobre la base del concepto de analogía (similaridad); partiendo de que no se definen con anterioridad las clases o grupos a los cuales los objetos a relacionar van a pertenecer, sino que estas clases van a ser determinadas a partir de las agrupaciones realizadas. El propósito es juntar (agrupar) los objetos según su analogía (parecido, semejanza, cercanía si se está hablando de un espacio de representación con distancia definida). En este sentido se pueden encontrar dos situaciones diferentes, a saber, el número de grupos (clúster) es conocido previamente o no (Shulcloper 2009).

Cuando en el universo de objetos dado, se conoce la existencia de ciertas clases e incluso se tienen muestras de algunas de ellas, pero no de todas; lo que se evidencia es el *problema de clasificación*

parcialmente supervisada. Esta es quizás la familia de problemas de RP en la que menos estudios se han realizado. A pesar de la existencia de muchos problemas reales donde aparecen situaciones de este tipo, la clasificación parcialmente supervisada no ha recibido igual atención que las restantes familias de problemas en RP. El problema central es clasificar nuevos objetos en circunstancias, en las que no se tienen muestras de todas las clases y en las que incluso pudieran existir clases que se desconocen (Shulcloper 2009).

El problema consiste en una combinación de los problemas de clasificación anteriormente descritos. En el universo de objetos dado, se conoce la existencia de ciertas clases e incluso se tienen muestras de algunas de ellas, pero no de todas. Por ejemplo, se sabe que en una habitación hay un grupo de personas conversando. No se conocen las voces de todas las personas, que se encuentran en la habitación. Se puede incluso no conocer cuántas personas se encuentran en la habitación. El problema pudiera ser determinar qué dijo cada persona durante la conversación (Shulcloper 2009).

En el caso de que previamente a la clasificación se definan las clases en las cuales los objetos serán agrupados se estará en presencia de un problema de clasificación supervisada que por ser los problemas base de esta investigación se explica a profundidad en la subepígrafe siguiente.

1.3.1 Problema de Clasificación Supervisada

Dado un universo de objetos y el conocimiento acerca de la existencia de ciertas clases con características (propiedades) de especial interés y una muestra de objetos que pertenecen a cada una de ellas, el problema consiste en determinar para cada uno de los objetos no clasificados las relaciones de pertenencia de los mismos con cada una de las clases (Shulcloper 2009) (García-Borroto, Martínez-Trinidad et al. 2014).

Es natural considerar el problema de la clasificación supervisada (Figura 2) desde el punto de vista de la Teoría de Conjuntos, ya que la solución de un problema cualquiera de clasificación es, en esencia, estudiar las relaciones de pertenencia de ciertos elementos respecto a ciertos conjuntos (Shulcloper, Arenas et al. 1999) (Borroto 2010).

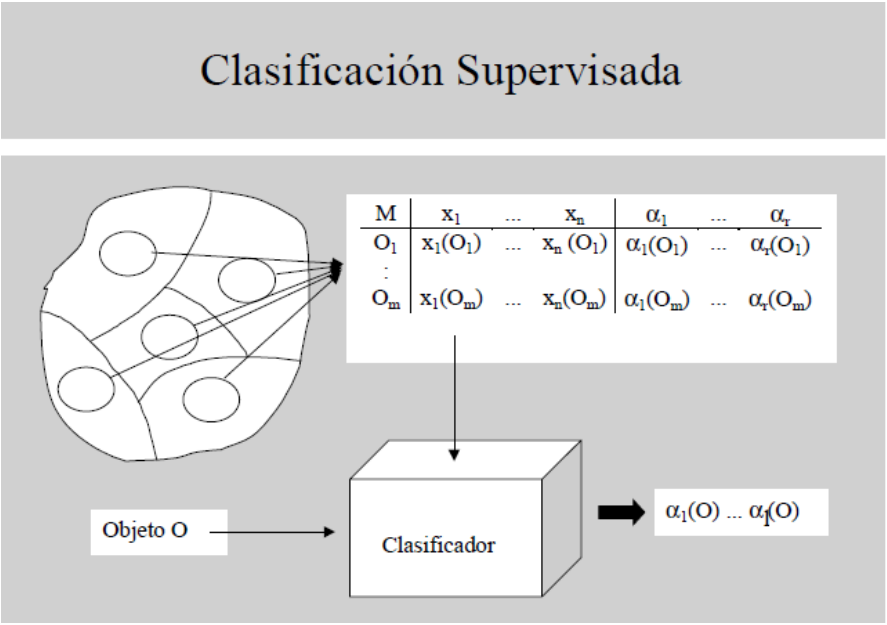

Figura 2 Proceso de Clasificación Supervisada (Carrasco-Ochoa, 2011 #25)

Los conjuntos no se definen, no existe un concepto en término del cual se pueda dar la definición de conjunto. Colección, agrupamiento, grupo, y otros, no son más que sinónimos de conjunto y ninguno de ellos es conceptualmente, semánticamente anterior al de conjunto. Así pues, no los podemos definir, pero sí se pueden determinar (Shulcloper, Arenas et al. 1999).

Existen dos formas de saber si un elemento pertenece o no a un conjunto:

1. si se tiene el listado de todos sus elementos componentes, se comprueba si el elemento aparece en el mismo,
2. si se conoce la propiedad que lo caracteriza, que lo determina, se verifica si la cumple o no.

Sin embargo, en muchas ocasiones no se tiene el listado completo de los elementos de un conjunto ni se conoce la propiedad que caracteriza al conjunto de manera unívoca o se conoce la propiedad de una manera poco precisa. Puede ocurrir que se dispone de una formulación de dicha propiedad que presupone determinados conocimientos previos de la persona que tiene que decidir en cuanto a la pertenencia de un elemento a un conjunto dado (Shulcloper, Arenas et al. 1999).

Un problema de clasificación supervisada consiste en que se asume que el universo U está estructurado en un número finito r de subconjuntos propios de cada uno de los cuales se tiene una muestra de descripciones de sus objetos. El problema consiste en encontrar las relaciones de pertenencia de un nuevo objeto de U (fuera de la muestra dada) con las r clases. Esta relación, como ya se ha mencionado, no tiene que ser de: todo o nada, ni única, es decir, se puede admitir multiclasificación (clases solapadas) e incluso pertenencia

gradual a las clases (modelación difusa) (Shulcloper 2009) (Arenas and Trinidad 2001) (Shulcloper, Ochoa et al. 2015).

Los problemas de clasificación supervisada dentro del Reconocimiento Lógico Combinatorio de Patrones se presentarán formalmente de la manera siguiente: Sea M un conjunto de objetos. Para cada objeto $O \in M$ se tiene una descripción $I(O)$ valores para los n rasgos $\mathrm{x}_1(\mathrm{O}),\ldots,\mathrm{x}\,(\mathrm{O}\,)_n$ en términos de los rasgos $\mathrm{R} = \{\,x_1,\ldots,x_n\,\}$, la cual es representada por una sucesión de valores para los n rasgos $\mathrm{x}_1(\mathrm{O}),\ldots,\mathrm{x}_n\,(\mathrm{O})$, con $x_i(\mathrm{O}) \in \mathrm{M}_i$, el conjunto de valores admisibles n de la variable $x_i, \mathrm{i} = 1,\ldots,\mathrm{n}$, al cual se le añade un símbolo $(*)$ que denotará la ausencia de información. En otras palabras, se admiten descripciones incompletas de los objetos, i.e., al menos uno de sus valores es desconocido (Ochoa 2015) (Shulcloper and Cortés 1995).

Así, se considera que $\mathrm{I(O)} \in \mathrm{M}_1\mathrm{X}\ldots\mathrm{XM}_n = \mathrm{ERI}$, el producto cartesiano de los conjuntos de valores admisibles de las variables, al que denominaremos espacio de representación inicial (ERI). La naturaleza de las variables puede ser, simultáneamente, cualquiera (cualitativa: Booleana, multivaluada, difusa, lingüística y otras - como la descripción del dolor, la valoración de la palidez, la evaluación cualitativa de la presión arterial, etc.; o cuantitativas: enteras, reales y otras). Esto es, sobre M_i no supondremos definida a priori operación algebraica o lógica alguna, ni una métrica. Dichas operaciones y medidas pueden estar presentes, pero no son necesarias (Ochoa 2015).

Cuando de ERI se conoce que sus elementos están estructurados en clases $\mathrm{K}_1,\ldots,\mathrm{K}_r$ y que una matriz de aprendizaje (MA) está formado por una muestra de cada una de ellas, esto es, MA es la unión de $\mathrm{K}'_1,\ldots,\mathrm{K}'_r$ con $\mathrm{K}'_i \subseteq \mathrm{K}_i\,, \mathrm{i} = 1,\ldots,\mathrm{r}, \mathrm{r} \geq 2$, un problema de Clasificación Supervisada consiste en hallar un algoritmo tal que permita determinar las relaciones de pertenencia entre cualquier objeto del ERI y las diferentes clases K_i, las cuales pudieran ser conjuntos duros o difusos, disjuntos o solapados. Ejemplos de este problema lo son, el pronóstico de complicaciones postoperatorias, la lectura diagnóstica de bioseñales, el diagnóstico precoz de enfermedades, el pronóstico de esperanza de vida, la lectura diagnóstica de imágenes médicas, y muchas otras (Ochoa 2015).

Se puede suponer, en general, que las clases $\mathrm{K}'_1,\ldots,\mathrm{K}'_r$ son subconjuntos difusos de ERI, asumiendo que los objetos del espacio considerado, pertenecen a cada uno de esos conjuntos en un cierto grado. A cada objeto $\mathrm{O} \in \mathrm{ERI}$ puede asociársele un r-uplo de pertenencia $\alpha(\mathrm{O}) = (\alpha_1(\mathrm{O}),\ldots,\alpha_r(\mathrm{O}))$ de modo que $\alpha_i(\mathrm{O}) = \mu_{K_i}(\mathrm{O})$ denota el grado de pertenencia del objeto O a la clase K_i, i=1,...,r (Ochoa 2015).

1.4 Algoritmos de Clasificación Supervisada

El problema de la clasificación supervisada ha sido abordado desde ópticas diferentes, a partir de enfoques distintos (Figura 3). Así, suponiendo que los objetos están descritos en forma de n-uplos de un cierto

espacio, se han desarrollado modelos basados en superficies de separación, cuya idea esencial consiste en que un conjunto de objetos representados en un espacio vectorial, digamos $\Re^n$, (denotando por $\Re$ al conjunto de los números reales), puede ser separado por una cierta hipersuperficie si estos pertenecen a clases diferentes. Presuponiendo además que los objetos, al agruparse en clases diferentes, se deben ubicar en dicho espacio de modo tal que los que pertenecen a la misma clase están más cercanos entre sí que con respecto a clases diferentes. Esto en la práctica no siempre es así (Shulcloper 2009).

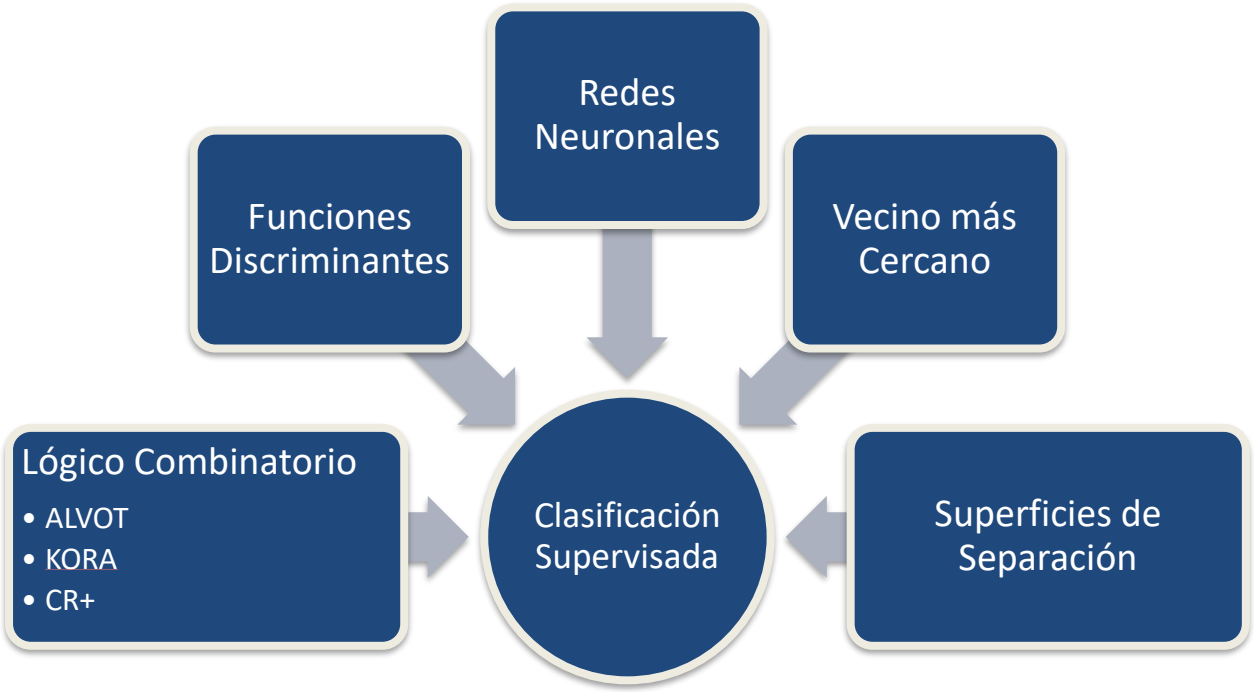

Figura 3 Modelos de algoritmos de clasificación supervisada (elaboración propia)

Teniendo esta misma idea como base metodológica, se ha desarrollado otro modelo denominado vecinos más cercanos cuya idea esencial es la distancia entre los objetos. Al ubicar los objetos en el espacio, que se supondrá métrico, decidiremos por una u otra clase en dependencia de cuán cercanos estén éstos respecto a ciertos objetos de los ya ubicados en las diferentes clases definidas en el problema. La selección de esos objetos distinguidos entre los ya clasificados se realiza de varias formas, todas basadas en la idea fundamental de este modelo: la cercanía entre las representaciones de objetos en el espacio seleccionado con las características antes mencionadas (Rey, Pérez et al. 2008).

En los modelos mencionados anteriormente, las ideas esenciales de los mismos han estado relacionadas con ciertos aspectos geométricos, en cierto sentido topológico, que pueden aparecer en el espacio de representación de los objetos admisibles. Bien la superficie de separación, bien la menor distancia a un n-uplo, o a un conjunto de ellos, requieren del agrupamiento de los n-uplos en el espacio, requieren de una cierta disposición espacial para que los modelos se aproximen con una aceptable eficiencia a la realidad.

Más aún, requieren de determinadas características de ese espacio de representación inicial, como por ejemplo la de permitir definir una distancia al menos.

Es oportuno precisar tres términos que no se usan en este documento como sinónimos a pesar que, en la literatura, en ocasiones, sí se hace, se trata de los conceptos de modelo de algoritmos, algoritmo y clasificador.

De manera intuitiva se puede decir que un modelo o familia de algoritmos es una descripción general de un conjunto de algoritmos, quienes a su vez son descripciones generales de un conjunto de clasificadores. Esto significa que un clasificador es una instancia de un algoritmo quien a su vez es una instancia de un modelo de algoritmos (Shulcloper 2009).

En particular un algoritmo de clasificación supervisada es un procedimiento efectivo que cumple las condiciones siguientes (Shulcloper 2009):

a) Requiere de una información estándar $I(K_1, \dots, K_r)$ correcta para las clases $K'_1, \dots, K'_r$ de un universo U dado.
b) Requiere de un criterio de comparación entre descripciones de los objetos de U.
c) Admite como entrada, la descripción de un objeto admisible de U en términos de los mismos rasgos que la información estándar.
d) Devuelve como salida un r-dupla de pertenencia.

En problemas de clasificación es deseable que los objetos en cada clase sean representativos de la misma y a su vez se diferencien suficientemente de los de otras clases, lo que claramente puede ayudar a la eficacia de los clasificadores que se empleen. El peso informacional de un objeto en una clase crecerá en la medida que lo sea su semejanza por rasgo con los objetos de esa clase y la importancia informacional de dichos rasgos. Mientras que el peso diferenciante de un objeto será mayor en la medida en que ese objeto se parezca menos a los objetos de las otras clases en los rasgos de mayor importancia. Con estos objetivos presentes, se han introducido una serie de conceptos que tienden a dar una respuesta en el sentido de la importancia informacional de los objetos, dentro de ellos la definición de los modelos de algoritmos de clasificación basados en el peso de los patrones dentro de los que se destacan (Shulcloper 2009):

- Algoritmos basados en los ideales de las clases.
- Algoritmos basados en umbrales de exactitud.
- Algoritmos basados en la tipicidad y el contraste.
- Algoritmos basados en las precedencias parciales.

La presente investigación centra su estudio en el último modelo de algoritmos mencionado. Este modelo descansa sobre la idea de las precedencias parciales, que permite la solución de problemas complejos que con frecuencia se encuentran en ciencias tales como las biomédicas y las Geociencias. Dentro de este modelo se encuentran algoritmos tales como:

Algoritmos de Votación (ALVOT): tuvo su origen en los años 1965 aproximadamente y sus desarrollos se deben al especialista ruso Yu. I. Zhuravliov y a su grupo. La idea esencial del modelo, como se ha mencionado, es la de las precedencias parciales (Ochoa and Trinidad 2002) (Ochoa and Trinidad 2003). Se trata de que un objeto puede parecerse a otro, pero no en su totalidad y las partes en que sí .se parecen pueden dar información acerca de posibles regularidades. Por supuesto, no todas en la misma magnitud (Shulcloper 2009) (Pérez and Shulcloper 2012) (Franco and Díaz 2009).

Algoritmos de clasificación usando conjuntos de representantes (CR+): es una generalización del modelo introducido por Baskakova y Zhuravliov que introduce la idea de valorar información a favor y en contra de la pertenencia de los objetos a las clases, así como la de considerar que los parámetros utilizados para la clasificación deben estar asociados a cada clase (Ochoa 2015) (Ochoa, Doria et al. 1998). Consta de 3 etapas: Estimación de parámetros, Cálculo de Representantes y Clasificación (Ochoa and Martínez 2013).

Algoritmos tipo KORA: fue desarrollado para resolver problemas de clasificación en Geofísica y Geología. Este modelo introduce la idea de emplear diferentes niveles de representación. La idea central del modelo es encontrar características complejas que sirvan para identificar objetos de una clase (Ochoa 2015).

Los algoritmos tipo KORA ocupan un lugar especial en la historia del desarrollo de los métodos de Reconocimiento de Patrones en la Geología. En primer lugar, éste fue el primer programa de clasificación supervisada que se utilizó en la solución de problemas geológicos. En segundo lugar, es el programa de más larga vida, activo aún después de más de 50 años (Shulcloper, Ochoa et al. 2015).

Por las razones antes expuesta esta última familia de algoritmos es la que se implementará como parte de la herramienta CEPAR en la presente investigación. Por lo que será más abordado en el Capítulo 2.

1.5 Herramienta CEPAR

La herramienta Entorno Cubano para el Reconocimiento Lógico Combinatorio de Patrones (por sus siglas en ingles CEPAR) en su versión 1.0.0 (Figura 4), es un Sistema Herramienta Universal para el RLCP.

Por sistema herramienta entenderemos un complejo de programas (de computadora) orientados a la solución de una familia de problemas de una o más áreas específicas del conocimiento, tal que el usuario sólo requiere del conocimiento del área específica y además el complejo de programas no resulta para él

una caja negra. Si el sistema es aplicable a cualquier área del conocimiento lo denominaremos Universal (Shulcloper and Pico-Peña 1992).

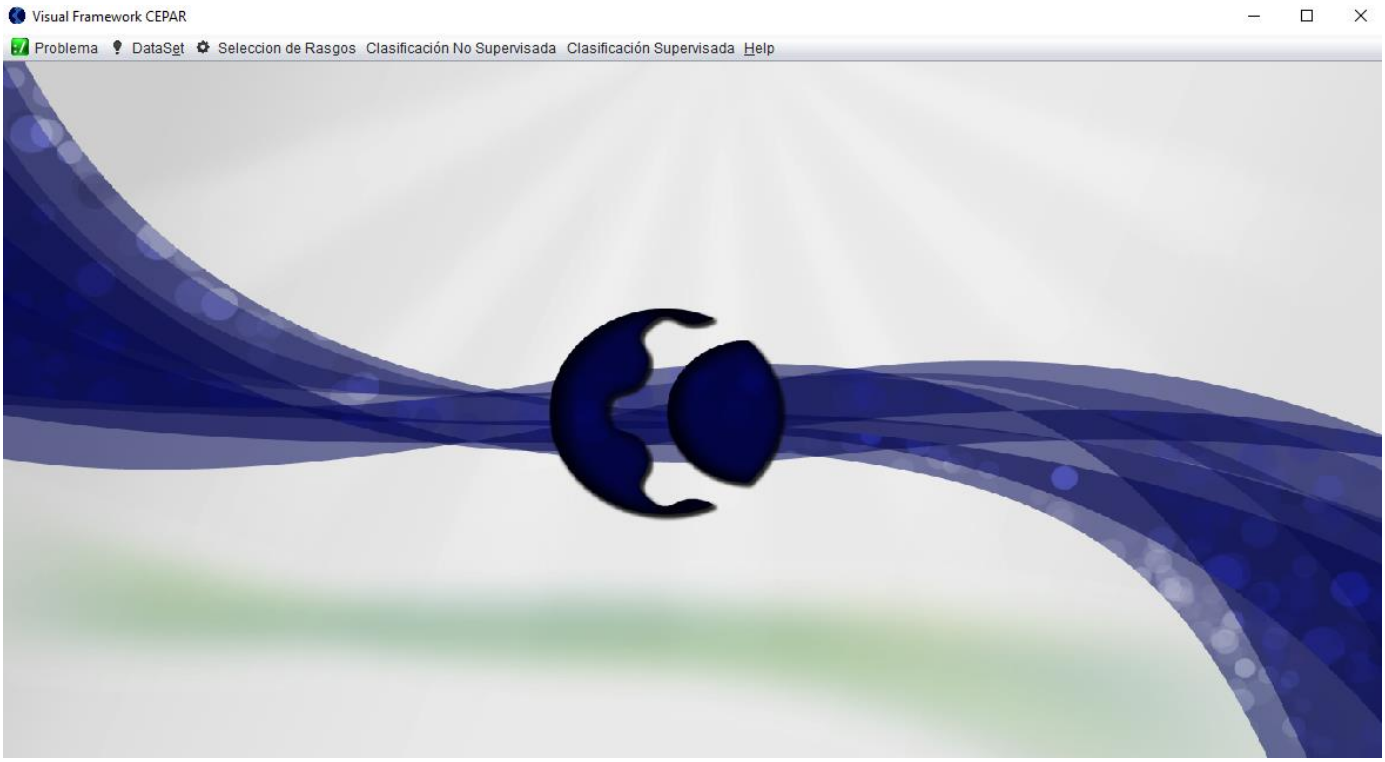

Figura 4 Herramienta CEPAR. Ventana de Inicio

CEPAR posee como objetivo fundamental apoyar en las labores cotidianas de los investigadores, docentes y estudiantes que trabajan o estudian los problemas del RLCP, además proveer de una herramienta que pueda ser embebida en desarrollos propios de una investigación (Oses, González et al. 2016).

Desarrollado empleando JAVA como lenguaje de programación, dado sus comodidades como lenguaje multiplataforma, CEPAR sólo requiere de la Máquina Virtual de Java (JVM) para poder ser utilizado.

CEPAR es una herramienta que presenta una arquitectura por capas (Figura 5) pues se definen capas diferentes; cada una ejecuta operaciones que se aproximan progresivamente al conjunto de instrucciones de máquina. En la capa externa, los componentes atienden las operaciones de la interfaz de usuario. En la interna, los componentes realizan la interfaz con el sistema operativo. Las capas intermedias proveen servicios de utilerías y funciones de software de aplicación.

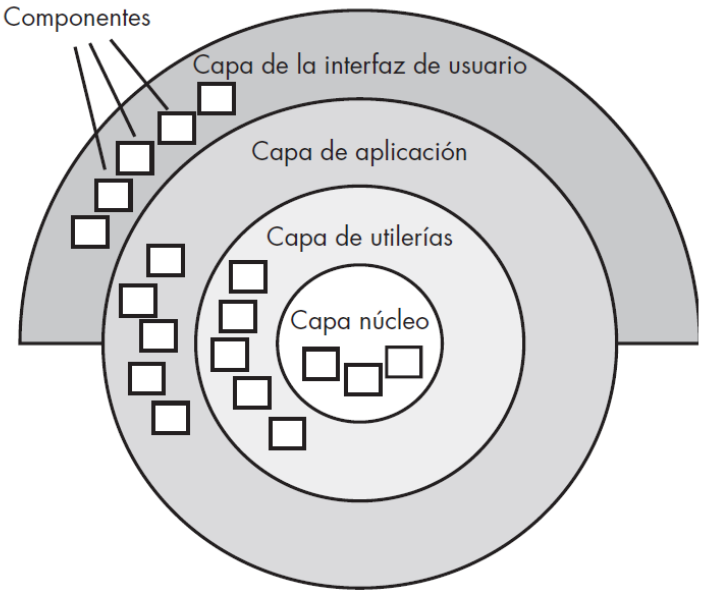

Figura 5 Arquitectura por capas presente en la herramienta CEPAR (Pressman 2010)

Plantea un diseño modular (Figura 6), de manera que permite la incorporación de nuevos rasgos, funciones de semejanza, o algoritmos. Cada módulo es independiente y se relaciona con los demás a través de las clases e interfaces definidas en su núcleo (*cepar.core*), que contiene las características más generales para modelar un problema de RLCP. Cada uno de los módulos, se encuentra a su vez fraccionado en distintos paquetes, atendiendo a las clasificaciones internas de cada uno de los algoritmos (Oses, González et al. 2016) (Hernández and Figarola 2016) (Ung 2016).

El diseño modular planteado permite al quinto módulo Interfaz Visual, cargar cualquier nuevo algoritmo, rasgo, criterio de comparación, función de semejanza u otras características, incluso una vez compilada la herramienta. Siendo esta característica una de las principales ventajas que presenta CEPAR la posibilidad de extenderse con nuevas funciones, no limitándose a los valores iniciales con que fue creada.

De forma nativa CEPAR maneja los tipos de rasgos:

- BooleanFeature: valores booleanos.
- DateFeature: valores de tipo fecha (día-mes-año).
- DoubleFeature: valores reales.
- FloatFeature: valores de punto flotante.
- IntegerFeature: valores enteros.
- StringFeature: valores alfanuméricos.

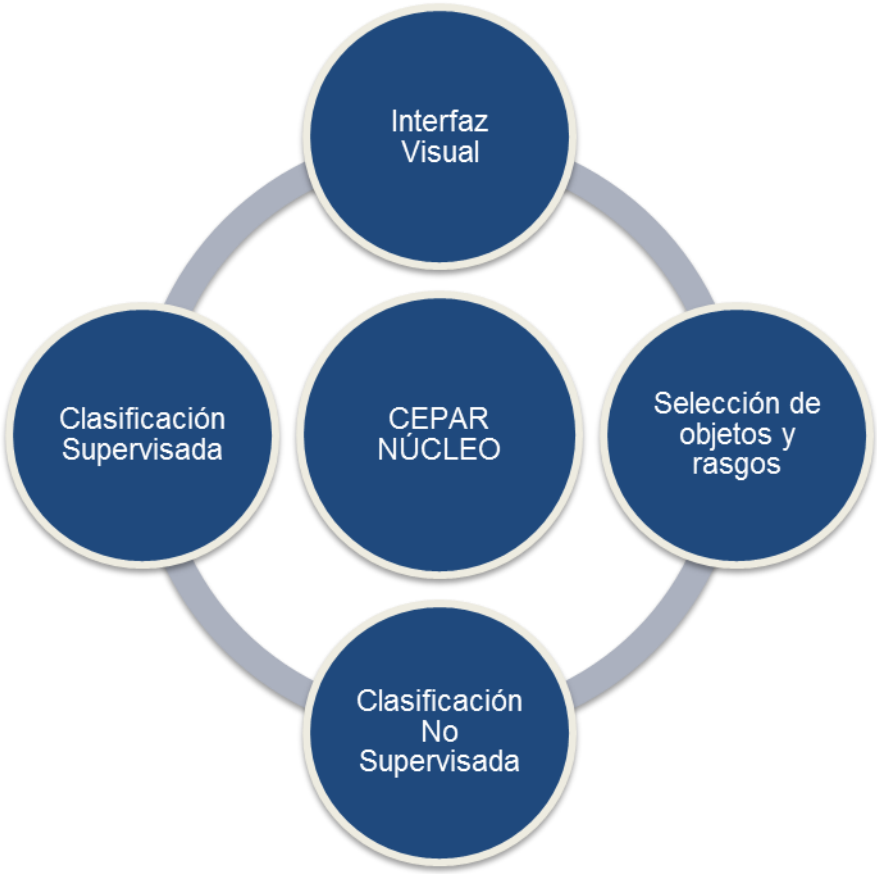

Figura 6 Diseño Modular de CEPAR (elaboración propia)

En la práctica esta cantidad de rasgos resulta minúscula, en comparación con los que pueden aparecer en problemas reales del RLCP, por lo que resalta en CEPAR el permitir la extensión según las interfaces definidas de nuevos rasgos por los usuarios, no limitando la herramienta solo al trabajo con los seis rasgos nativos

Ventajas

- Para todos los tipos de rasgos soporta la ausencia de información o el valor “?”.
- La herramienta permite el trabajo con variables cuantitativas y cualitativas (simultáneamente).
- Permite definir criterios de comparación y funciones de analogía entre patrones, de manera que puede extenderse con nuevos algoritmos para el procesamiento de la información desarrollados por los usuarios.
- Puede aplicar diferentes algoritmos de clasificación o de selección de rasgos sobre un mismo conjunto de datos.

- Define dominios en variables descriptivas, y manejo intrínseco de los dominios de datos más comunes (enteros, booleanos, reales, etc.), con posibilidad para su extensión atendiendo a las necesidades propias del problema.

CEPAR, en la actualidad, no posee algoritmos que permitan trabajar con grandes volúmenes de datos, ni con problemas de clasificación semisupervisada, pero admite que se le puedan agregar nuevos algoritmos para tratar este tipo de problemas, permitiendo re-utilizar los algoritmos y características que actualmente posee.

1.6 Herramientas y tecnologías utilizadas para la implementación

La herramienta CEPAR, fue desarrollada en el lenguaje de programación Java por sus facilidades como lenguaje multiplataforma, y por tal motivo es el que se adopta en este trabajo para la implementación del algoritmo. De igual forma se emplea el NetBeans como Entorno de Desarrollo Integrado (IDE), al estar declarado como el IDE de la herramienta. Además, para el modelado del diseño del diagrama de flujo del algoritmo y en la modelación de los diagramas de clases del algoritmo se emplea el lenguaje de modelado unificado (UML) con la utilización de la herramienta Visual Paradigm.

1.6.1 Lenguaje de programación JAVA

Java en su versión 8 es un Lenguaje de programación orientado a objetos (POO), de alto nivel y semicompilado que funciona con una máquina virtual. Puede ser ejecutado en la mayoría de los sistemas operativos. Este leguaje tiene un alto rendimiento ya que las optimizaciones integradas para entornos multiproceso lo hacen aún más rápido (Jendrock 2014).

El modelo Java para la gestión de la memoria, los procesos múltiples y la gestión de excepciones lo convierte en un lenguaje fácil de entender y eficaz para los desarrolladores novatos. Además, ofrece un entorno de aplicaciones avanzado con un alto nivel de seguridad que es idóneo para las aplicaciones de red (Jendrock 2014).

1.6.2 Entorno de Desarrollo Integrado NetBeans

NetBeans (en su versión 8.2) es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma se encuentran (González 2015):

- Administración de las interfaces de usuario (ej. menús y barras de herramientas).

- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- Administración de ventanas.

Otra característica es que NetBeans es compuesto por paquetes y que algunos pueden ser descargados individualmente.

1.6.3 Lenguaje de Modelado Unificado

El Lenguaje Unificado para la Construcción de Modelos (UML, por sus siglas en inglés) se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es un sistema de notación (que, entre otras cosas, incluye el significado de sus notaciones) destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Además es un lenguaje para construir modelos; no guía al desarrollador en la forma de realizar el análisis y diseño orientados a objetos ni le indica cual proceso de desarrollo adoptar (Rumbaugh, Jacobson et al. 2000). Por las razones antes expuestas se emplea este lenguaje para el modelado.

1.6.4 Herramienta CASE Visual Paradigm

Las herramientas CASE[2] se destinan a automatizar los aspectos claves de todo el proceso de desarrollo de un sistema. Existen cuatro razones para adoptar las herramientas CASE: incrementar la productividad del analista, mejorar la comunicación entre analistas y usuarios, integrar las actividades del ciclo de vida y analizar y valorar el impacto de los cambios en el mantenimiento (López, López et al. 2011).

Visual Paradigm en su versión 8.0 es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. (Pressman 2010)

Soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un software. Esta herramienta permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software, además generar diversos informes a partir de la información introducida en la misma (Torres and González 2012).

Se caracteriza por: (Pressman 2010)

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

[2] Ingeniería de software asistida por computadora (por sus siglas en inglés CASE)

- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, con diferentes especificaciones.
- Licencia: gratuita y comercial.
- Soporta aplicaciones Web.
- Varios idiomas.

Por las características y funcionalidades antes expuesta se empleará esta herramienta para el modelado del sistema.

Conclusiones parciales del Capítulo

A partir del estudio y análisis de los conceptos fundamentales de la ciencia Reconocimiento de Patrones, de la definición del enfoque Lógico Combinatorio y del estudio de los problemas de clasificación supervisada se obtiene la fundamentación teórica-conceptual de la investigación.

El análisis de los algoritmos para solucionar los problemas de clasificación supervisada en el RLCP basados en las precedencias parciales, específicamente el algoritmo KORA, así como la herramienta CEPAR permiten determinar que por su diseño modular posibilita la implementación de dicho algoritmo, a través del lenguaje de programación Java y del lenguaje de modelado unificado utilizando las herramientas Netbeans y VisualParagdim.

CAPÍTULO 2 DESCRIPCIÓN E IMPLEMENTACIÓN DEL ALGORITMO KORA

Introducción

En este capítulo se describen las características y el funcionamiento del algoritmo KORA y su extensión el algoritmo KORA- Ω partiendo de las definiciones fundamentales, las etapas de desarrollo y el análisis de complejidad, así como la implementación de los algoritmos para ser integrados en la herramienta CEPAR.

2.1 El principio de precedencias parciales de Zhuravlev

Las precedencias parciales constituyen una significativa forma de pensar en las ciencias naturales, de donde la tomó uno de sus iniciadores, el ruso Yuri Ivanovich Zhuravlev. En un inicio, el modelo se concibe en forma muy simplificada (en comparación con las exigencias de los problemas prácticos en las citadas disciplinas), como modelo Booleano (Shulcloper 2009).

Si se analiza el proceder de diferentes especialistas en diversos campos de las ciencias naturales se puede apreciar, sin la pretensión de modelar el pensamiento humano pero sí teniendo en cuenta algunas de sus manifestaciones externas, que casi en ningún caso se tienen en cuenta todos, de una vez, los rasgos que describen un objeto o fenómeno dado, sino más bien existe una tendencia al análisis de subconjuntos de rasgos (como proyecciones a subespacios de dimensiones más pequeñas que las del espacio inicial). Sobre la base de esos subconjuntos de rasgos, se llegan a conclusiones parciales, temporales, que permitan posteriormente, con la consecución de otras conclusiones de este tipo con otros subconjuntos de rasgos, llegar a una conclusión final. Es decir, se presupone que estos problemas tienen implícita una cierta precedencia parcial que permite el análisis por partes y posteriormente totalizarlos en una decisión final (Shulcloper 2009).

2.2 Algoritmo KORA-3

El algoritmo KORA-3 se basa en las precedencias parciales antes expuestas. Fue elaborado en la década de los 60s por un grupo de colaboradores de Instituto de Problemas de Transmisión de la Información de la AC URSS, dirigido por M. M. Bongard, con el fin de utilizar el programa para la solución de problemas de Geofísica y Geología (Shulcloper 2009) (Ortiz-Posadas, Martínez-Trinidad et al. 1996) (Luna and Riverón).

El algoritmo de clasificación KORA-3 consiste en: primero obtener las propiedades complejas que describen a las clases y después utilizar estas para clasificar nuevos objetos; un objeto será ubicado en la clase que aporte más propiedades complejas, que se cumplan para el objeto que se está clasificando.

El algoritmo está basado en la idea de que las clases están formadas por objetos que cumplen ciertas propiedades complejas, compuestas por la conjunción de tres propiedades simples. Trabaja sobre la base de conceptos de analogía, ya que se trabaja con una función de semejanza, las precedencias parciales porque se comparan subdescripciones de objetos, frecuencia puesto que una propiedad compleja sirve para describir una clase si se presenta suficientes veces en ella y utiliza en particular un de **sistema de conjuntos de apoyo** (Shulcloper 2009) (Shulcloper and Cortés 1999) (Doria 1994).

Definición: Un **sistema de conjuntos de apoyo**, el cual denotaremos por {Ω}, es un conjunto de subconjuntos de rasgos, el cual servirá para indicar que partes de los objetos serán comparadas en las diferentes etapas de aprendizaje. Cada $\Omega\epsilon\{\Omega\}$ es llamado un **conjunto de apoyo**. KORA-3 utiliza todos los posibles conjuntos de apoyo de 3 rasgos. (Doria 1994) (Ochoa 2015)

Por una propiedad simple, entenderemos un valor específico, para alguno de los atributos que describen a los objetos, estos atributos, que se usan para describir a los objetos, en adelante serán llamados rasgos. Una propiedad compleja en KORA-3, será llamada rasgo complejo

2.2.1 Rasgos Complejos

Definición: Sea $\Omega = \{x_{i_1}, x_{i_2}, x_{i_3}\}$ un subconjunto de 3 rasgos, y (a_1, a_2, a_3) una combinación de valores para $x_{i_1}, x_{i_2}, x_{i_3}$ respectivamente; entonces (a_1, a_2, a_3) y $\{x_{i_1}, x_{i_2}, x_{i_3}\}$ forman un **rasgo complejo** de la clase K_i si y sólo si el triplo (a_1, a_2, a_3) aparece al menos v_i veces en las Ω-partes de los objetos de K_i y no aparece en la Ω-partes de los objetos de la otra clase (Shulcloper 2009).

Aquellos objetos que tienen esta combinación de valores en la Ω-parte correspondiente, serán llamados **objetos caracterizados** por este rasgo complejo.

Notación. - Se pueden denotar los rasgos complejos como un arreglo de n elementos que tiene los valores de los rasgos, que intervienen en el rasgo complejo, en el lugar (orden) correspondiente a cada rasgo, y "– " en el lugar correspondiente a los rasgos que no intervienen en el rasgo complejo.

Definición: Se denomina **restos de la clase $\mathbf{K_i}$** a los objetos que son caracterizados por menos de $\eta_i > 0$ rasgos complejos (Shulcloper 2009). Para comprender mejor esta definición se realizó una entrevista al especialista del tema, el Dr. Cs. José Ruíz Shulcloper (Anexo 1), que permitió definir a η_i como un umbral que precisa cuantos rasgos complejos deben caracterizar a un objeto para que este no pertenezca al resto de la clase.

Definición: Dos **rasgos complejos son equivalentes** si y sólo si caracterizan a exactamente los mismos objetos (Shulcloper 2009).

Definición: Un **rasgo complejo** A se dice que es **más fuerte** que el rasgo complejo B si y sólo si el rasgo complejo A caracteriza a todos los objetos caracterizados por el rasgo complejo B, y al menos uno más (Shulcloper 2009).

2.2.2 Rasgos Complejos Complementarios

Definición: Un subconjunto de rasgos y una combinación de valores forman un **rasgo complejo complementario** si y sólo si la combinación de valores aparece al menos v_i' veces en las Ω-partes de los restos de K_i y ninguna vez en las Ω-partes de los objetos de la otra clase. $v_i' < v_i$, donde v_i es el parámetro utilizado para calcular los rasgos complejos (Shulcloper 2009)

2.2.3 Etapas del algoritmo KORA-3

El algoritmo consta de 3 etapas para realizar la clasificación: (Shulcloper 2009)

1. **Etapa de Aprendizaje.**
   - En esta etapa se necesita definir los parámetros $v_1\ y\ v_2$, para K_1 y K_2 respectivamente.
   - A continuación, se analizan todos los subconjuntos de 3 rasgos, para buscar los rasgos complejos para cada clase.
   - Cuando se encuentra un rasgo complejo, se almacena si y sólo si entre los ya almacenados, no existe un rasgo complejo más fuerte o equivalente. Si el nuevo rasgo complejo es más fuerte que alguno de los almacenados anteriormente, éstos deben ser eliminados.
2. **Etapa de Reaprendizaje.**
   - En esta etapa se necesita definir los parámetros η_1 y η_2, para K_1 y K_2 respectivamente, para poder calcular los restos de cada una de las clases.
   - Después se definen nuevos $v_1' < v_1$ y $v_2' < v_2$ y se repite el proceso anterior para encontrar los denominados rasgos complejos complementarios.
   - Esta etapa de reaprendizaje puede repetirse varias veces, definiendo nuevos valores para los parámetros. Este proceso es finito ya que $v_i' < v_i$.
3. **Etapa de Clasificación.**
   - Para la clasificación de nuevos objetos, se cuenta cuántos rasgos complejos de cada clase caracterizan (votan a favor) del nuevo objeto, y se elige la clase que aporte más rasgos complejos.

Para una mejor comprensión en la Figura 7 se muestra el diagrama de flujo del algoritmo utilizado como guía para la implementación del código.

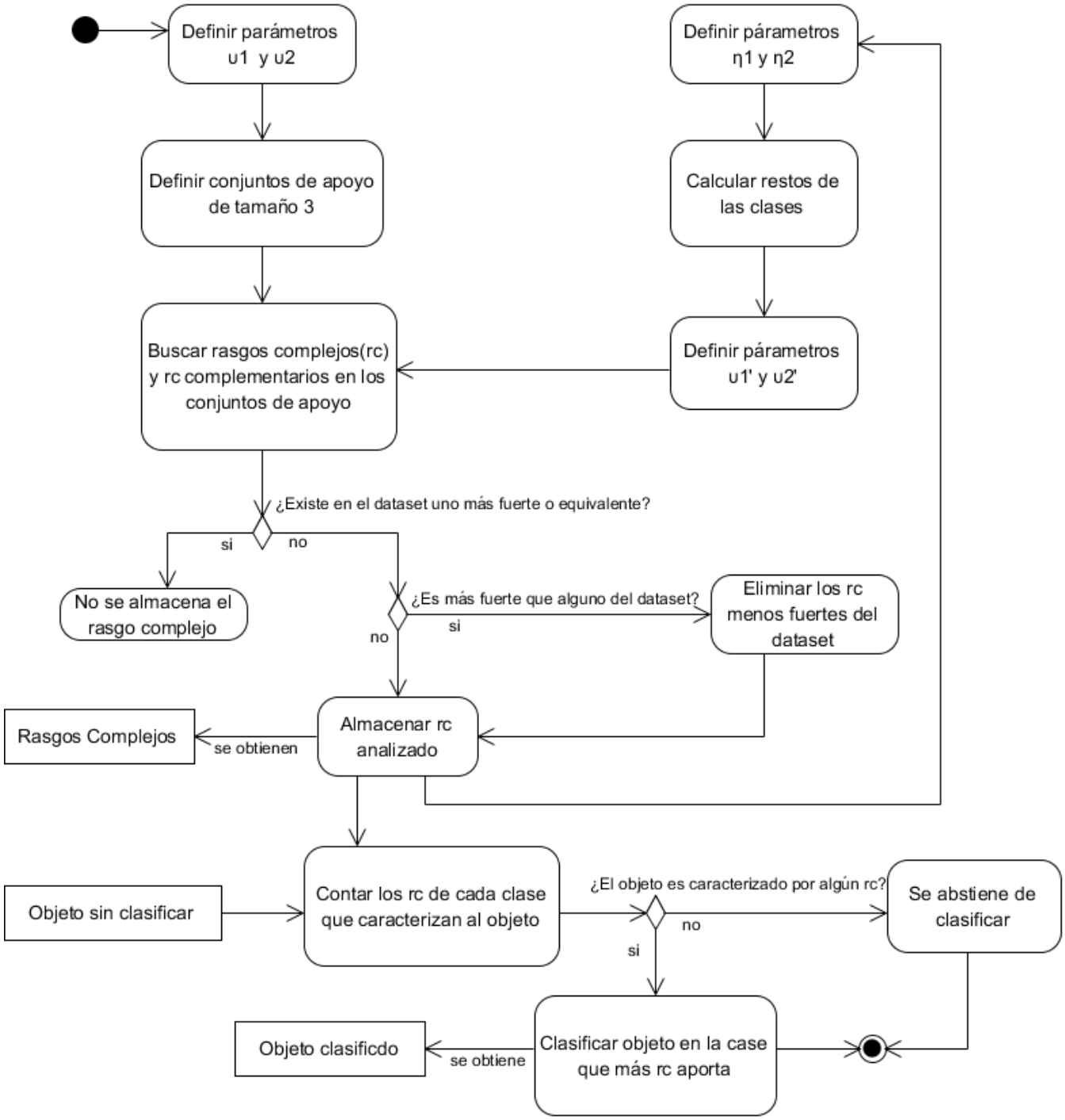

Figura 7 Diagrama de flujo del algoritmo KORA-3 (elaboración propia)

En la etapa de aprendizaje como en la de reaprendizaje se eliminan rasgos complejos (o complementarios) siguiendo los criterios de rasgos equivalentes y rasgos más fuertes. Sin embargo, no es difícil mostrar con ejemplos que esto provoca inconsistencias en el proceso de clasificación, es decir, objetos que podrían ser

clasificados por lo que no es difícil concluir que no hay tal equivalencia y que la definición de rasgos equivalentes y rasgos más fuertes, si bien simplifican el trabajo del algoritmo, resulta inconveniente para la eficacia del mismo.

2.2.4 Análisis de Complejidad

Para el cálculo de la complejidad temporal es fundamental el problema del cálculo de los rasgos complejos, puesto que la clasificación de un objeto es lineal con respecto al número de rasgos complejos obtenidos, y no se necesita más espacio que el requerido para almacenar la descripción del objeto.

Sean:

n el número de rasgos del problema,

m el número de objetos de la matriz de aprendizaje,

t el número de clases,

m_i el número de objetos en la clase K_i, i=1,…,t.

Como se ocupan todas las combinaciones posibles de 3 rasgos, en total se tienen:

$$\binom{n}{3} = \frac{n!}{(n-3)!\,3!} = \frac{n(n-1)(n-2)}{6}$$

conjuntos de apoyo, y como cada uno está formado por 3 rasgos se tienen que verificar, para decidir si son rasgos complejos, cada una de las 8 posibles combinaciones de valores, por lo cual el total de candidatos a rasgo complejo analizados son: (Doria 1994)

$$\frac{8n(n-1)(n-2)}{6}$$

Cuando se está trabajando con la clase K_i, cada uno de estos candidatos se tiene que comparar contra todos los objetos de la matriz de aprendizaje, para saber si se presenta suficientes veces dentro de la clase, y ninguna fuera de la misma, y como esto se hace para cada candidato, en total se hacen:

$$\frac{8mn(n-1)(n-2)}{6}$$

comparaciones entre Ω-partes de objetos (Doria 1994). Además, esto tiene que hacerse en cada una de las etapas de aprendizaje, y como el máximo de etapas de aprendizaje es m_i puesto que el valor inicial para v_1 es a lo más m_i, ya que una combinación de valores no puede presentarse más de m_i veces en K_i, y en

el peor de los casos v_1 va decrementándose de uno en uno, entonces tenemos que en total durante todo el proceso de aprendizaje se hacen:

$$\frac{8m_i mn(n-1)(n-2)}{6}$$

comparaciones entre Ω-partes de objetos (Doria 1994). Como este proceso se realiza para cada una de las clases, tenemos que el total de comparaciones entre Ω-partes de objetos es:

$$\sum_{i=1}^{t} \frac{8m_i mn(n-1)(n-2)}{6} = \frac{8mn(n-1)(n-2)}{6} \sum_{i=1}^{t} m_i$$

$$= \frac{8m^2 n(n-1)(n-2)}{6}$$

puesto que:

$$\sum_{i=1}^{t} m_i = m$$

Por otro lado, como cada combinación sólo puede ser rasgo complejo de una sola clase, entonces el máximo número de rasgos complejos que se pueden obtener es:

$$\frac{8n(n-1)(n-2)}{6}$$

Concluyendo, para un problema den rasgos y m objetos, **la complejidad en el tiempo** es de

$$\frac{8m^2 n(n-1)(n-2)}{6}$$

comparaciones entre Ω -partes de objetos, y **la complejidad en el espacio** es de:

$$\frac{8n(n-1)(n-2)}{6} T(rc)$$

donde T(rc) es el espacio necesario para almacenar un rasgo complejo (Doria 1994).

2.2.5 Limitaciones

Según (Doria 1994) y (Shulcloper 2009) KORA-3 se aplica a problemas de clasificación que cumplan con las hipótesis siguientes:

I) Debe ser de clasificación supervisada, es decir, se tiene un universo dividido en clases, con una muestra de objetos para cada clase, y que consiste en decidir, para un objeto, a que clase pertenece.

2) El universo considerado debe estar dividido únicamente en 2 clases. Por ejemplo, en los que se quiere determinar si un objeto es, o no, de un cierto tipo, digamos zona perspectiva para un cierto mineral o no perspectiva.

3) Cada objeto, de la muestra, debe estar descrito en términos de rasgos booleanos, es decir, rasgos que únicamente nos indiquen si el objeto tiene una cierta característica o no.

4) No se admite ausencia de información.

5) Las clases en que se divide el universo deben ser disjuntas y duras, es decir, basados en la Teoría Clásica de Conjuntos.

6) Todos los rasgos y los objetos deben ser igualmente informativos, así como todas las combinaciones de 3 rasgos y tres valores, también deben ser igualmente informativas, por lo que los criterios de comparación de los rasgos todos son iguales (criterio de igualdad); y también la función de semejanza que se emplea es aquella que toma el valor 1 sólo cuando todos los rasgos son coincidentes

2.2.6 Restricciones

Como antes se menciona, el algoritmo KORA-3 presenta un conjunto de limitaciones para su aplicación en muchos problemas de la práctica profesional, en particular en algunas zonas poco formalizadas del conocimiento.

De este modelo de algoritmos se presenta en (Doria 1994) y (Shulcloper 2009), además de las ya mencionadas, las restricciones siguientes:

- todos los rasgos complejos y todos los rasgos complejos complementarios se suponen igualmente informativos para la clasificación;
- existen ejemplos, en el que el orden en que se eliminen los rasgos complejos equivalentes y/o los menos fuertes, altera la clasificación de nuevos objetos;
- todos los rasgos y objetos de la matriz de aprendizaje son también igualmente informativos (tener la misma importancia desde el punto de vista de la clasificación), así como todas las combinaciones de 3 rasgos y tres valores, también deben ser igualmente informativas y;
- los rasgos complejos son combinaciones exclusivamente de tres valores de sendos rasgos.

Por lo que se definen extensiones del algoritmo con la idea de contrarrestar esta situación.

2.3 Extensión del Algoritmo KORA-3: el Algoritmo KORA-Ω

El modelo de algoritmos KORA-Ω (Figura 8), consiste en permitir el uso de cualquier sistema de conjuntos de apoyo. Para este algoritmo siguieron un conjunto de redefiniciones de los conceptos relativos a este

modelo con sus correspondientes adecuaciones a la idea central del algoritmo que lo transformaron en el modelo de algoritmos KORA-Ω, en el cual todas las restricciones antes mencionadas fueron salvadas y resueltos los problemas computacionales que las nuevas extensiones implicaron (Shulcloper 2009).

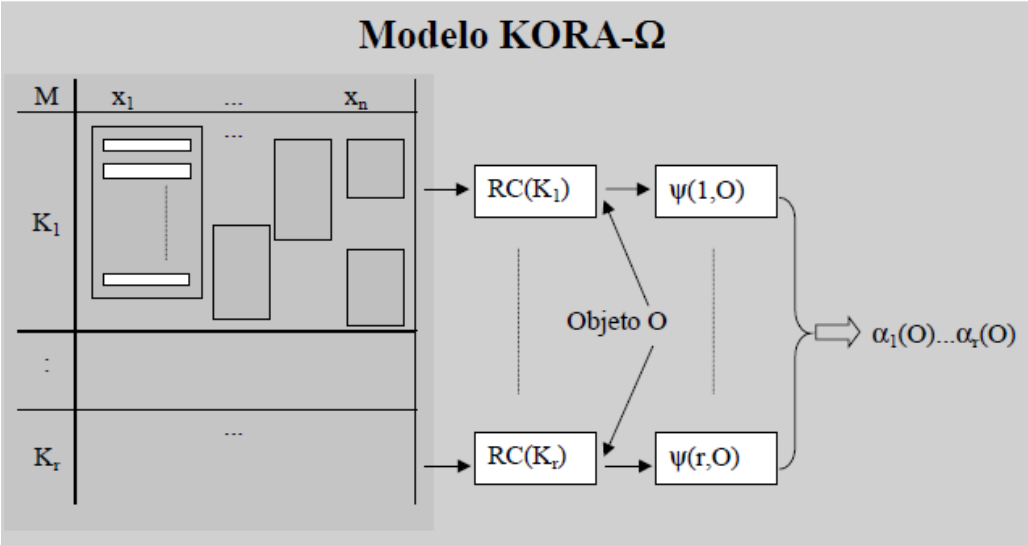

Figura 8 Modelación del funcionamiento del algoritmo KORA-Ω (Ochoa 2015)

Sea R el conjunto de los rasgos en términos de los cuales se definen los objetos de la muestra de entrenamiento M en un problema de clasificación supervisada con r≥2 clases, no necesariamente disjuntas, K_1^i, …, K_r^i . Sea $K_i \subseteq K_i^i$, los objetos de la clase K_i^i presentes en M, i=1,…,r (Shulcloper, Ochoa et al. 2015).

Definición: Supóngase que tanto K_i como K_i^i son subconjuntos duros, i=1,...,r y sea $\Omega = \{\mathrm{x}_{i_1}, \ldots, \mathrm{x}_{i_p}\}$. El **complemento de una clase** K_i, el cual denotaremos como CK_i está definido como $\mathrm{CK}_i = \{O_j | \alpha_i(O_j) = 0\}$, siendo $\alpha_i(O_j)$ el valor de la i-ésima componente del r-uplo informacional del objeto O_j (Shulcloper, Ochoa et al. 2015).

Definición: Sea $\Omega = \{\mathrm{x}_{i_1}, \ldots, \mathrm{x}_{i_p}\}$ un conjunto de apoyo, y $(\mathrm{a}_1, \ldots, \mathrm{a}_p)$ una combinación de valores para $\mathrm{x}_{i_1}, \ldots, \mathrm{x}_{i_p}$respectivamente, entonces $(\mathrm{a}_1, \ldots, \mathrm{a}_p)$ y $\{\mathrm{x}_{i_1}, \ldots, \mathrm{x}_{i_p}\}$ forman un **rasgo β_i-complejo** de la clase K_i si y sólo si: (Shulcloper, Ochoa et al. 2015) (Godoy-Calderón, Calvo et al. 2010)

1) $\exists O \in \mathrm{K}_i [\, x_{i_1}(O) = \mathrm{a}_1 \wedge \ldots \wedge x_{i_p} = \mathrm{a}_p]$
2) $\sum_{O \in K_i} \Gamma\left(\Omega O, (a_{1,\ldots,} a_p)\right) \geq \beta_i$
3) $\sum_{O \in CK_i} \Gamma\left(\Omega O, (a_{1,\ldots,} a_p)\right) \leq \lambda_i$ para i=1,…, r.

Se denotará por RC(K_i) al conjunto de todos los rasgos β_i-complejos de la clase K_i.

Definición: Un objeto $O \epsilon K_i$ será denominado $\boldsymbol{\delta_i}$**-resto** si y sólo si:

$$\sum_{\Omega rc \in RC(K_i)} \Gamma(\Omega O, \Omega rc) < \delta_i$$

Para simplificar la notación, denotaremos por r(K_i) al conjunto de todos los δ_i-restos de la clase K_i (Shulcloper, Ochoa et al. 2015).

Definición: $(\mathrm{a}_1, \ldots, \mathrm{a}_p)$ y $\{\mathrm{x}_{i_1}, \ldots, \mathrm{x}_{i_p}\}$ forman un **rasgo β_i-complejo complementario** de la clase K_i si y sólo si: (Shulcloper, Ochoa et al. 2015)

1) $\exists O \in \mathrm{r}(\mathrm{K}_i)[\, x_{i_1}(O) = \mathrm{a}_1 \wedge \ldots \wedge x_{i_p}(O) = \mathrm{a}_p]$
2) $\sum_{O \in r(K_i)} \Gamma\left(\Omega O, \left(a_{1,\ldots,} a_p\right)\right) \geq \beta_i'$
3) $\sum_{O \in CK_i} \Gamma\left(\Omega O, \left(a_{1,\ldots,} a_p\right)\right) \leq \lambda_i$ para i=1,…,r siendo $\beta_i' < \beta_i$

2.3.1 Ponderación de Rasgos Complejos

El algoritmo KORA-3 asume el hecho de que todos los rasgos complejos y complejos complementarios tienen la misma importancia informacional. Esto es de esperar que no ocurra debido a que cada rasgo por si sólo puede tener distinta importancia informacional, además, no todos los rasgos complejos caracterizan a los mismos objetos.

Por las razones expuestas, el algoritmo KORA-Ω introduce una manera de ponderar los votos provenientes de diferentes rasgos complejos.

Definición: Sea rc un rasgo complejo, para la clase K_j, formado por $(\mathrm{a}_1, \ldots, \mathrm{a}_p)$ y $\Omega = \{\mathrm{x}_{i_1}, \ldots, \mathrm{x}_{i_p}\}$ entonces la **ponderación de un rasgo complejo** rc, la cual denotaremos por P(rc), estaría dada por: (Doria 1994)

$$\sum_{x_p \epsilon \Omega} P(x_p) \sum_{O_i \epsilon OC_j(rc)} P_j(O_i)$$

donde:

$P(x_p)$ es el peso informacional del rasgo x_p.

$P_j(O_i)$ es el peso informacional del objeto O_i en la clase K_j.

$OC_j(rc)$ es el conjunto de los objetos, de K_j, caracterizados por rc.

Los rasgos complejos complementarios, a pesar de que poseen un peso diferente al de un rasgo complejo, utilizan esta fórmula para calcular su ponderación; pues su valor siempre será menor por el hecho de que caracterizan menos objetos.

Con esta ponderación, cuando se quiere clasificar un nuevo objeto O, se puede calcular cuál es **la votación que obtiene el objeto O en la clase** K_i, la cual se denota por $\Gamma_i(O)$, de la manera siguiente:

$$\Gamma_i(O) = \sum_{rc \in RC_i(O)} P(rc)$$

donde $RC_i(O)$ es el conjunto de los rasgos complejos, de la clase K_i, que caracterizan a O.

Finalmente el objeto se clasificaría en la clase que obtenga la votación más alta, si más de una clase obtiene la votación más alta, el método se abstiene de clasificar al objeto o multiclasifica (Shulcloper, Ochoa et al. 2015).

En este punto vale la pena mencionar que la regla de solución que con frecuencia es utilizada es la de mayoría simple, pero el modelo no se restringe a ella, es decir, podría utilizarse cualquier otra regla de decisión.

Por otra parte, el modelo extendido admitirá cualquier tipo de variable, no sólo las Booleanas, e incluso pudieran ser las descripciones heterogéneas, es decir, con datos mezclados e incompletos, pues la ausencia de información en este modelo es admisible.

Esto conlleva a que los criterios de comparación de valores de cada una de los rasgos no tienen que ser los mismos, ni tienen que ser la igualdad estricta.

Esta extensión a cualquier tipo de criterio de comparación obliga a incluir la condición 1) en la definición de rasgo βi-complejo (complementario) ya que de lo contrario el conjunto de rasgos complejos (complementarios) podría ser infinito. Por ello, sólo se considerarán como candidatas para ser rasgos complejos (complementarios) las combinaciones de valores de los rasgos que aparecen en la matriz de entrenamiento. Estos rasgos complejos (complementarios) se denominan restringidos (Shulcloper 2009).

De igual manera el modelo admitirá cualquier tipo de función de similaridad para comparar las subdescripciones de los objetos, que como ya se dijo, no tienen que ser de tres rasgos exclusivamente.

Con estas definiciones, se puede trabajar con el resto del algoritmo de clasificación KORA-Ω, sin tener que hacerle más cambios respecto al algoritmo KORA-3, excepto los correspondientes parámetros introducidos: umbrales, funciones de semejanza, criterios de comparación de cada uno de los rasgos.

Para una mejor comprensión en la Figura 9 se muestra el diagrama de flujo del algoritmo utilizado como guía para realizar la implementación del código.

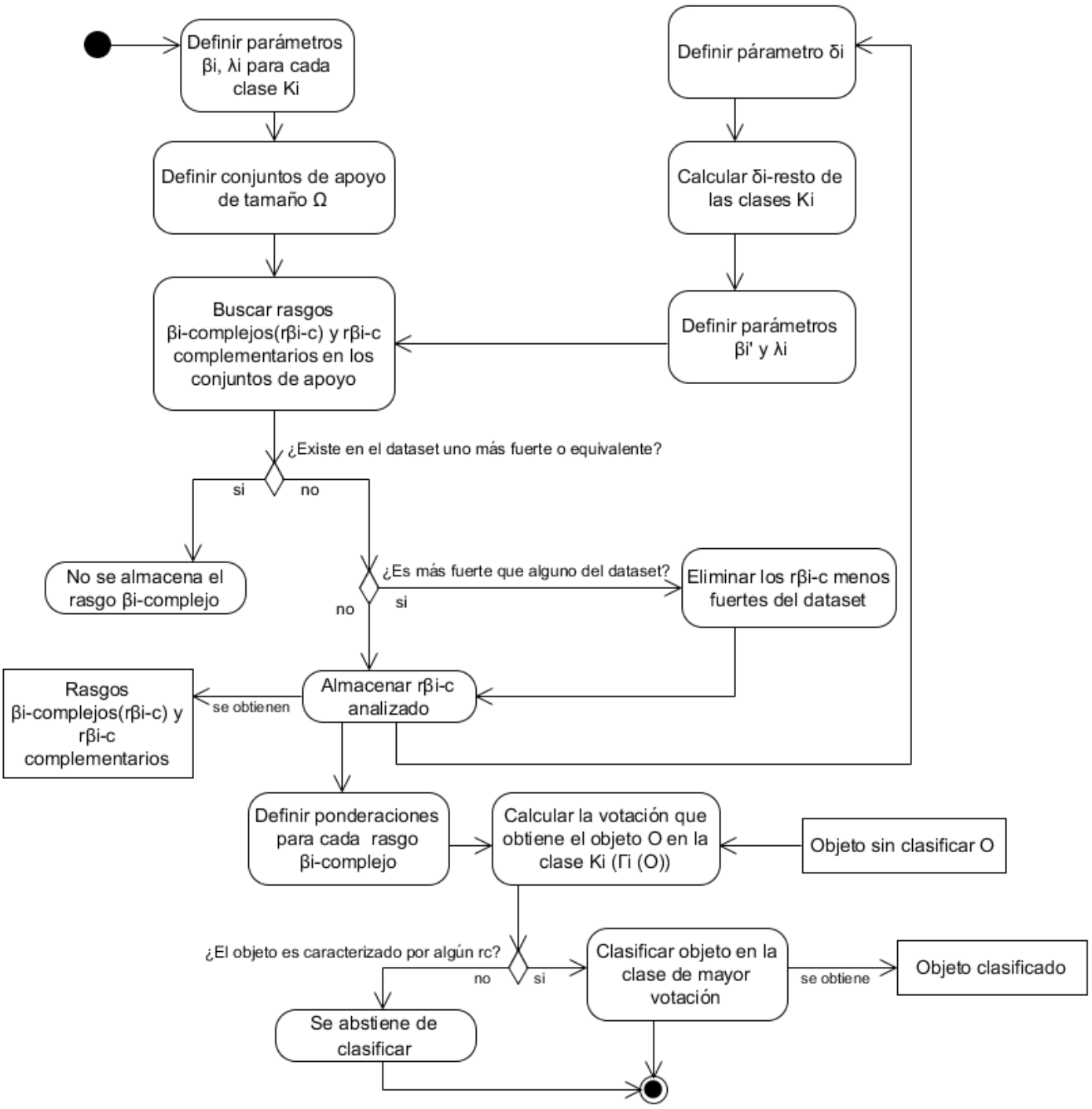

Figura 9 Diagrama de flujo del algoritmo KORA-Ω (elaboración propia)

Todas estas extensiones fueron motivadas por situaciones que se presentaron durante la solución de problemas de clasificación supervisada, dentro de la Geofísica y la Geología.

Las características originales de KORA-3, así como las de la extensión de algoritmos KORA-Ω, se engloban en tabla siguiente: (Shulcloper 2009)

Tabla 1 Características del algoritmo KORA-3 y su extensión KORA- Ω

Características	KORA-3	KORA-Ω
Más de Dos Clases	No	Si
Clases Disjuntas	Si	Si
Clases No Disjuntas	No	Si
Rasgos Booleanos	Si	Si
Rasgos No Booleanos	No	Si
Criterio de Comparación por Igualdad	Si	Si
Distintos Criterios de Comparación	No	Si
Función de Semejanza por Igualdad	Si	Si
Distintas Funciones de Semejanza	No	Si
Ausencia de Información	No	Si
Ponderación de Rasgos Complejos	No	Si
Distintos Conjuntos de Apoyo	No	Si

2.3.1 Análisis de Complejidad

Para el cálculo de la complejidad es fundamental el problema del cálculo de los rasgos β_i-complejos, puesto que la clasificación de un objeto es lineal con respecto al número de rasgos complejos obtenidos, y no se necesita más espacio que el requerido para almacenar la descripción del objeto.

Sean:

n el número de rasgos del problema,

m el número de objetos de la matriz de aprendizaje,

t el número de clases

m_i el número de objetos en la clase K_i, i=1,…,t

k el número de conjuntos de apoyo que se utilizará.

Cuando se está trabajando con la clase K_i, dado que los rasgos complejos están restringidos a valores existentes en la clase, para cada conjunto de apoyo se tienen que verificar, para decidir si son rasgos complejos, m_i posibles combinaciones de valores, una por cada objeto de K_i, por lo cual el total de candidatos a rasgo complejo analizados son: (Doria 1994)

$$km_i$$

Cada uno de estos candidatos se tiene que comparar contra todos los objetos de la matriz de aprendizaje, para saber si se presenta suficientes veces dentro de la clase, y ninguna fuera de la misma, y como esto se hace para cada candidato, en total se hacen:

$$kmm_i$$

comparaciones entre Ω-partes de objetos (Doria 1994). Además, esto tiene que hacerse en cada una de las etapas de aprendizaje, y como el máximo de etapas de aprendizaje es m_i, puesto que el valor inicial para β_i es a lo más m_i, ya que la semejanza total de una combinación de valores, contra los objetos de la clase K_i, no puede ser mayor que m_i, y en el peor de los casos β_i va decrementándose de uno en uno, entonces tenemos que en total durante todo el proceso de aprendizaje se hacen:

$$kmm_i^2$$

comparaciones entre Ω-partes de objetos (Doria 1994). Como este proceso se realiza para cada una de las clases, tenemos que el total de comparaciones entre Ω-partes de objetos es:

$$km\sum_{i=1}^{t} m_i^2$$

Por otro lado, como cada combinación sólo puede ser rasgo complejo de una sola clase, entonces el máximo número de rasgos complejos que se pueden obtener es:

$$k\sum_{i=1}^{t} m_i = km$$

puesto que:

$$\sum_{i=1}^{t} m_i = m$$

Concluyendo, para un problema den rasgos, m objetos, t clases, k conjuntos de apoyo y m_i objetos en cada clase K_i, i=1,…,t; **la complejidad en el tiempo** es de: (Doria 1994)

$$km\sum_{i=1}^{t} m_i^2$$

comparaciones entre Ω-partes de objetos, y **la complejidad en el espacio** es de:

$$kmT(rc)$$

donde T(rc) es el espacio necesario para almacenar un rasgo complejo (Doria 1994).

2.4 Integración del algoritmo KORA en la Herramienta CEPAR

En el paquete *cepar.core* de la herramienta CEPAR se encuentran implementadas todas las funcionalidades que permiten el correcto modelo de un problema del RLCP, este paquete además posee todas las clases, interfaces y abstracciones necesarias para facilitar la comunicación entre los distintos paquetes relacionados a cada uno de los problemas del RP. La herramienta cuenta con un paquete destinado a la clasificación supervisada de objetos (*cepar.clasificacionSupervisada*) en el cual se localizan las características necesarias para la correcta implementación del algoritmo para la Clasificación Supervisada. Además, cuenta con un paquete (*cepar.clasificacionSupervisada.util*) para las clases y los métodos auxiliares que fueron necesarios implementar en el desarrollo del código del algoritmo.

En la Figura 10 se puede observar el paquete *cepar.clasificacionSupervisada* sin la implementación de algoritmos ni clases y en la Figura 11 como parte del aporte, se muestra el algoritmo KORA-3 y su extensión KORA-Ω propuesto incorporado a dicho paquete.

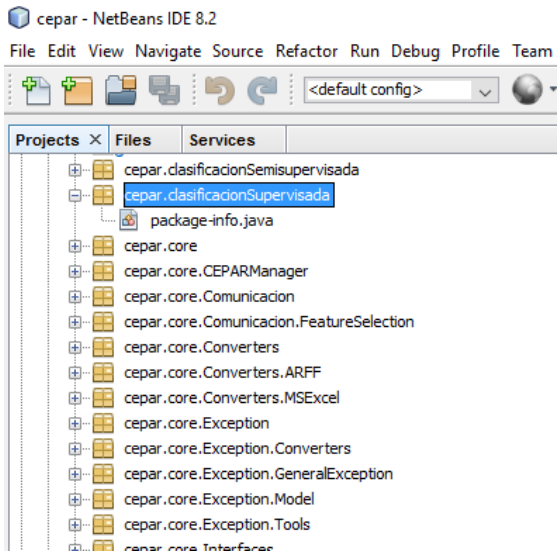

Figura 10 Paquete para la clasificación supervisada de CEPAR

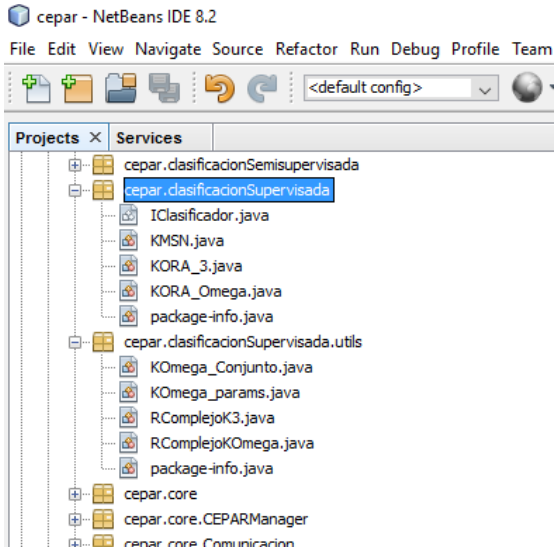

Figura 11 Paquete para la clasificación supervisada de CEPAR con el algoritmo KORA

En los diagramas de clases siguientes se muestra la relación entre el algoritmo KORA-3 (Figura 12) y su extensión KORA-Ω (Figura 13) implementados con las clases incluidas en el código de CEPAR.

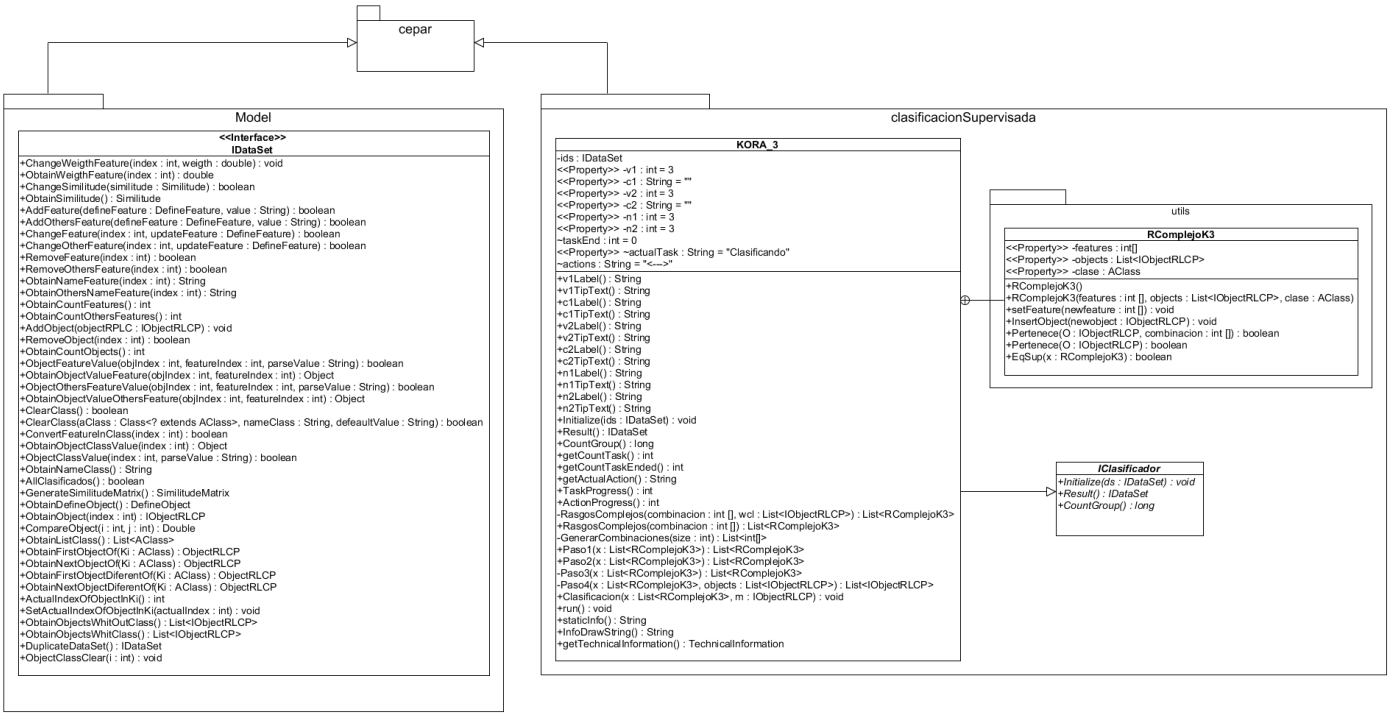

Figura 12 Diagrama de clases del Algoritmo KORA-3

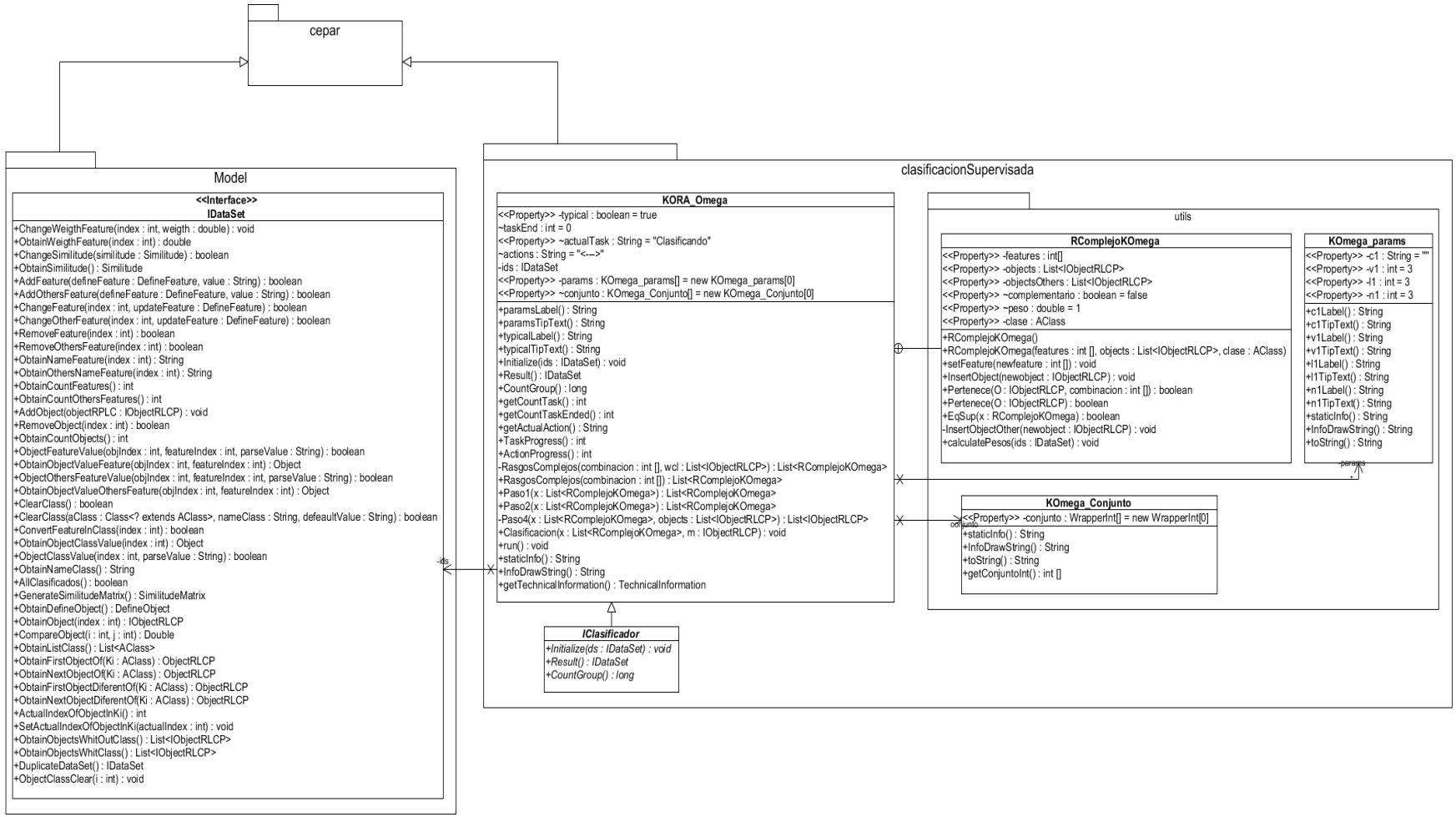

Figura 13 Diagrama de clases del Algoritmo KORA- Ω

Al compilar nuevamente la herramienta se puede observar cómo desde el menú de Clasificación Supervisada, Algoritmos, Seleccionar clasificador (Figura 14) pueden ser elegidos el algoritmo KORA-3 implementado y su extensión KORA-Ω (Figura 15).

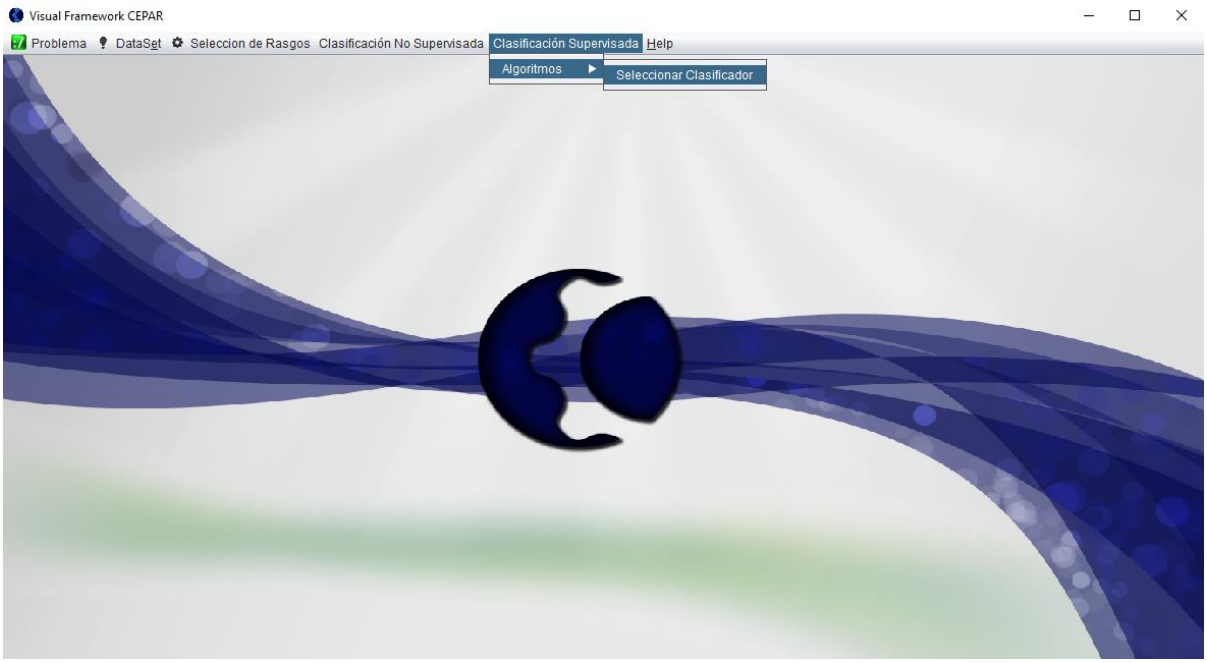

Figura 14 Menú de CEPAR para los problemas de Clasificación Supervisada

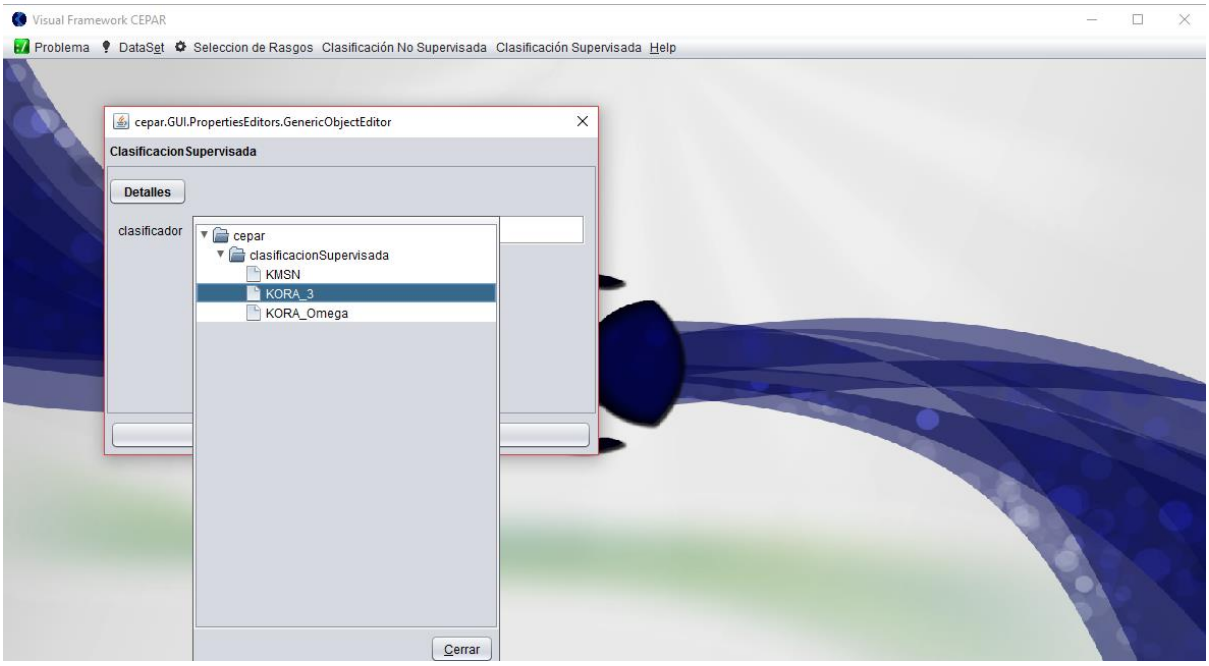

Figura 15 Menú para seleccionar el algoritmo KORA-3 y su extensión KORA- Ω

Los algoritmos implementados trabajan con parámetros; al accionar sobre el nombre del algoritmo se puede observar el formulario para introducir los datos necesarios (Figura 16).

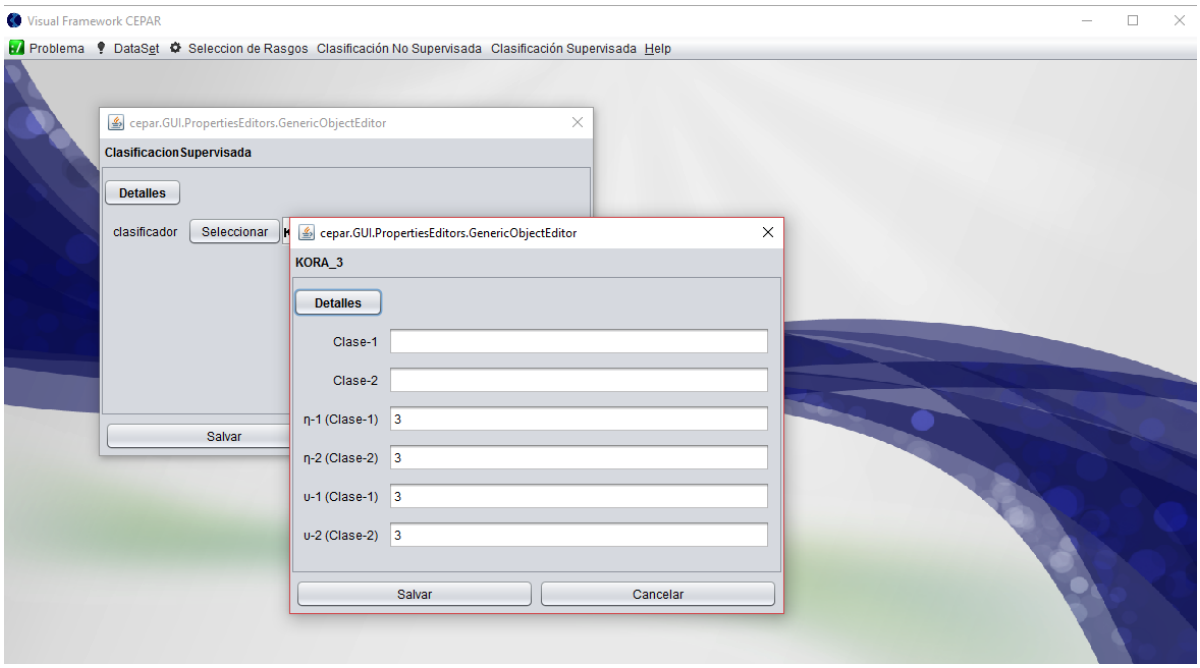

Figura 16 Formulario para introducir los parámetros del algoritmo KORA-3

Como se pudo observar en las imágenes anteriores una vez inicializada la herramienta CEPAR y cargada la base de datos con la que se realizarán los experimentos, el usuario tiene la posibilidad de seleccionar uno de los nuevos algoritmos incluidos en la herramienta.

2.4.1 Funcionalidades implementadas

Para la implementación de los algoritmos KORA-3 y su extensión KORA-Ω se desarrollaron en el paquete *cepar.clasificacionSupervisada* las clases:

KORA_3.java en la que se encuentran programadas, mediante los métodos, las funcionalidades necesarias para el correcto trabajo del algoritmo KORA-3, estas se listan a continuación:

- *GenerarCombinaciones()*: genera las combinaciones posibles de 3 rasgos que conforman el sistema de conjuntos de apoyo.
- *RasgosComplejos()*: teniendo en cuenta los conjuntos de apoyo y los objetos clasificados del dataset[3] realiza el análisis y las comparaciones pertinentes para generar los rasgos complejos los cuales almacena en una lista.
- *Paso1()*: teniendo la lista de los rasgos complejos elimina de esta los rasgos complejos de una clase que se encuentran en otra.

[3] Archivo de datos modificable que contiene los rasgos de los objetos y sus clases.

- *Paso2()*: dada una la lista de rasgos complejos realiza las verificaciones de rasgos complejos equivalentes y más fuertes, depurando de la lista, los que cumplen con las condiciones necesarias para ser eliminados.
- *Paso3()*: a partir de la lista de rasgos complejos y del parámetro definido por υ_i elimina los rasgos complejos que la cantidad de objetos que caracterizan es menor que υ_1.
- *Paso4()*: dada la lista de rasgos complejos y la lista de objetos con clases, a partir del parámetro η_i definido por el usuario, se calculan los objetos que pertenecen al resto de las clases y los almacena en una lista.
- *Clasificación()*: teniendo el objeto a clasificar y la lista de rasgos complejos determina cuántos rasgos complejos de cada clase caracterizan al objeto y le asigna la clase que más rasgos complejos aporta.
- *run()*: método en el cual se integran todas las funcionalidades anteriores siguiendo las etapas de desarrollo definidas del algoritmo KORA-3.

KORA_Omega.java en la que se encuentran programadas, mediante los métodos, las funcionalidades necesarias para el correcto trabajo del algoritmo KORA- Ω, estas se listan a continuación:

- *RasgosComplejos()*: teniendo en cuenta un conjunto de apoyo definido por el usuario y los objetos clasificados del dataset realiza el análisis y las comparaciones pertinentes para generar los rasgos complejos los cuales almacena en una lista.
- *Paso1()*: teniendo la lista de los rasgos complejos y el parámetro λ_i elimina de esta los rasgos complejos de una clase que la cantidad de objetos que caracterizan es mayor que λ_i.
- *Paso2()*: a partir de la lista de rasgos complejos y del parámetro definido por β_i elimina los rasgos complejos que la cantidad de objetos que caracterizan es menor que β_i.y realiza las verificaciones de rasgos complejos equivalentes y más fuertes, depurando de la lista, los que cumplen con las condiciones necesarias para ser eliminados.
- *Paso4()*: dada la lista de rasgos complejos y la lista de objetos con clases, a partir del parámetro δ_i definido por el usuario, se calculan los objetos que pertenecen al resto de las clases y los almacena en una lista.
- *Clasificación()*: teniendo el objeto a clasificar y la lista de rasgos complejos con su peso, determina la votación de cada clase y le asigna al objeto la clase que más rasgos complejos aporta.
- *run()*: método en el cual se integran todas las funcionalidades anteriores siguiendo las etapas de desarrollo definidas del algoritmo KORA-Ω.

Además, en el paquete *cepar.clasificacionSupervisada.util* se implementaron las clases auxiliares: ***KOmega_Conjunto.java***, ***KOmega_params.java***, ***RComplejoK3.java***, ***RComplejoKOmega.java*** que facilitan el manejo de los datos necesarios en las clases principales

2.4.2 Patrones de diseño GRASP

Entre los patrones de diseño se encuentran los patrones GRASP (General Responsibility Assignment Software Patterns). Estos patrones describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades (Autores 2012). Los mismos tuvieron una importante utilidad en el diseño de la propuesta de solución, asignándose a cada clase las tareas que podían realizar según la información que poseía, además de crear las instancias de otras clases en correspondencia con la responsabilidad dada. Los patrones GRASP que se presentaron en el diseño realizado fueron el Experto y el Creador. Con esto se logró conservar el encapsulamiento, pues los objetos logran valerse de su propia información para realizar lo que se les pide. El uso de los patrones Bajo acoplamiento y Alta cohesión permitió la colaboración entre las clases, sin verse afectada la reutilización de las mismas y el entendimiento de estas cuando se encuentran aisladas (Larman 2003).

Experto: Se evidencia en la clase ***KORA_3.java***, que se responsabiliza de crear el objeto de la clase ***RComplejoK3.java*** a partir del método *RasgosComplejos()* el cual construye los objetos de tipo RComplejoK3 para conformar una lista.

Creador: Está evidenciado en la clase ***KORA_Omega.java*** que crea las instancias de las clases ***KOmega_Conjunto.java***, ***KOmega_params.java***, ***RComplejoKOmega.java***.

Bajo Acoplamiento: Se pone de manifiesto en la clase ***RCmplejoKOmega.java*** al asignarle sus responsabilidades, de forma tal que pudiera ser independiente de las demás clases.

Alta Cohesión: Se manifiesta en las clases ***KORA_3.java*** y ***KORA_Omega.java***, cada una presenta y maneja la información que necesita.

Conclusiones parciales del capítulo

El análisis de los algoritmos KORA-3 y de su extensión KORA-Ω, partiendo de las definiciones fundamentales, las etapas de desarrollo y el análisis de complejidad posibilitaron la obtención de los requerimientos necesarios para la implementación y su integración en la herramienta CEPAR.

Como un valor agregado de la investigación se desarrollaron los diagramas de flujo de cada algoritmo, utilizados como guía para la implementación del código.

CAPÍTULO 3 VALIDACIÓN DEL ALGORITMO KORA EN LA HERRAMIENTA CEPAR

Introducción

En el presente capítulo se muestra el funcionamiento de la solución desarrollada a través de los resultados de pruebas de software realizadas al código de los algoritmos KORA-3 y su extensión KORA-Ω implementados e integrados a la herramienta CEPAR.

3.1 Estrategia de Pruebas de Software

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, una estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados.

Una estrategia para la prueba de software debe incluir pruebas de bajo nivel, que son necesarias para verificar que un pequeño segmento de código fuente se implementó correctamente, así como pruebas de alto nivel, que validan las principales funciones del sistema a partir de los requerimientos del cliente (Pressman 2010).

También puede verse en el contexto de la espiral de la Figura 17. La prueba de unidad comienza en el vértice de la espiral y se concentra en cada unidad (por ejemplo, componente, clase o un objeto de contenido de una aplicación) del software implementada en el código fuente. La prueba avanza al moverse hacia afuera a lo largo de la espiral, hacia la prueba de integración, donde el enfoque se centra en el diseño y la construcción de la arquitectura del software. Al dar otra vuelta hacia afuera de la espiral, se encuentra la prueba de validación, donde los requerimientos establecidos como parte de su modelado se validan confrontándose con el software que se construyó. Finalmente, se llega a la prueba del sistema, donde el software y otros elementos del sistema se prueban como un todo (Pressman 2010).

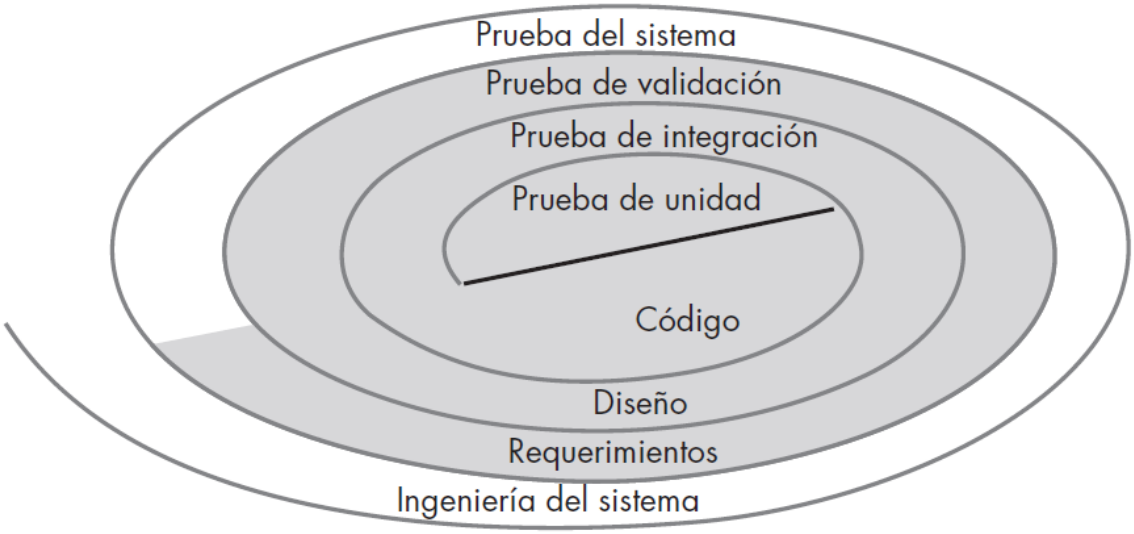

Figura 17 Estrategias de Pruebas de Software (Pressman 2010)

Al considerar el proceso desde un punto de vista procedural, las pruebas dentro del contexto de la ingeniería del software en realidad son una serie de cuatro pasos que se implementan de manera secuencial. Éstos se muestran en la **¡Error! No se encuentra el origen de la referencia.**. Inicialmente, las pruebas se enfocan e ncada componente de manera individual, lo que garantiza que funcionan adecuadamente como unidad. De ahí el nombre de prueba de unidad. Esta prueba utiliza mucho de las técnicas de prueba que ejercitan rutas específicas en una estructura de control de componentes para asegurar una cobertura completa y la máxima detección de errores (Pressman 2010).

A continuación, los componentes deben ensamblarse o integrarse para formar el paquete de software completo. La prueba de integración aborda los conflictos asociados con los problemas duales de verificación y construcción de programas. Durante la integración, se usan más las técnicas de diseño de casos de prueba que se enfocan en entradas y salidas, aunque también pueden usarse técnicas que ejercitan rutas de programa específicas para asegurar la cobertura de las principales rutas de control. Después de integrar (construir) el software, se realiza una serie de pruebas de orden superior. Deben evaluarse criterios de validación (establecidos durante el análisis de requerimientos). La prueba de validación proporciona la garantía final de que el software cumple con todos los requerimientos informativos, funcionales, de comportamiento y de rendimiento (Pressman 2010).

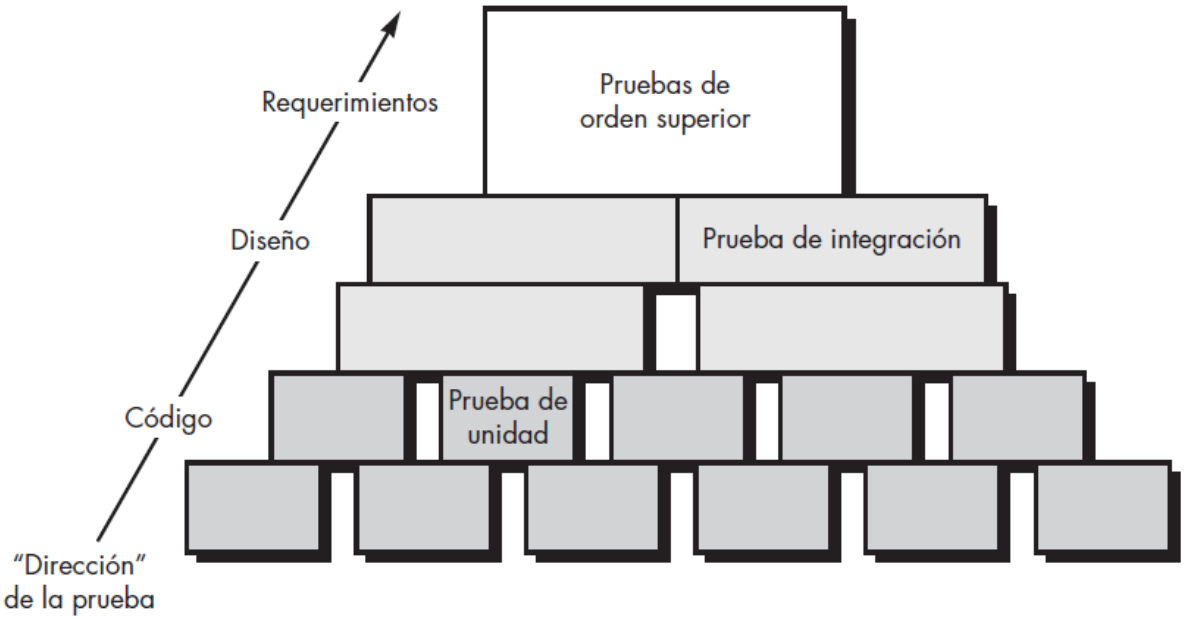

Figura 18 Pasos de las Pruebas de Software (Pressman 2010)

Por las características de la propuesta de solución de la presente investigación se emplean las pruebas de unidad y de integración.

3.2 Pruebas de Unidad

Las pruebas de unidad (Figura 19) por lo general se consideran como adjuntas al paso de codificación. El diseño de las pruebas de unidad puede ocurrir antes de comenzar la codificación o después de generar el código fuente. La revisión de la información del diseño proporciona una guía para establecer casos de prueba que es probable que descubran errores en cada una de las categorías analizadas anteriormente. Cada caso de prueba debe acoplarse con un conjunto de resultados esperados (Pressman 2010).

Las pruebas de unidad verifican el funcionamiento de forma aislada de elementos de software que son comprobables por separado. Normalmente se prueban los métodos de las clases eliminando las dependencias que existan con otras estructuras lógicas. Típicamente, las pruebas de unidad se producen con acceso al código a prueba (IEEE 2014) (Sommerville 2011).

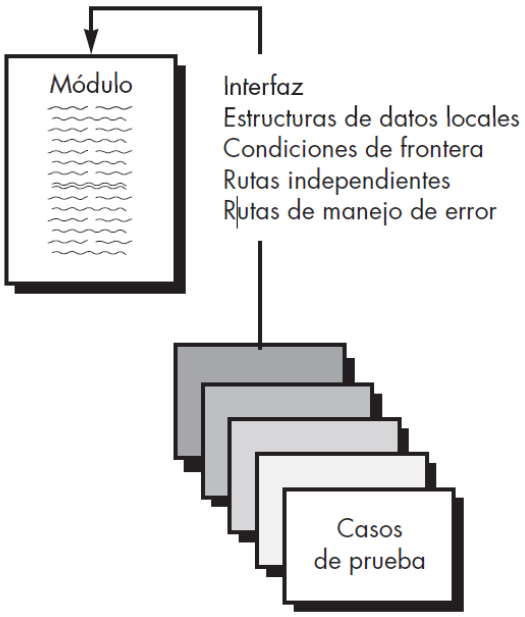

Figura 19 Pruebas de Unidad (Pressman 2010)

La automatización de estas pruebas es fundamental para agilizar el proceso, por tal motivo es utilizado el framework JUnit, integrado en la herramienta NetBeans. Este brinda al programador una serie de clases, interfaces y herramientas que puede emplear para implementar su patrón de prueba.

3.2.1 Casos de Prueba

A continuación, se detallan los casos de prueba para las funcionalidades descritas en el Capítulo 2, los que se muestran en forma de tablas (Tabla 2, Tabla 3) y recogen los resultados obtenidos después de aplicar aquellos métodos que ofrece JUnit para comprobar el correcto funcionamiento.

En el encabezado de las tablas se escribe el nombre del tipo de prueba que se diseña. Entre los primeros datos que recoge se encuentra el código del caso de prueba, la iteración en la que se realiza la prueba y una descripción de ésta, así como el nombre del encargado de la prueba. Como parte de la realización del caso de prueba se toma en cuenta en el primer campo (Funcionalidad), el cual corresponde al nombre de la funcionalidad comprobada, el segundo (Método utilizado) es el método de la clase *KORA_3IT.java* o de la clase *KORA_OmegaIT.java* utilizado, el tercer campo (Recibe) contiene los parámetros que recibe la funcionalidad al ser comprobada, el cuarto es el resultado que se espera que devuelva el método, en el quinto se escribe el resultado que se espera devuelva la prueba luego de ser ejecutada y la última columna es el valor resultado real que obtuvo finalmente dicha prueba. El resultado es satisfactorio si coinciden los resultados reales de las pruebas con los esperados, en caso de no coincidir es detectado un error.

Tabla 2 Caso de Prueba de Unidad 01

Caso de Prueba de Unidad				
Código del Caso de Prueba: 01				
Iteración: 1ra				
Descripción de la Prueba: genera las combinaciones posibles de 3 rasgos que conforman el sistema de conjuntos de apoyo.				
Nombre del encargado: Eliani Cabrera García				
Funcionalidad	Método utilizado	Recibe	Resultado Esperado del método	Resultado Real de la prueba
Generar Combinaciones	testGenerarCombinaciones()	-	Todas las combinaciones posibles de 3 rasgos.	Falló. Muestra resultados nulos

Tabla 3 Caso de Prueba de Unidad 02

Caso de Prueba de Unidad				
Código del Caso de Prueba: 02				
Iteración: 1ra				
Descripción de la Prueba: dada la lista de rasgos complejos y la lista de objetos con clases, a partir del parámetro η_i definido por el usuario, se calculan los objetos que pertenecen al resto de las clases y los almacena en una lista.				
Nombre del encargado: Eliani Cabrera García				
Funcionalidad	Método utilizado	Recibe	Resultado Esperado del método	Resultado Real de la prueba
Paso4	testPaso4()	• Lista de rasgos complejos. • Lista de objetos con clases.	Se calculan los objetos que pertenecen al resto de las clases y los almacena en una lista.	Falló. Incluye en la lista del resto de la clase objetos con más de η_i rasgos complejos.

3.2.2 Resultado de las Pruebas de Unidad

Se implementaron las pruebas de manera automática mediante los métodos de JUnit las cuales arrojaron en la primera iteración los resultados mostrados en la Figura 20.

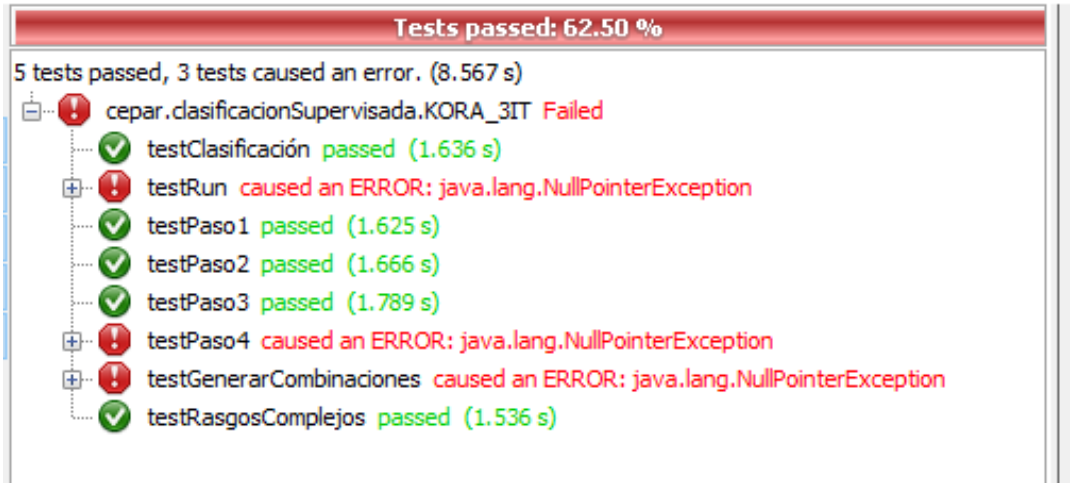

Figura 20 Resultados de la 1ra Iteración de las Pruebas de Unidad

- Esta prueba arrojó que el método run() poseía una excepción no tratada.
- Se comprobó que algunos métodos no validaban apropiadamente un conjunto de excepciones de cálculos matemáticos.

Se realizan tres iteraciones de las pruebas con la finalidad de solucionar las no conformidades encontradas en cada iteración anterior; los resultados obtenidos se muestra en la gráfica de la Figura 21.

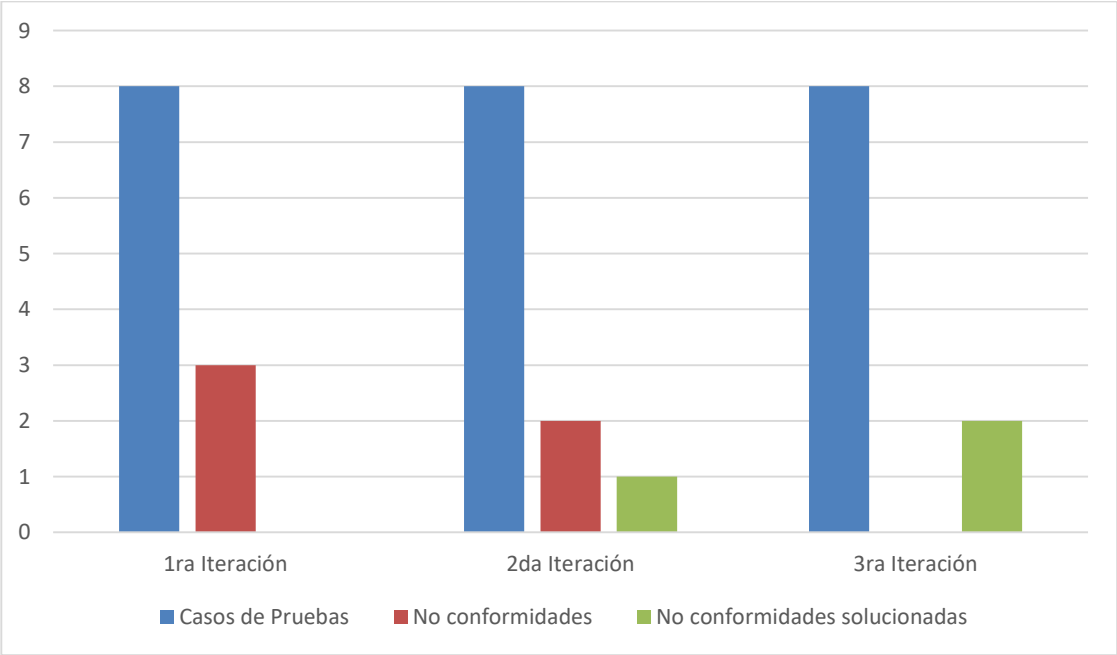

Figura 21 Representación gráfica de los resultados de las pruebas de unidad

En la última iteración como se muestra en la Figura 22 no fue encontrada alguna no conformidad en dependencia de los casos de pruebas aplicados.

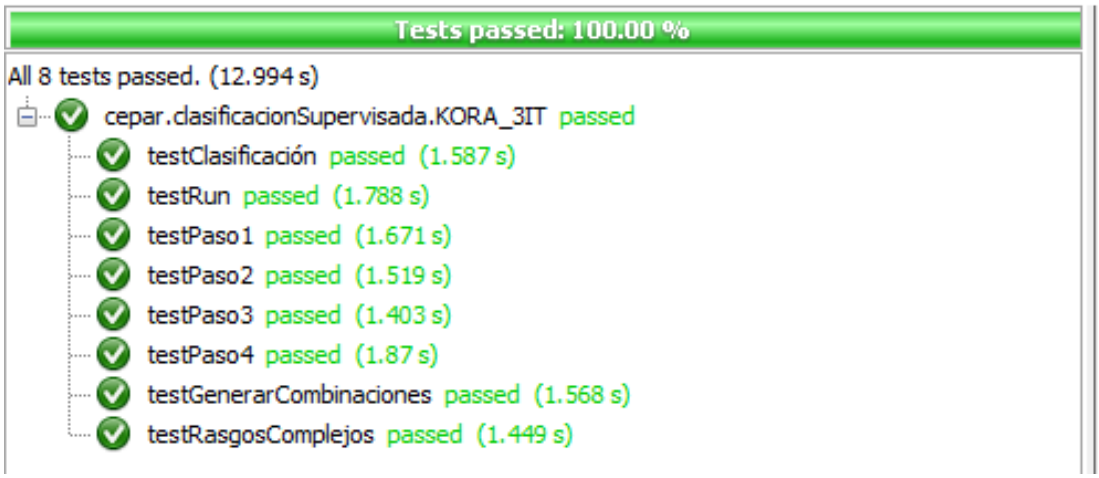

Figura 22 Resultados de la 3ra Iteración de las Pruebas de Unidad

Al finalizar esta iteración se evidencia que todos los problemas detectados fueron resueltos satisfactoriamente. Se comprobó que los métodos y clases funcionan de forma independiente, de una manera correcta y eficiente.

3.3 Pruebas de Integración

Las pruebas de integración son una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera individual y construir una estructura de programa que se haya dictado por diseño (Pressman 2010).

El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente; con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas y cubren las funcionalidades establecidas.

Utilizando los métodos de la clase IDataSet, el método RasgosComplejos() de la clase KORA-3, el método Clasificar() de la clase KORA-Omega y los casos de pruebas confeccionados de la Tabla 4 para las funcionalidades mencionadas se realizan las pruebas de integración de estos componentes comprobando que interactúan satisfactoriamente entre ellos.

Tabla 4 Casos de Prueba de Integración

No. Caso de Prueba de Integridad	Componente	Descripción de lo que se Probará	Prerrequisitos
Caso de Prueba 1	Rasgos Complejos	Se quiere probar la integración del modelo IDataSet existente, con las tareas programadas de la clase KORA_3.java. LA clase KORA_3.java, emplea la capa de Modelado de los Datos IDataSet para que la información sea procesada. Luego esta información llega al método RasgosComplejos().	Tiene que existir un conjunto de apoyo.
Caso de prueba 2	Clasificar	La tarea se empieza a ejecutar y solicita a la capa de Modelado de los Datos IDataSet la abstracción de los objetos sin clasificar y solicita toda la información a la clase KORA-Omega.java.	Tienen que existir objetos sin clasificar

Se constató que el funcionamiento de los métodos que relacionan al algoritmo implementado con el resto de la plataforma CEPAR retornaban el valor esperado, demostrando que la integración entre las clases del sistema y las de los algoritmos funcionan de forma monolítica y robusta.

Conclusiones parciales del capítulo

La validación de la implementación del algoritmo KORA-3 y su extensión KORA-Ω en la herramienta CEPAR se realizó a través de la aplicación de las pruebas unitarias al código desarrollado y las de integración. El desarrollo de los casos de prueba permitió constatar que el funcionamiento de los métodos para interconectar el algoritmo con el resto de la plataforma CEPAR retornaban el valor esperado, que los métodos y clases funcionan de forma independiente de una manera correcta.

CONLUSIONES GENERALES

1. El análisis de los conceptos fundamentales del reconocimiento de patrones, del enfoque Lógico Combinatorio, de los problemas de clasificación supervisada en la herramienta CEPAR y del algoritmo KORA contribuyeron a la fundamentación del marco teórico-conceptual de la investigación.
2. Se ofrece como solución de los problemas de clasificación supervisada en CEPAR el desarrollo del algoritmo KORA-3 y su extensión KORA-Ω.
3. La implementación e integración del algoritmo KORA-3 y de su extensión KORA-Ω permite hacer más extensible la herramienta CEPAR en la solución de problemas de clasificación supervisada.
4. La aplicación de las pruebas unitarias y de integración permitieron validar la implementación del algoritmo KORA-3 y su extensión KORA-Ω en la herramienta CEPAR, .

RECOMENDACIONES

1. Se recomienda la incorporación de otros algoritmos para la solución de problemas tanto de clasificación supervisada como de clasificación semisupervisada en la herramienta CEPAR, resaltando entre ellos los que utilicen las precedencias parciales.
2. Incluir funcionalidades al algoritmo KORA-Ω que permitan que para cada clase sea posible definir el conjunto de apoyo diferente.
3. Incluir en la herramienta CEPAR métricas para evaluar la eficacia de los algoritmos que posee.

REFERENCIAS BIBLIOGRÁFICAS

1. Arenas, A. G. and J. F. M. Trinidad (2001). "The logical combinatorial approach to pattern recognition, an overview through selected works." Pattern Recognition: 10.
2. Autores, C. d. (2012). "Patrones de Asignación de Responsabilidades (GRASP)."
3. Borroto, M. G. (2010). "Searching extended emerging patterns for supervised classification."
4. Borroto, M. G., Y. Villuendas, M. Á. M. P. Rey, J. M. López and J. R. Shulcloper (2013). "Reconocimiento de Patrones."
5. Coello, S. and R. A. H. León (2002). El Paradigma Cuantitativo de la Investigación Científica.
6. Doria, L. D. V. (1994). Extensión al caso Difuso del Algoritmo de Clasificación KORA-3. Maestro en Ciencias, Centro de Investigación y Estudios Avanzados.
7. Eduardo, N. C. S. (2015). Herramienta computacional para la caracterización de los estudiantes en las universidades públicas de Angola a partir del rendimiento académico Máster en Informática Aplicada, Universidad de las Ciencias Informáticas.
8. Franco, U. E. and G. S. Díaz (2009). "Algoritmo de votación incremental INC-ALVOT para clasificación supervisada." Revista de la Facultad de Ingeniería de la Universidad Antioquia **50**: 9.
9. Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition.
10. García-Borroto, M., J. F. Martínez-Trinidad and J. A. Carrasco-Ochoa (2014). "A survey of emerging patterns for supervised classification." Artificial Intelligence Review **42**(4): 705-721.
11. Godoy-Calderón, S., H. Calvo, V. M. Martínez-Hernández and M. A. Moreno-Armendáriz (2010). "The CR-Ω+ classification algorithm for spatio-temporal prediction of criminal activity." Journal of applied research and technology **8**(1): 5-23.
12. González, G. M. (2015). "Herramienta de Desarrollo Netbeans."
13. González, Y. R. (2014). Modelo para la adaptación de las soluciones en un Sistema Basado en Casos utilizando el agrupamiento conceptual., Universidad de las Ciencias Informáticas.
14. Guzmán, J. E. (2009). "Extensión de un lenguaje formal (LCARS) para especificar problemas de clasificación y de seleccion de rasgos." Centro de Investigación en Computación.
15. Hernández, E. P. and P. G. Figarola (2016). Extensión de la herramienta CEPAR con algoritmos de escala exterior para el cálculo de Testores Típicos. Ingeniero en Ciencias Informáticas Tesis de Grado, Universidad de las Ciencias Informáticas.
16. Hernández, V. M. M. (2009). Caracterización y predicción espacio-temporal de patrones delictivos mediante modelos lógico-combinatorios. MAESTRO EN CIENCIAS DE LA COMPUTACIÓN, Instituto Politécnico Nacional de México.

17. IEEE (2014). "Guide to the Software Engineering."
18. Jain, A. K., R. P. W. Duin and J. Mao (2000). "Statistical Pattern Recognition: A Review." IEEE **Vol.22**: 4-37.
19. Jendrock, E. (2014). "Java Platform, Enyerprise Edition."
20. Larman, C. (2003). "UML y Patrones." Prentice Hall.
21. López, P. G., A. A. G. López and J. A. G. Lázaro (2011). "Herramientas CASE.¿Cómo incorporarlas con éxito en nuestra organización?".
22. Luna, E. C. H. and E. M. F. Riverón (2014). "Identificación del autor de un texto manuscrito."
23. Ochoa, J. A. C. (2015). "Clasificación Supervisada en el Enfoque Lógico Combinatorio."
24. Ochoa, J. A. C., L. D. V. Doria and J. R. Shulcloper (1998). "Un modelo de clasificación supervisada basado en conjuntos de representantes."
25. Ochoa, J. A. C. and Y. M. Martínez (2013). "Cálculo del Sistema de Conjuntos de Apoyo y de los Umbrales de Representación para CR+ utilizando Técnicas de Paralelismo."
26. Ochoa, J. A. C. and J. F. M. Trinidad (2002). Combining Evolution Techniques to Estimate Feature Weights and the Support Sets System for ALVOT. Reconocimiento de Patrones. Avances y Perspectivas. J. L. D. L. Santiago and C. Y. Márquez. México. **Vol.1**.
27. Ochoa, J. A. C. and J. F. M. Trinidad (2003). "Combining Evolutionary Techniques to Improve ALVOT Efficiency." WSEAS Transactions on Systems **Vol.2**(4).
28. Ochoa, J. A. C. and J. F. M. Trinidad (2011). "Reconocimiento de patrones." Komputer Sapiens **Vol.II**: 4.
29. Ortiz-Posadas, M., J. Martínez-Trinidad and J. Ruiz-Shulcloper (1996). "A new approach to differential diagnosis of diseases." International journal of bio-medical computing **40**(3): 179-185.
30. Oses, Y., Y. R. González and N. M. Sánchez (2016). "CEPAR: Un Sistema Herramienta de Apoyo al docente del Reconocimiento Lógico Combinatorio de Patrones." XI Peña Tecnológica.
31. Peña, R. C. P. (1995). Prognosis. Sistema Herramienta de reconocimiento de patrones, Instituto de Matemática, Cibernética y Computación Ministerio de Ciencia, Técnica y Medio Ambiente.
32. Pérez, M. A. M. and J. R. Shulcloper (2012). Algoritmos de Votación Para Clasificación Supervisad, Editorial Académica Española.
33. Pressman, R. S. (2010). "Ingeniería del Software.Un enfoque práctico."
34. Razo Gil, L. J. (2013). Sistema para el reconocimiento del alfabeto dactilologico.
35. Rey, Y. V., M. R. E. Pérez, M. A. M. Pérez, M. G. Borroto and J. R. Shulcloper (2008). "Métodos para acelerar la búsqueda de los k vecinos más cercanos utilizando distancias y disimilaridades." Serie Azul.
36. Rumbaugh, J., I. Jacobson and G. Boch (2000). "El lenguaje Unificado de Modelado."

37. Shulcloper, J. R. (2007). "Clasificación de datos mezclados e incompletos." Revista Cubana de Ciencias Informáticas **Vol.1**: 13.
38. Shulcloper, J. R. (2009). Reconocimiento Lógico Combinatorio de Patrones: Teoría y Aplicaciones TESIS EN OPCIÓN AL GRADO CIENTÍFICO DE DOCTOR EN CIENCIAS.
39. Shulcloper, J. R. (2013). "Acerca del surgimiento del Reconocimiento de Patrones en Cuba." Revista Cubana de Ciencias Informáticas **Vol. 7, No. 2**: 24.
40. Shulcloper, J. R., A. G. Arenas and J. F. M. Trinidad (1999). Enfoque Lógico Combinatorio al Reconocimiento de Patrones. México.
41. Shulcloper, J. R. and M. L. Cortés (1995). "Herramientas para la clasificación supervisada en analogías parciales." Revista Mexicana de Ingeniería Biomédica **Vol.XVI**: 18.
42. Shulcloper, J. R. and M. L. Cortés (1999). "Mathematical Algorithms for the Supervised Classification Based on Fuzzy Partial Precedence." Mathematical and Computer Modelling **29**: 8.
43. Shulcloper, J. R., E. A. Edurado and M. L. Cortés (1995). "Introducción al Reconocimiento de Patrones. Enfoque Lógico Combinatorio." Verde **No. 51**: 238.
44. Shulcloper, J. R., J. A. C. Ochoa and J. F. M. Trinidad (2013). "Reconocimiento de patrones: conceptos y metodología." Serie Azul: 30.
45. Shulcloper, J. R., J. A. C. Ochoa and J. F. M. Trinidad (2015). Clasificadores Supervisados basados en precedencias parciales. Serie Azul.
46. Shulcloper, J. R. and R. C. Pico-Peña (1992). "Un nuevo enfoque en la construcción de sistemas de reconocimiento: los sistemas herramientas." Revista Ciencias Matemáticas **Vol.13**.
47. Sommerville, I. (2011). Ingeniería de Software. México.
48. Torres, L. P. and E. R. C. González (2012). "Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información Visual Paradigm for UML plug-in for Model Driven Development of Information Management Systems." Serie Científica de la Universidad de las Ciencias Informáticas **Vol. 5**: p. 1-11.
49. Ung, L. G. (2016). Algoritmos de escala interior para el cálculo de los testores típicos como extensión de la herramienta CEPAR. Ingeniero en Ciencias Informáticas Tesis de Grado, Universidad de las Ciencias Informáticas.
50. Vázquez, L. E. Q. (2009). Reconocimiento de Patrones, el enfoque Lógico Combinatorio Para obtener el título de Ingeniero Matemático, Instituto Politécnico Nacional Escuela Superior de Física y Matemáticas.

BIBLIOGRAFÍA

1. Arenas, A. G. and J. F. M. Trinidad (2001). "The logical combinatorial approach to pattern recognition, an overview through selected works." Pattern Recognition: 10.
2. Autores, C. d. (2012). "Patrones de Asignación de Responsabilidades (GRASP)."
3. Borroto, M. G. (2010). "Searching extended emerging patterns for supervised classification."
4. Borroto, M. G., Y. Villuendas, M. Á. M. P. Rey, J. M. López and J. R. Shulcloper (2013). "Reconocimiento de Patrones."
5. Coello, S. and R. A. H. León (2002). El Paradigma Cuantitativo de la Investigación Científica.
6. Doria, L. D. V. (1994). Extensión al caso Difuso del Algoritmo de Clasificación KORA-3. Maestro en Ciencias, Centro de Investigación y Estudios Avanzados.
7. Eduardo, N. C. S. (2015). Herramienta computacional para la caracterización de los estudiantes en las universidades públicas de Angola a partir del rendimiento académico Máster en Informática Aplicada, Universidad de las Ciencias Informáticas.
8. Franco, U. E. and G. S. Díaz (2009). "Algoritmo de votación incremental INC-ALVOT para clasificación supervisada." Revista de la Facultad de Ingeniería de la Universidad Antioquia 50: 9.
9. Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition.
10. García-Borroto, M., J. F. Martínez-Trinidad and J. A. Carrasco-Ochoa (2014). "A survey of emerging patterns for supervised classification." Artificial Intelligence Review 42(4): 705-721.
11. Godoy-Calderón, S., H. Calvo, V. M. Martínez-Hernández and M. A. Moreno-Armendáriz (2010). "The CR-Ω+ classification algorithm for spatio-temporal prediction of criminal activity." Journal of applied research and technology 8(1): 5-23.
12. González, G. M. (2015). "Herramienta de Desarrollo Netbeans."
13. González, Y. R. (2014). Modelo para la adaptación de las soluciones en un Sistema Basado en Casos utilizando el agrupamiento conceptual., Universidad de las Ciencias Informáticas.
14. Guzmán, J. E. (2009). "Extensión de un lenguaje formal (LCARS) para especificar problemas de clasificación y de seleccion de rasgos." Centro de Investigación en Computación.
15. Hernández, E. P. and P. G. Figarola (2016). Extensión de la herramienta CEPAR con algoritmos de escala exterior para el cálculo de Testores Típicos. Ingeniero en Ciencias Informáticas Tesis de Grado, Universidad de las Ciencias Informáticas.
16. Hernández, V. M. M. (2009). Caracterización y predicción espacio-temporal de patrones delictivos mediante modelos lógico-combinatorios. MAESTRO EN CIENCIAS DE LA COMPUTACIÓN, Instituto Politécnico Nacional de México.

17. IEEE (2014). "Guide to the Software Engineering."
18. Jain, A. K., R. P. W. Duin and J. Mao (2000). "Statistical Pattern Recognition: A Review." IEEE Vol.22: 4-37.
19. Jendrock, E. (2014). "Java Platform, Enyerprise Edition."
20. Larman, C. (2003). "UML y Patrones." Prentice Hall.
21. López, P. G., A. A. G. López and J. A. G. Lázaro (2011). "Herramientas CASE.¿Cómo incorporarlas con éxito en nuestra organización?".
22. Luna, E. C. H. and E. M. F. Riverón (2014). "Identificación del autor de un texto manuscrito."
23. Ochoa, J. A. C. (2015). "Clasificación Supervisada en el Enfoque Lógico Combinatorio."
24. Ochoa, J. A. C., L. D. V. Doria and J. R. Shulcloper (1998). "Un modelo de clasificación supervisada basado en conjuntos de representantes."
25. Ochoa, J. A. C. and Y. M. Martínez (2013). "Cálculo del Sistema de Conjuntos de Apoyo y de los Umbrales de Representación para CR+ utilizando Técnicas de Paralelismo."
26. Ochoa, J. A. C. and J. F. M. Trinidad (2002). Combining Evolution Techniques to Estimate Feature Weights and the Support Sets System for ALVOT. Reconocimiento de Patrones. Avances y Perspectivas. J. L. D. L. Santiago and C. Y. Márquez. México. Vol.1.
27. Ochoa, J. A. C. and J. F. M. Trinidad (2003). "Combining Evolutionary Techniques to Improve ALVOT Efficiency." WSEAS Transactions on Systems Vol.2(4).
28. Ochoa, J. A. C. and J. F. M. Trinidad (2011). "Reconocimiento de patrones." Komputer Sapiens Vol.II: 4.
29. Ortiz-Posadas, M., J. Martínez-Trinidad and J. Ruiz-Shulcloper (1996). "A new approach to differential diagnosis of diseases." International journal of bio-medical computing 40(3): 179-185.
30. Oses, Y., Y. R. González and N. M. Sánchez (2016). "CEPAR: Un Sistema Herramienta de Apoyo al docente del Reconocimiento Lógico Combinatorio de Patrones." XI Peña Tecnológica.
31. Peña, R. C. P. (1995). Prognosis. Sistema Herramienta de reconocimiento de patrones, Instituto de Matemática, Cibernética y Computación Ministerio de Ciencia, Técnica y Medio Ambiente.
32. Pérez, M. A. M. and J. R. Shulcloper (2012). Algoritmos de Votación Para Clasificación Supervisad, Editorial Académica Española.
33. Pressman, R. S. (2010). "Ingeniería del Software.Un enfoque práctico."
34. Razo Gil, L. J. (2013). Sistema para el reconocimiento del alfabeto dactilologico.
35. Rey, Y. V., M. R. E. Pérez, M. A. M. Pérez, M. G. Borroto and J. R. Shulcloper (2008). "Métodos para acelerar la búsqueda de los k vecinos más cercanos utilizando distancias y disimilaridades." Serie Azul.
36. Ruiz-Shulcloper, J. (2000). "Logical Combinatorial Pattern Recognition." libro no editado.

37. Rumbaugh, J., I. Jacobson and G. Boch (2000). "El lenguaje Unificado de Modelado."
38. Shulcloper, J. R. (2007). "Clasificación de datos mezclados e incompletos." Revista Cubana de Ciencias Informáticas Vol.1: 13.
39. Shulcloper, J. R. (2009). Reconocimiento Lógico Combinatorio de Patrones: Teoría y Aplicaciones TESIS EN OPCIÓN AL GRADO CIENTÍFICO DE DOCTOR EN CIENCIAS.
40. Shulcloper, J. R. (2013). "Acerca del surgimiento del Reconocimiento de Patrones en Cuba." Revista Cubana de Ciencias Informáticas Vol. 7, No. 2: 24.
41. Shulcloper, J. R., A. G. Arenas and J. F. M. Trinidad (1999). Enfoque Lógico Combinatorio al Reconocimiento de Patrones. México.
42. Shulcloper, J. R. and M. L. Cortés (1995). "Herramientas para la clasificación supervisada en analogías parciales." Revista Mexicana de Ingeniería Biomédica Vol.XVI: 18.
43. Shulcloper, J. R. and M. L. Cortés (1999). "Mathematical Algorithms for the Supervised Classification Based on Fuzzy Partial Precedence." Mathematical and Computer Modelling 29: 8.
44. Shulcloper, J. R., E. A. Edurado and M. L. Cortés (1995). "Introducción al Reconocimiento de Patrones. Enfoque Lógico Combinatorio." Verde No. 51: 238.
45. Shulcloper, J. R., J. A. C. Ochoa and J. F. M. Trinidad (2013). "Reconocimiento de patrones: conceptos y metodología." Serie Azul: 30.
46. Shulcloper, J. R., J. A. C. Ochoa and J. F. M. Trinidad (2015). Clasificadores Supervisados basados en precedencias parciales. Serie Azul.
47. Shulcloper, J. R. and R. C. Pico-Peña (1992). "Un nuevo enfoque en la construcción de sistemas de reconocimiento: los sistemas herramientas." Revista Ciencias Matemáticas Vol.13.
48. Sommerville, I. (2011). Ingeniería de Software. México.
49. Torres, L. P. and E. R. C. González (2012). "Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información Visual Paradigm for UML plug-in for Model Driven Development of Information Management Systems." Serie Científica de la Universidad de las Ciencias Informáticas Vol. 5: p. 1-11.
50. Ung, L. G. (2016). Algoritmos de escala interior para el cálculo de los testores típicos como extensión de la herramienta CEPAR. Ingeniero en Ciencias Informáticas Tesis de Grado, Universidad de las Ciencias Informáticas.
51. Vázquez, L. E. Q. (2009). Reconocimiento de Patrones, el enfoque Lógico Combinatorio Para obtener el título de Ingeniero Matemático, Instituto Politécnico Nacional Escuela Superior de Física y Matemáticas.

ANEXOS

Anexo 1. Entrevista semiestructurada al especialista

Objetivos de la entrevista:

- Obtener información sobre el funcionamiento del algoritmo KORA-3 y su extensión el algoritmo KORA-Ω.
- Obtener información sobre cómo se calcula el resto de las clases.
- Obtener información sobre el funcionamiento de la etapa de reaprendizaje de los algoritmos.
- Identificar los parámetros υ_i, η_i, β_i, λ_i, δ_i, presentes en las descripciones de los algoritmos.

Entrevistado: Dr. Cs. José Ruíz Shulcloper. Profesor y metodólogo de la Vicerrectoría Primera de la Universidad de las Ciencias Informáticas.

Secuencia de preguntas:

- Después de realizada la etapa de aprendizaje del algoritmo KORA-3 ¿a qué se le denomina resto de la clase? ¿cómo se calcula?
- ¿Qué caracterizan los parámetros υ_i, η_i, presentes en la descripción del algoritmo KORA-3?
- ¿En el algoritmo KORA-Ω cómo se define el tamaño de los conjuntos de apoyo? ¿cómo se define el Ω? ¿este Ω pueden ser los Testores Típicos?
- ¿Qué caracterizan los parámetros β_i, λ_i, δ_i, presentes en la descripción del algoritmo KORA- Ω?