Implementación del principio MDL en un algoritmo evolutivo GEP de clasificación

Trabajo de Diploma presentado como requisito parcial para optar al título de:

Ingeniero en Ciencias Informáticas

Autor: Gedric Suárez González

Tutores:

MSc. Alain Guerrero Enamorado

MSc. Yeneit Delgado Kios

Universidad de las Ciencias Informáticas

Facultad 1, Centro CISED

Habana, Cuba

2016

Dedicatoria

A mi padre, por ser mi ídolo, mi amigo y guía, por enseñar a sus hijos a no rendirse ante la lucha por sus sueños, por ser la luz que nunca se apagará en mi camino.

A mi madre, por tanto derroche de amor, por ser el centro de mi universo, por enseñarme a cambiar lágrimas por sonrisas, tristezas por alegrías y problemas por bendiciones.

A mi hermano, por ser mi mejor amigo, por ser un regalo de la vida, para que nunca se deje guiar por el camino, y valla por donde no hay camino construyendo su sendero.

A mis abuelos en especial a mima Griselda, por todos sus cuidados y atenciones, por ser mi segunda madre.

Agradecimientos

La vida no se mide ni en horas, ni en días, ni en años; se mide en momentos. Hoy tengo la oportunidad de agradecerles a todos aquellos que formaron parte de la construcción de este momento tan especial:

A mi tutor Alain, padre de esta investigación y constante consejero docente e investigativo.

A mi tutora Yeneit, por todo su esfuerzo y paciencia.

A mi familia, por siempre confiar en mí, por su apoyo incondicional, especialmente a mi mamá, a mi padre, mi hermano y a mi abuela Griselda. Quiero que sepan que los amo y los admiro mucho y este logro es de ustedes también, nunca le podré estar lo suficientemente agradecido por todo lo que han hecho por mí, gracias por ser mi apoyo y mi vida.

A mis hermanos de la vida: Jorge Luis (El Zurdo) y especialmente a Lázaro Miguel (Lachy) por todas las malas noches que hemos pasado durante la realización de esta tarea tan importante, por ser más que mi amigo, mi hermano y por apoyarme en todo momento que hemos vivido juntos y también agradecerle a toda su familia. Decirte Lachy que en las malas cuenta conmigo porque en las buenas sé que te sobran amigos.

A todos mis amigos, a quiénes no menciono para no olvidar a nadie, a todos, especialmente a Rachel y Ángel. Rache gracias por tanto apoyo y paciencia conmigo, nunca pierdas tu esencia, eres un ser extraordinario, gracias por todo.

A mi novia por apoyarme durante todo este tiempo, por su amor y también agradecerle a toda su familia.

A mis profesores y compañeros de la universidad, principalmente a los profesores Alien, Yaili, Yadier, Yoel, Evelyn y Yusleydi.

A la Universidad de las Ciencias Informáticas, a quién siempre le estaré agradecido.

Declaración jurada de autoría

Declaro por este medio que yo Gedric Suárez González, con carné de identidad 92020531681, ser el único autor de este trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ___ días del mes de ___ del año 2016.

_____

Gedric Suárez González

_____ _____

MSc. Alain Guerrero Enamorado MSc.Yeneit Delgado Kios

Resumen

La implementación del principio Longitud de Descripción Mínima (MDL) en un algoritmo evolutivo Programación de Expresiones Genéticas (GEP) de clasificación puede ser utilizada para reducir el sobreajuste de las reglas obtenidas después de aplicado dicho algoritmo. Para esto se procede por grupos, conteniendo cada grupo todas las reglas que cubran una misma clase. Dentro de cada uno de esos grupos se selecciona un subgrupo de reglas basándose en el principio MDL. Este principio indica que entre diferentes teorías para explicar una determinada evidencia, se debe preferir la teoría más simple. La idea del principio es buscar un compromiso para cubrir bastantes ejemplos y capturar así las regularidades presentes en los datos sin llegar a encontrar reglas demasiado específicas.

En la presente investigación se implementan nuevas variantes del principio MDL en un algoritmo evolutivo GEP de clasificación para reducir el sobreajuste. Este tipo de sistema clasificador presentará importantes ventajas para su aplicación en entidades, empresas y en sistemas informático creados en la Universidad de las Ciencias Informáticas, pues permitirá minimizar los riesgos a la hora de decidirse estratégicamente, lo que traerá consigo un ahorro considerable de tiempo, recursos humanos y financieros a la organización.

Palabras claves: clasificación, principio Longitud de Descripción Mínima, Programación de Expresiones Genéticas, reglas, regularidades, sobreajuste.

Contenido

Lista de figuras

Lista de tablas

Introducción

El desarrollo tecnológico y empresarial alcanzado en los últimos años, ha contribuido a aumentar la capacidad de generar datos, almacenarlos y clasificarlos en pequeños grupos que describan sus características principales, basándose en la similitud o diferencia entre ellos. Sin embargo, almacenar estas enormes masas de información en grandes bases de datos para realizar solamente procesos de consulta tradicional resulta de poca utilidad, por lo que suele ser necesario un procesamiento más avanzado de los mismos, habitualmente buscando extraer conocimiento para la toma de decisiones.

Transformar los datos en información útil para la toma de decisiones es una tarea compleja. Con el propósito de extraer la mayor cantidad de información provechosa para las empresas o instituciones, surge una nueva generación de técnicas computacionales y sistemas informáticos, los cuales constituyen el centro del emergente campo de investigación denominado: Minería de Datos (MD).

La MD es una de las fases del proceso de extracción de conocimientos a partir de los datos. La mayoría de los autores (Giráldez, 2003; Lara, 2010; Olmo, 2013) la consideran como la etapa principal de todo el proceso. Esta incluye diferentes técnicas de aprendizaje automático, la estadística, los sistemas de toma de decisiones y otras áreas de la informática y de la gestión de información. Esto permite que las entidades o empresas minimicen los riesgos a la hora de decidirse estratégicamente, lo que trae consigo un ahorro considerable de tiempo, recursos humanos y financieros a la organización.

Entre la gran cantidad de técnicas de aprendizaje automático existentes, destacan las basadas en Algoritmos Evolutivos (AE) (Muñoz, y otros, 2008; Rivas, y otros, 2016). Estos algoritmos son usados para procesar problemas con un conjunto de soluciones muy grande donde resulta imposible aplicar una búsqueda exhaustiva en la práctica. El crecimiento de sus aplicaciones demuestra la efectividad y fortaleza de utilizarlos en la esfera empresarial.

Un nuevo paradigma computacional dentro del grupo de los AE son los algoritmos de Programación de Expresiones Genéticas (del inglés *Gene Expression Programming*, en adelante GEP). A grandes rasgos, un GEP es una técnica para el análisis de datos que combina las ventajas de sus predecesores: los Algoritmos Genéticos y la Programación Genética (Ferreira, 2006). De los Algoritmos Genéticos heredó los cromosomas lineales de tamaño fijo; y de la Programación Genética su representación a modo de árboles de variados tamaños y formas.

Los algoritmos GEP constituyen un componente fundamental en una amplia gama de aplicaciones, desde su uso en el campo de la medicina para diagnosticar enfermedades (Ferreira, 2006; Dai, y otros, 2009), economía (Ferreira, 2006; Jingfeng, y otros, 2009), ecología (Ghani, y otros, 2010), telecomunicaciones (Janeiro, y otros, 2012), biología (H. Z., y otros, 2006), geografía (Eldrandaly, 2009; Bellini Saibene, y otros, 2015; Mohammadpour, y otros, 2016), hasta en la programación automática (Chen, y otros, 2007). Además la Sociedad Cubana de Reconocimiento de Patrones y Minería de Datos ha fomentado el intercambio de información entre expertos nacionales y extranjeros, permitiendo que actualmente se desarrollen proyectos en diferentes centros investigativos de todo el país y se produzcan aplicaciones de alta competitividad que implementen este tipo de paradigma computacional.

En la Universidad de las Ciencias Informáticas, centro cubano de altos estudios, se promueve la competitividad internacional de la industria cubana del software. Es por ello que constituye una ventaja la utilización de los algoritmos GEP en sistemas informáticos que implementen la clasificación en su ejecución.

En la mayoría de estas aplicaciones los resultados tienen que ser lo más precisos posibles. No obstante, el método recursivo utilizado por GEP para construir los árboles de expresión conduce a que estos sean complejos, ajustándose demasiado en ocasiones a los datos que intenta representar. Esto trae como consecuencia que el modelo sea muy específico y perjudica su comportamiento global. A este problema se le conoce como sobreajuste.

El sobreajuste lo definió la Real Academia de Ingeniería como: “Efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado” (Real Academia de Ingeniería, 2012).

Por tanto, el sobreajuste puede dar lugar a una descripción irrealista de los datos. Esto evidencia la necesidad de reducirlo en función de obtener un modelo mucho más confiable a la hora de clasificar nueva información.

El principio de “Longitud de Descripción Mínima” o en inglés *Minimum Description Length* (MDL) introducido por Jorma Rissanen en 1978 (Rissanen, 1978), es un método que selecciona modelos en términos de ganancia de información (Grunwald, 2005). Este principio indica que entre diferentes teorías para explicar una determinada evidencia, se debe preferir la teoría más simple. Al reflexionar sobre este tema se entiende que dado un conjunto de datos, el mejor modelo es aquel que requiere la mínima cantidad de información para ser construido. Teniendo en cuenta que tanto modelo como datos pueden codificarse mediante bits, se trata de minimizar la longitud de la codificación. Esto evita las teorías que satisfacen los datos al extremo sobreajuste y penaliza las teorías demasiado extensas.

A partir de la situación descrita anteriormente se define el siguiente **problema científico** en términos de pregunta de investigación: ¿Cómo reducir el sobreajuste de un algoritmo evolutivo GEP de clasificación a través del principio MDL?

El **objeto de estudio** lo constituye por tanto, el proceso de análisis del principio MDL.

A raíz del objeto de estudio definido anteriormente se tiene como **campo de acción** el principio MDL en un algoritmo evolutivo GEP de clasificación.

Con la finalidad de darle una solución práctica al problema en cuestión, se traza como **objetivo general** implementar el principio MDL en un algoritmo evolutivo GEP de clasificación para reducir el sobreajuste.

En el marco de esta investigación se ha desglosado el objetivo general en los siguientes **objetivos específicos**:

1. Fundamentar la codificación de algoritmos evolutivos GEP y sus aplicaciones en clasificación.
2. Fundamentar las aplicaciones del principio MDL en diferentes algoritmos de clasificación.
3. Implementar el principio MDL de modo que permita reducir el sobreajuste en un algoritmo GEP.

4. Experimentar con juegos de datos para comprobar el correcto funcionamiento del principio implementado.
5. Utilizar el método de validación cruzada en GEP con el principio implementado para garantizar la independencia entre los datos de entrenamiento y prueba.
6. Aplicar test estadísticos al algoritmo GEP para realizar comparaciones múltiples con las variantes de MDL implementadas y sin MDL.

La **idea a defender** de la presente investigación plantea que la implementación del principio MDL permitirá reducir el sobreajuste de un algoritmo evolutivo GEP de clasificación.

Para darle cumplimiento a los objetivos trazados se han determinado las siguientes **tareas de investigación**:

- Revisión y análisis de la bibliografía existente relacionada con los algoritmos evolutivos GEP y su aplicación en clasificación.
- Revisión y análisis de la bibliografía existente relacionada con las aplicaciones del principio MDL en diferentes algoritmos.
- Implementación del principio MDL en el algoritmo evolutivo GEP de clasificación.
- Validación del comportamiento del principio implementado en el algoritmo evolutivo GEP de clasificación.
- Uso del método de validación cruzada para la confirmación de la independencia entre los datos de entrenamiento y prueba.
- Aplicación de test estadísticos para la realización de comparaciones múltiples entre los algoritmos implementados.

Para dar cumplimiento al objetivo propuesto se han combinado diferentes métodos teóricos y empíricos de la investigación científica, en la búsqueda y procesamiento de la información. Ellos son:

- Métodos teóricos: El método analítico-sintético al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta. El método histórico-lógico para el estudio de los trabajos anteriores que constituyen referentes

teórico-prácticos en el tratamiento del problema planteado. El método de modelado para el desarrollo de los algoritmos propuestos en el trabajo.

- Métodos empíricos: Método experimental para comprobar la utilidad y corrección de los resultados obtenidos.

El presente trabajo consta, además de la introducción, por tres capítulos, conclusiones, recomendaciones, bibliografía y glosario de términos. Los capítulos están estructurados como sigue:

Capítulo 1 Algoritmos Evolutivos y Programación de Expresiones Genéticas: En este capítulo se realiza una introducción al cómputo evolutivo. Se destacan sus principales algoritmos y se realiza un enfoque hacia los algoritmos de Programación de Expresiones Genéticas, entidad de estudio para la aplicación del principio de Longitud de Descripción Mínima.

Capítulo 2 El principio MDL para la reducción del sobreajuste en la tarea de clasificación: En este capítulo se abordan definiciones y conceptos importantes del principio de Longitud de Descripción Mínima para la reducción del sobreajuste. Se describen características, ventajas y desventajas de dicho principio que se utilizan para dar solución al problema expuesto en la presente investigación. Además se proponen métricas para la medición del sobreajuste.

Capítulo 3 Descripción de la implementación y resultados experimentales: En este capítulo se indica el algoritmo evolutivo GEP de clasificación al cual se le implementará la propuesta de solución. Se describe la relación de los estados del nuevo clasificador mediante un diagrama de estados. Se representa el proceso de calcular el principio MDL mediante un diagrama de procesos. Además se definen los experimentos a ejecutar y se realiza el análisis estadístico a los resultados obtenidos.

Capítulo 1 Algoritmos Evolutivos y Programación de Expresiones Genéticas

En este capítulo se exponen los principales conceptos que sirven de base para adentrarse en el tema de los Algoritmos Evolutivos, se caracterizan los Algoritmos Genéticos y la Programación Genética cimientos de la Programación de Expresiones Genéticas. A esta última se le realiza una descripción más profunda, haciendo hincapié en su aplicación en la tarea de clasificación.

1.1 Antecedentes históricos

En 1859 el mundo científico cambia radicalmente la forma de ver el universo a raíz de la obra escrita por Charles Darwin “El origen de las especies” (Darwin, 1859). Los criterios de Darwin plantean: “que las especies evolucionan acorde al medio, para adaptarse a éste” (Darwin, 1859). De esta forma la naturaleza deja de ser vista como una creación de Dios y pasa a ser un conjunto de individuos en constante competición y evolución para poder perpetuar su especie en el tiempo. Las especies que no se adapten al medio desaparecen y solo los más aptos, los que mejor se adapten sobreviven para inmortalizar sus habilidades.

Una de las formas de representar esta perspectiva de la evolución desde las ciencias informáticas es tratando este marco como un evidente problema de optimización, dónde las entidades que constituyen soluciones al problema se nombran individuos o cromosomas, y el conjunto de estos, población. Los individuos se modifican utilizando operadores genéticos, esencialmente la selección, el cruce y la mutación, arrojando este proceso individuos que pasan a ser posibles mejores soluciones del problema en cuestión. De esta manera se mejora el promedio de adaptación de toda la población.

La idea básica de aplicar los principios darwinianos para la automatización de solución de problemas complejos data de los años 40. Ya en 1948 Alan Turing propuso “La búsqueda

de la evolución genética". Pero no fue hasta 1962 donde los experimentos realizados por Bremermann (Bremermann, 1962) dan comienzo a la aplicación de este nuevo concepto. Luego aparecen diferentes implementaciones de esta idea. Fogel, Owens y Walsh inventan la Programación Evolutiva (Fogel, y otros, 1965; Fogel, y otros, 1966), Holland llama a su método Algoritmo Genético (Holland, 1973; Holland, 1975) y Rechenberg y Schwefel introducen las Estrategias Evolutivas (Rechenberg, 1973; Schwefel, 1977). A principios de los 90 surge otra corriente basada en la perspectiva de la evolución creada por Koza, la Programación Genética (Koza, 1992; Koza, 1994).

Todo este campo es acreditado actualmente como una de las ramas de la computación evolutiva (Back, 1996; Eiben, y otros, 2003), los Algoritmos Evolutivos (AE) (Eiben, y otros, 2003). Cuyo propósito es realizar una búsqueda estocástica, haciendo evolucionar un conjunto de estructuras y seleccionando de modo iterativo las más adecuadas (Rivas, y otros, 2016). La Programación Evolutiva, los Algoritmos Genéticos, las Estrategias Evolutivas y la Programación Genética constituyen paradigmas de AE.

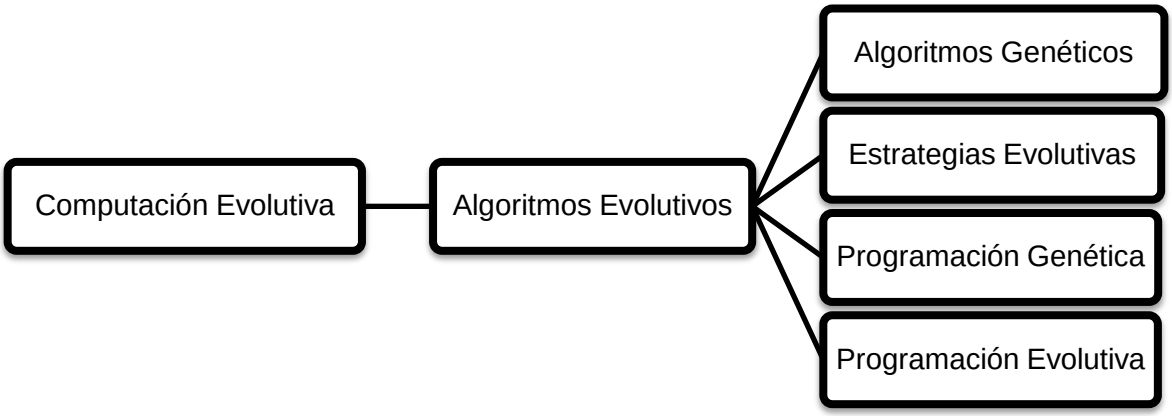

Figura 1: Taxonomía de la computación evolutiva (Muñoz, y otros, 2008).

1.2 Algoritmos Evolutivos

Algunos autores definen los Algoritmos Evolutivos como:

"…Los Algoritmos Evolutivos (AE) son metaheurísticas que emplean modelos computacionales del proceso evolutivo. Existen una gran variedad de AE, los principales incluyen: Algoritmos Genéticos, Programación Evolutiva, Estrategias Evolutivas y Programación Genética. Todos estos algoritmos comparten un concepto base común que

es simular la evolución de los individuos que forman la población usando un conjunto de operadores predefinidos…" (Villagra, y otros, 2006).

"…Los algoritmos evolutivos (*Evolutionary Algorithms*, EA): corresponden a un grupo de técnicas estocásticas que utilizan los conceptos de evolución biológica. Los EA actúan sobre una población de soluciones potenciales aplicando los principios de diversidad de individuos y supervivencia del más fuerte para producir mejores aproximaciones a una solución. En cada generación es creado un nuevo grupo de aproximaciones por el proceso de selección de individuos de acuerdo a su nivel de "desempeño" en el dominio del problema, y se cruzan entre sí utilizando operadores que imitan los conceptos genéticos. Este proceso lleva a la evolución de poblaciones de individuos que están mejor adaptados a su ambiente. Sus principales técnicas son los algoritmos genéticos (*Genetic Algorithms*, GA), las estrategias evolutivas (*Evolutions Strategies*, ES), la programación evolutiva *(Evolutionary Programming*, EP) y la programación genética (*Genetic Programming*, GP)…" (Muñoz, y otros, 2008).

"…Un algoritmo evolutivo puede definirse de manera simplista como un algoritmo que actúa sobre un conjunto de soluciones candidatas a través de dos actividades o principios: selección y variación. Mientras que la selección trata de imitar la competición por los recursos que existe en la naturaleza y por la reproducción de los seres vivos, la variación trata de imitar capacidad de creación de nuevas formas de vida a través de la recombinación y la mutación…" (Cárdenas, 2015).

Teniendo en cuenta la síntesis de estos conceptos se puede determinar la idea fundamental que persiguen los Algoritmos Evolutivos (AE): conservar un conjunto de individuos que constituyen una posible solución del problema de optimización a tratar, estos individuos se mezclan y compiten entre sí, persiguiendo el principio de selección natural por el cual solo los mejor adaptados sobreviven en el tiempo. Esto provoca una evolución hacia soluciones cada vez más idóneas.

Según José Alberto García Gutiérrez**:** "…Esta forma de funcionamiento les confiere su característica más destacable: un algoritmo evolutivo puede ser implementado de forma independiente del problema, o a lo sumo, con un conocimiento básico de éste, lo cual los hace algoritmos robustos, por ser útil para cualquier problema, pero a la vez débiles, pues no están especializados en ningún problema concreto siendo los operadores

genéticos empleados los que en gran parte confieren la *especificabilidad* al método empleado…" (García Gutiérrez, 1980).

1.2.1 Características de los Algoritmos Evolutivos

En correspondencia con las investigaciones de Cristina Mazuera y Yenifer Henao (Mazuera, y otros, 2013) se consideran como características principales de los AE las siguientes:

- Realizan una búsqueda global al trabajar con un conjunto de posibles soluciones llamado población de individuos, evitando de esta forma quedarse atascado en óptimos locales.
- Utilizan un método de selección sesgado con base al ajuste del individuo, indicando que tan bueno es el individuo, para identificar la probabilidad de que el individuo sea seleccionado y herede a otro su material genético.
- Generan nuevos individuos por medio de un mecanismo de herencia, a través del uso de operadores estocásticos como el cruce y la mutación.
- Aumentan la probabilidad de encontrar un óptimo global, al trabajar con poblaciones de individuos, que serán posibles soluciones.
- La utilización de una población de soluciones favorece las implementaciones paralelas.
- Manejan variables discretas.
- Son de naturaleza robusta, pueden aplicarse a muchos y muy variados problemas.

1.2.2 Funcionamiento de los Algoritmos Evolutivos

Tal como hace mención (Muñoz, y otros, 2008) existen diferentes técnicas que se agrupan bajo la definición de AE. Sin embargo, a pesar de sus diferentes implementaciones, se puede definir un algoritmo evolutivo genérico al que todas estas técnicas se ajustan en gran medida.

Procedimiento AE:

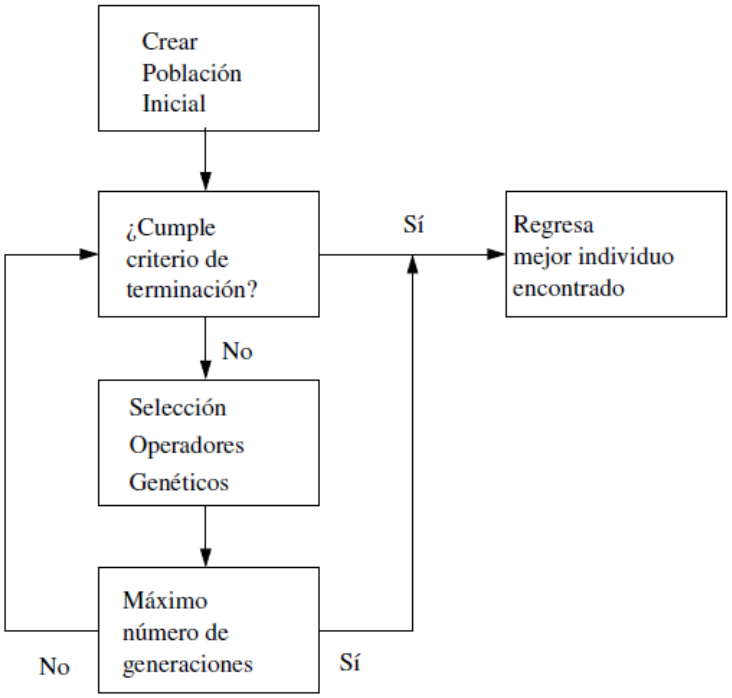

Figura 2: Diagrama de flujo de un Algoritmo Evolutivo genérico (Graff, 2010).

```
Inicializar (P (0))
generación = 0
Evaluar (P (0))
Mientras (no Criterio_Parada) hacer
        Padres = Selección (P (generación))
        Hijos = Operadores de Reproducción (Padres)
        Nueva_Población = Reemplazar (Hijos, P (generación))
        generación ++
        P (generación) = Nueva_Población
Retornar Mejor Solución Hallada
```

Figura 3: Seudo código de un Algoritmo Evolutivo genérico (Por elaboración propia).

A continuación, se describen cada uno de los pasos de este algoritmo:

Se comienza con la creación de una población inicial de individuos. Esta población se inicializa de manera aleatoria, pero puede ser inicializada basándose en conocimiento específico del dominio del problema, para tratar de dirigir la búsqueda hacia lugares más prometedores del espacio de investigación.

La evaluación de cada candidato consiste en asignarles a los mismos un valor de ajuste, que indica el grado de ajuste de este individuo con respecto al objetivo del problema en cuestión. Para obtener el valor de ajuste de cada candidato se le aplica la función de calidad o función objetivo.

El proceso evolutivo es iterativo por lo que se encuentra dentro de un bucle, que está controlado por una condición de parada que puede basarse en distintos tipos de criterios:

- El grado de evolución del mejor individuo de la población.
- El grado de diversidad entre los individuos. Si todos son similares es momento de detener el proceso evolutivo.
- Un contador, fijando de antemano un número máximo de generaciones o de evaluaciones de la función de ajuste.
- La combinación de las anteriores condiciones de parada.

La selección de padres reside en elegir individuos de la población actual para que tengan descendencia. Para esto existen numerosos métodos de selección, todos ellos basados en el ajuste de individuos.

Luego se prosigue a aplicar operadores evolutivos a los individuos seleccionados. Los operadores más habituales son la mutación y el cruce. Estos operadores son utilizados con una determinada probabilidad (elección estocástica).

Por último, se actualiza la población actual con los individuos que pasarán a ser la nueva generación.

Los AE son muy sensibles al modo en que se codifiquen los individuos de la población y la codificación utilizada puede influir sensiblemente en las posibilidades de convergencia a la solución del problema. Por tanto, la primera tarea más importante en un AE es encontrar la representación apropiada para las soluciones. La segunda tarea crítica será elegir la función de aptitud (*fitness*) que regirá la búsqueda.

1.2.3 Componentes de los Algoritmos Evolutivos

Al referirse a este aspecto (Eiben, y otros, 2003) esclarecen que los componentes fundamentales de los AE son:

- Representación (definición de individuos).

- Función de evaluación (o función *fitness*).
- Población.
- Mecanismo de selección de los padres.
- Operadores de variación, recombinación o mutación.
- Mecanismo de selección de sobrevivientes.

Representación: La construcción de los individuos es lo primero que se realiza en los AE. Es inevitable pasar del contexto original del problema a su espacio de soluciones.

Los objetos que constituyen soluciones posibles al problema son llamados fenotipos (o fenomas). La codificación de estos objetos son los individuos con los cuales trabaja el Algoritmo Evolutivo y se denominan genotipos (o genoma). La Figura 4 muestra un ejemplo de lo anterior.

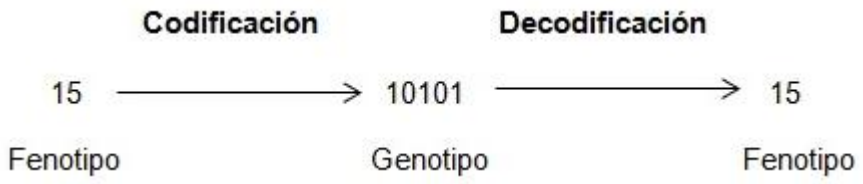

Figura 4: Codificación y Decodificación de soluciones (Por elaboración propia).

La codificación de las soluciones se clasifica en cuatro grupos atendiendo a los símbolos utilizados para la escritura del gen:

Codificación binaria: se utilizan ceros y unos en cada posición del cromosoma.

Codificación entera: es conocida como codificación de permutación literal y se utiliza en problemas de optimización combinatoria.

Codificación con números reales: es aplicada en problemas de optimización de funciones y en problemas de optimización con restricciones.

Codificación con estructuras de datos generales: es empleada en problemas complejos de la vida real. Cada gen del cromosoma es representado con una estructura de datos general.

Función de evaluación o función de aptitud: La función de aptitud (*fitness*) es equivalente a la función objetivo del problema de optimización a resolver. Una característica que debe tener esta función es que debe ser capaz de "castigar" a las

malas soluciones, y de "premiar" a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez (Coello Coello, 2011). Su objetivo es representar los requerimientos para la adaptación de los individuos al medio ambiente.

Población: La población es el conjunto de todas las soluciones (genotipo o fenotipo) en el medio ambiente y es el elemento que evoluciona en los AE.

Mecanismo de selección de los padres: El operador de selección tiene como función principal enfocar la búsqueda hacia las áreas más prometedoras de espacio de soluciones a explorar.

El Dr. Carlos A. Coello (Coello, 2015) confirma que los métodos de selección más usuales son:

Selección proporcional: Consiste en dar a cada individuo una probabilidad de ser seleccionado proporcional a su *fitness*.

Selección por torneo: La idea principal de este método consiste en realizar la selección en base a comparaciones directas entre individuos. Existen dos versiones de selección mediante torneo según esta elección se realice de forma determinista o probabilística.

Selección de estado uniforme: Según este criterio se reemplazan en cada generación los individuos menos aptos (suele emplearse en sistemas clasificadores).

Operadores de variación: Tienen como objetivo implantar una nueva descendencia tomando una descendencia anterior y de esta manera generar nuevas posibles soluciones al problema.

Cruce o recombinación:

El cruzamiento permite representar la combinación genética entre los individuos. Este puede realizarse sobre duplos de individuos o sobre toda la población. Su idea principal se basa en tomar dos individuos correctamente adaptados al medio y obtener una descendencia que comparta genes de ambos, existiendo la posibilidad de que los genes heredados sean los causantes de la bondad de los padres.

Martín Pedemonte (Pedemonte, 2003) y Carlos A. Coello (Coello, 2015) consideran que los métodos de cruce más empleados son los siguientes:

Cruce de un punto: Una vez seleccionados dos individuos se cortan sus cromosomas por un punto seleccionado aleatoriamente para generar dos segmentos diferenciados en

cada uno de ellos: la cabeza y la cola. Se intercambian las colas entre los dos individuos para generar los nuevos descendientes. De esta manera ambos descendientes heredan información genética de los padres.

Cruce de n puntos: Se realizan n cortes. Deberá tenerse en cuenta que ninguno de estos puntos de corte coincida con el extremo de los cromosomas para garantizar que se originen n+1 segmentos. Para generar la descendencia se escogen probabilísticamente los segmentos de los padres.

Mutación:

La mutación es una pequeña variación impuesta a un individuo, este proceso tiene como objetivos: preservar la diversidad genética evitando la convergencia prematura y explorar áreas posiblemente no abordadas del espacio de búsqueda.

Entre los operadores de mutación más utilizados según Estevez Valencia (Estevez Valencia, 2000) se encuentran:

Mutación simple: Este es el más elemental de los operadores de mutación. Es aplicable a codificaciones del tipo binaria y simplemente cambia el valor de un gen por su opuesto. Posee dos niveles de aleatoriedad, primero para decidir si el individuo en cuestión mutará, y segundo para seleccionar el gen específico que cambiará de valor.

Mutación de intercambio: Consiste en seleccionar al azar dos posiciones del cromosoma e intercambiar sus valores. El proceso de selección de las posiciones se hace usando el método de la ruleta para garantizar igualdad de probabilidad para todas.

Mutación de inversión: Es una generalización de la Mutación de intercambio. Se seleccionan dos posiciones al azar y se invierte el recorrido entre ellas.

Mecanismo de selección de sobreviviente: El rol del mecanismo de selección de sobrevivientes es distinguir los individuos en función de su calidad. Es similar a la selección de padres pero se utiliza en una etapa diferente del ciclo evolutivo (Eiben, y otros, 2003).

Otros componentes a tener en cuenta son la inicialización y la condición de término.

Las condiciones de término o parada del algoritmo se han expuesto anteriormente (Ver **1.2.2 Funcionamiento de los Algoritmos Evolutivos**).

Inicialización: es la primera población, que es generada por individuos aleatorios, pero puede ser inicializada basándose en conocimiento específico del dominio del problema, para ayudar al algoritmo en el proceso de búsqueda.

1.2.4 Algoritmos Evolutivos en la tarea de clasificación

Los AE son unas de las metaheurísticas más utilizadas en la actualidad. Entre sus usos más destacados, se encuentra la resolución de problemas basados en la extracción de conocimiento en el campo de Minería de Datos (*Data Mining*, MD) (Freitas, y otros, 2001).

Sobre este particular Witten y Frank definen la MD como: "…un área multidisciplinar, donde confluyen multitud de paradigmas de cómputo tales como: la construcción de árboles de decisión, la inducción de reglas, las redes neuronales artificiales, el aprendizaje basado en instancias, el aprendizaje bayesiano, la programación lógica y varios tipos de algoritmos estadísticos…" (Witten, y otros, 1999).

El objetivo de las técnicas de MD consiste en procesar y analizar la información para encontrar patrones repetitivos, tendencias, o reglas que expliquen el comportamiento de los datos en un determinado contexto (González, y otros, 2015).

Entre las tareas generales de la MD se encuentra la clasificación. Para la construcción de un modelo de clasificación usualmente se requiere de un algoritmo de aprendizaje y un conjunto de datos de entrenamiento (*training data set*). Un conjunto de datos de entrenamiento está compuesto por muestra de datos, donde cada muestra, $X = \{X_1, X_2,\ldots X_d\}$, es descrita por d atributos. También tiene un atributo clase $C= \{C_1, C_2,\ldots C_k\}$, en el cual k representa la cantidad de clases (Rosales, 2016).

Su finalidad es predecir la clase a la que pertenece una determinada instancia a partir de los valores de sus otros atributos, conocidos como atributos predictivos. Para ello, a partir de un conjunto de datos de entrenamiento se infiere un modelo o clasificador, que se utilizará para clasificar nuevas instancias o patrones no etiquetados en una de las categorías existentes (Olmo, 2013).

Esta tarea puede ser abordada mediante distintos métodos o técnicas (Villagra, y otros, 2006). Entre las técnicas más conocidas se encuentra el aprendizaje de reglas de clasificación.

Las principales ventajas del empleo de los AE en el aprendizaje de reglas de clasificación es su capacidad para ejecutar una búsqueda global y el tratamiento óptimo del problema en cuestión.

En el contexto de descubrimiento de reglas utilizando AE, un individuo corresponderá a una regla o conjunto de reglas candidatas. Existen dos maneras diferentes de representar un conjunto de reglas. En el enfoque “Michigan” (Holland, 1975; Booker, y otros, 1989), cada individuo de la población representa una regla de longitud fija y la población entera representa el objetivo. En el enfoque de “Pittsburgh” (Smith, 1980; Smith, 1983; De Jong, y otros, 1993), cada individuo de tamaño variable representa un conjunto entero de reglas.

La función objetivo corresponderá a alguna medida de la calidad de la regla o conjunto de reglas. El proceso de selección utilizará los valores del ajuste del candidato para elegir las mejores reglas o conjunto de reglas. La estrategia de búsqueda empleada por los AE es la principal diferencia con respecto a los algoritmos clásicos de inducción de reglas. Estos últimos suelen emplear una estrategia de búsqueda voraz local, mientras que los AE la emplean de manera global basándose en la selección natural.

1.2.5 Tipos de Algoritmos Evolutivos

Programación Evolutiva (PE): La Programación Evolutiva es introducida por Fogel (Fogel, y otros, 1965). En sus principios fue orientada como un proceso de aprendizaje para crear inteligencia artificial. Sus candidatos son comúnmente representados cómo máquinas de estado finito.

Estrategias de Evolución (EE): Las Estrategias Evolutivas fueron originadas en Alemania en los años 60 por Rechenberg (Rechenberg, 1973). Al comienzo estaban dirigidas a resolver problemas de optimización discretos y continuos. Sus individuos son simbolizados cómo vectores con valores reales.

Algoritmos Genéticos (AG): Los Algoritmos Genéticos fueron implantados por Holland (Holland, 1973). Estos se utilizan en la optimización de procesos no lineales (Michalewicz, 1996; Eiben, y otros, 2003). Sus candidatos son representados cómo cadenas de caracteres de un cierto alfabeto.

Programación Genética (PG): Hacen su aparición en los años 90. Originados por Koza (Koza, 1992), permite abordar problemas de optimización no lineal basada en un lenguaje simbólico. Sus individuos son representados en forma de árboles.

1.3 Algoritmos Genéticos

Los Algoritmos Genéticos (*Genetic Algorithms*, AG) fueron inicialmente concebidos por Holland como resultado de su estudio acerca del comportamiento adaptativo (Eiben, y otros, 1998) y modelado en su libro *"Adaptation in natural and artificial system"* (Holland, 1975). Su aplicación más común ha sido la solución de problemas de optimización, en donde han mostrado ser muy eficientes y confiables (Coello Coello, 2011).

Una definición a grandes rasgos de un AG es la propuesta por Jhon Koza: "…un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud…" (Koza, 1992).

En la formulación original concebida por Holland (Holland, 1975), un AG cuenta con una representación tradicional, llamada codificación binaria, una selección proporcional y la recombinación como su principal operador de evolución. Actualmente se han desarrollado varios cambios en esta formulación, incluyendo diferentes tipos de representaciones y de operadores: selección, recombinación, mutación y elitismo (Rosales, 2016).

1.3.1 Paralelismo implícito

La eficiencia de los AG se fundamenta en que, aunque sólo procesan n estructuras en cada generación, es demostrable que, bajo hipótesis muy generales, se procesan de manera ventajosa como mínimo n^3 soluciones. Este paralelismo implícito se logra sin ningún dispositivo o memoria adicionales, sólo con la propia población. Esto se debe a que los AG poseen descendencia múltiple, por tanto pueden explorar el espacio de

soluciones en múltiples direcciones a la vez. Si una ruta llega a ser un callejón sin salida, pueden desecharlo simplemente y continuar la ejecución con caminos más prometedores. Lo que implica una mayor probabilidad de encontrar la solución en cada iteración.

1.3.2 Ventajas

- Son intrínsecamente paralelos.
- Son eficaces en la resolución de problemas con espacios de soluciones potenciales grandes (problemas no lineales).
- A diferencia de las redes neuronales clásicas, es poco perceptivo a los mínimos locales, lo cual le otorga robustez.
- Poseen un buen desempeño a la hora de resolver problemas que tienen una función de calidad ruidosa, que varía en el tiempo, discontinua.
- No dependen de las condiciones iniciales, pues usan búsqueda estocástica y ésta hace al principio una gran cantidad de intentos aleatorios.
- Tienen gran habilidad para manipular simultáneamente muchos parámetros.
- Su tiempo de convergencia es predecible por la naturaleza paralela de la exploración estocástica.
- Al no conocer nada de los problemas que deben resolver, pueden viajar ampliamente por todo el espacio de soluciones, explorando regiones que algoritmos producidos podrían no haber tenido en consideración, e ir obteniendo soluciones potencialmente exitosas.

1.3.3 Desventajas

- Los algoritmos AG tendrán un tiempo de ejecución mayor que las técnicas tradicionales, si la población inicial es próxima a la solución óptima.
- La representación seleccionada para codificar las soluciones tiene que ser robusta. De forma tal que puedan sufrir cambios aleatorios sin caer en resultados sin sentido.

- Elegir cuidadosamente los parámetros de tamaño de población, mutación, cruce y tipo de selección. Si el tamaño de población es muy pequeño, el algoritmo probablemente no explorará adecuadamente el espacio de soluciones. La población puede entrar en catástrofe de errores, si los parámetros de mutación, cruce y selección son escogidos de manera inadecuada.
- El problema de definir una función objetivo que sea la más apropiada, para garantizar la convergencia a soluciones óptimas del problema.
- Durante la ejecución del algoritmo puede emerger la convergencia prematura. Si el algoritmo encuentra un candidato demasiado óptimo en poco tiempo, provoca que disminuya la diversidad de la población. Por tanto ocurre la convergencia hacia ese óptimo local que representa el candidato, en lugar de encontrar el óptimo global.

1.3.4 Aplicaciones

- Navegación robótica (Dos Santos, 2016).
- Calibración de imágenes (Molina, 2015).
- Minería de Datos (Blokdijk, 2015).
- Optimización de redes neuronales (Bikakis, y otros, 2015).
- Diseño de controlador difuso (Melin, y otros, 2015).
- Generación de datos de prueba para programas (Calero, y otros, 2015).

1.4 Programación Genética

La Programación Genética (*Genetic Programming*, PG) fue introducida por Koza (Koza, 1992), como una variación de los AG siguiendo el mismo principio Darwiniano de selección natural. Su característica sustancial se fundamenta en la representación de sus candidatos en forma de árboles, lo que permite su aplicación en una serie de problemas donde los algoritmos genéticos no pueden ser utilizados.

José J. Martínez Páez conceptualiza la PG como: "…un retoño de los Algoritmos Genéticos, en la cual los cromosomas que sufren la adaptación son en sí mismos

programas de computador. Se usan operadores genéticos especializados que generalizan la recombinación sexual y la mutación […] La PG trata de resolver uno de las cuestiones más excitantes e interesantes de las ciencias de la computación: ¿cómo pueden aprender los computadores a solucionar problemas sin que se les programe explícitamente? En otras palabras, la cuestión es cómo se puede hacer para que los computadores hagan lo que tienen que hacer, sin necesidad de la intervención humana que les diga exactamente como lo deben hacer…" (Martínez Páez, 2001). La PG usa un lenguaje de representación más complejo (normalmente un árbol) para la codificación de los individuos (Olmo, 2013).

1.4.1 Codificación de los individuos

Los individuos son de tamaño variable, el tamaño de los árboles varía a medida que sufren cruces y mutaciones. Para representar un individuo se deben de tener en cuenta los siguientes elementos:

Nodos terminales: En un individuo árbol, sus hojas representan los nodos terminales (variables y constantes).

Funciones: Los nodos internos, los que poseen uno o más hijos, son nodos no terminales y constituyen funciones u operaciones.

A continuación se aprecia un ejemplo de árbol, dónde se representa el programa f(x) = 2*(3+x). Que posee como funciones la suma y el producto, y como nodos terminales los valores de 2 y 3 y la variable x.

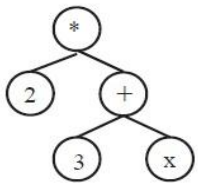

Figura 5: Representación del árbol correspondiente a la expresión 2*(3+x) (Por elaboración propia).

El conjunto de elementos terminales y funciones deben responder a dos características fundamentales, suficiencia y cerradura (Koza, 1992). El requisito de suficiencia dice que la solución al problema debe poder ser especificada con el conjunto de operadores especificados. El requisito de cerradura dice que cualquier árbol que se construya con estos operadores debe ser correcto (Gestal, y otros, 2010).

Par evitar la representación de códigos redundante en los árboles y su crecimiento excesivo, fenómeno llamado *bloat* (Soule, y otros, 1997). Se define desde un principio el tamaño máximo que puedan alcanzar los árboles, puede basarse en una cantidad permitida de nodos o en una profundidad máxima.

1.4.2 Ventajas

De acuerdo con Dania I. Ahumada (Ahumada, 2015) se consideran como ventajas principales de la PG, las siguientes:

- No necesitan conocimientos específicos sobre el problema que intentan resolver.
- Operan de forma simultánea con varias soluciones, en vez de trabajar de forma secuencial como las técnicas tradicionales.
- Cuando se utilizan para problemas de optimización, maximizar una función objetivo, resulta menos afectada por los máximos locales (falsas soluciones) que las técnicas tradicionales.
- Resultan muy fáciles de ejecutarse en las modernas arquitecturas masivamente paralelas.
- Utilizan operadores probabilísticos, en vez de los típicos operadores determinísticos de las otras técnicas.

1.4.3 Desventajas

- Los algoritmos pueden converger tempranamente debido a problemas de distintas cualidades.
- El proceso de evaluación de la función objetivo, puede requerir horas e incluso días en problemas de gran envergadura.
- Pueden demorar en converger, o no converger, dependiendo de los parámetros que se utilicen como la cantidad de generaciones y el tamaño de la población.
- Pueden llegar a caer en el fenómeno *bloat*, si los parámetros no son correctamente definidos en el inicio del algoritmo.
- Demandan grandes recursos de cómputos para problemas complejos.

1.4.4 Aplicaciones

- Procesamiento de imágenes y señales (Nandi, y otros, 2015).
- Finanzas (Resta, 2015).
- Control de procesos industriales (Gandomi, y otros, 2015).
- Medicina y Bioinformática (Ortuño, y otros, 2015).
- Híper-heurística (Gendreau, y otros, 2010).
- Entretenimientos y juegos para computadoras (Chorianopoulos, y otros, 2015).

1.5 Programación de Expresiones Genéticas

La Programación de Expresiones Genéticas (*Gene Expression Programming*, GEP) es, como los Algoritmos Genéticos y la Programación Genética, un Algoritmo Genético ya que utiliza la población de los individuos, los selecciona de acuerdo a la aptitud, e introduce la variación genética utilizando uno o más operadores genéticos. La diferencia fundamental entre los tres algoritmos reside en la naturaleza de los individuos: en AG los individuos son cadenas lineales de longitud fija (cromosomas); en PG los individuos son entidades no lineales de diferentes tamaños y formas (árboles); y en GEP los individuos se codifican como cadenas lineales de longitud fija (el genoma o cromosoma), que son posteriormente expresadas como entidades lineales de diferentes tamaños y formas (es decir, representaciones de diagramas simples o árboles de expresión) (Ferreira, 2006).

El algoritmo GEP, ideado por Ferreira (Ferreira, 2006), es una técnica basada en la filosofía de la computación evolutiva. Los algoritmos GEP se pueden considerar un híbrido entre los Algoritmos Genéticos y la Programación Genética (Langdon, y otros, 2002). GEP supera muchas limitaciones de los AG y la PG (Teodorescu, y otros, 2008).

1.5.1 Funcionamiento de la Programación de Expresiones Genéticas

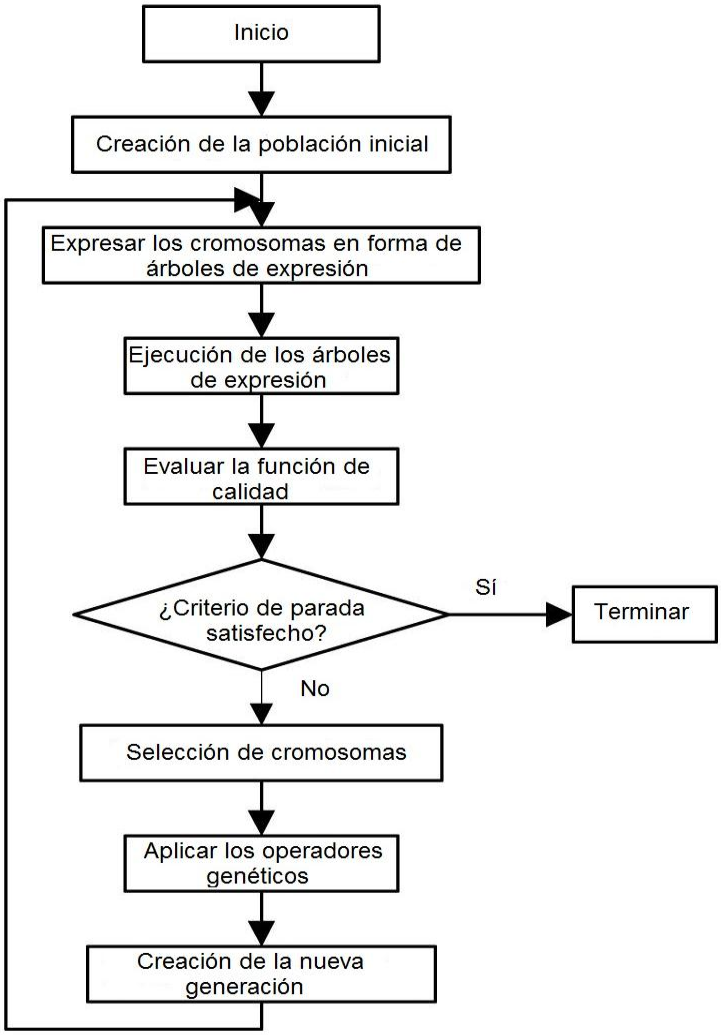

Figura 6: Diagrama de flujo de un típico algoritmo GEP (Teodorescu, y otros, 2008).

Al comienzo del algoritmo evolutivo GEP de clasificación se determinan los cinco componentes siguientes (Behnia, y otros, 2013):

1. Conjunto de funciones.
2. Conjunto de terminales.
3. Función de aptitud.
4. Parámetros de control.
5. Criterio de parada.

El flujo representado (Ver **Figura 6**) se inicia con la generación de los individuos de la población inicial, esto se realiza de forma aleatoria. Luego los cromosomas se expresan en forma de árboles de expresión (*Expression Trees,* ET) y se ejecutan los ET.

La Figura 7 muestra un ejemplo de la representación de un cromosoma en GEP:

Expresión: (a/b)*(c+d)

Genotipo

*	/	+	a	b	c	d

Fenotipo

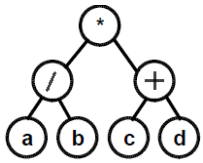

Figura 7: Ejemplo de representación de un cromosoma en GEP (Por elaboración propia).

Cada gen está formado por una cabeza y una cola. La cabeza contiene símbolos que representan funciones y terminales, mientras que la cola contiene solo terminales. La longitud de la cabeza (h) se elige dependiendo del problema. La longitud de la cola (t) depende de h y del número de argumentos (n) de la función que tenga más argumentos: t=h*(n-1)+1. Con esta regla se garantiza que las funciones no queden sin suficientes argumentos y que el programa no pueda crecer indefinidamente como sucede en GP.

Ferreira refiere además que un cromosoma puede estar compuesto por más de un gen (Ferreira, 2006):

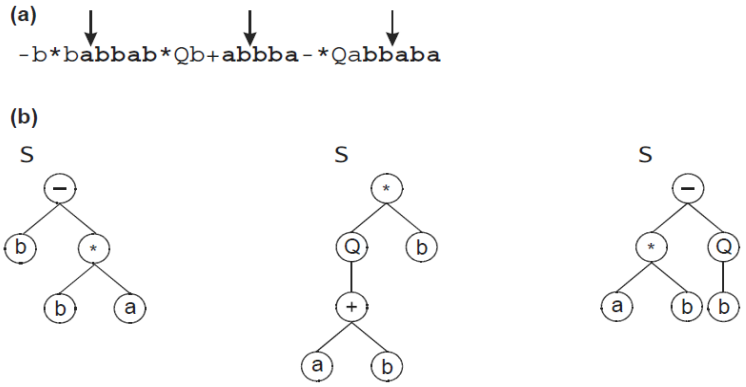

Figura 8: Expresión de genes GEP como sub-ET. (a) Un cromosoma de tres genes con las colas que se muestran en negrita. Las flechas muestran el punto de terminación de cada gen. (b) El sub-ET codificado por cada gen (Ferreira, 2006).

En cuarto lugar del flujo se evalúa la aptitud de cada individuo y seguidamente se verifica el criterio de terminación definido.

Si no se ha alcanzado el fin del ciclo entonces se prosigue a seleccionar los candidatos que formarán parte de la nueva generación y se modifican utilizando los operadores genéticos: cruce, mutación y rotación.

El cruce y la mutación se realizan de la misma manera que en PG. Se debe tener en cuenta que la mutación hay que hacerla con la restricción de que la organización estructural de los cromosomas debe ser preservado.

Rotación: Consiste en que dos partes de la secuencia del genoma de un elemento, se hacen girar con respecto a un punto elegido al azar. La rotación puede reformar drásticamente el árbol de expresión (Zhou, y otros, 2003).

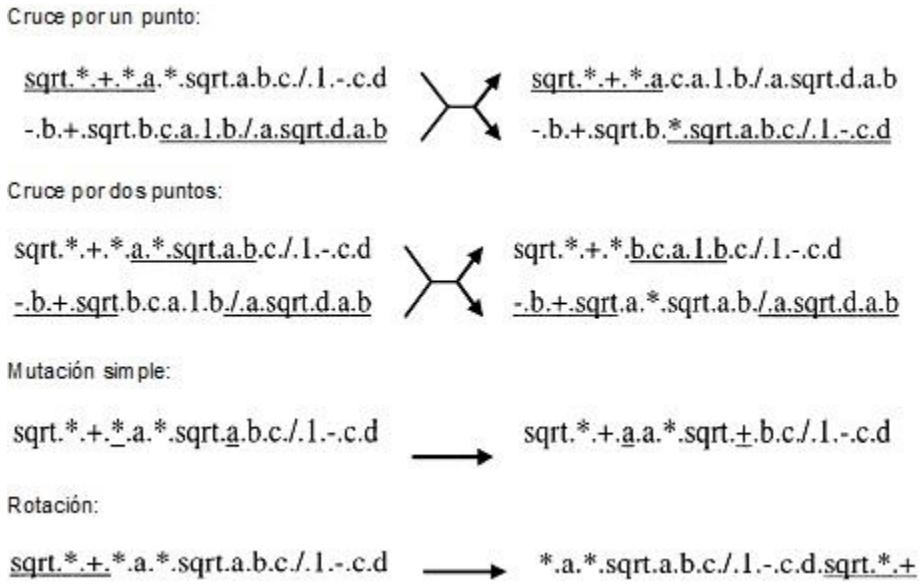

Figura 9: Operadores genéticos para los algoritmos GEP (Zhou, y otros, 2003).

Operadores de transposición (Ferreira, 2001):

Transposición *insertion sequence* (IS):

Se inicia mediante la selección al azar de una secuencia de elementos a transponer. Esta secuencia se inserta en una posición aleatoria en la cabeza del gen que está siendo modificado, excepto en el principio de la cabeza para evitar la generación de individuos con solamente un terminal (Ferreira, 2006).

Transposición *root of insertion sequence* (RIS):

Se inicia mediante la selección de una secuencia de elementos de transposición al azar, pero con la restricción de que comienza con una función. Un punto en la cabeza se elige

al azar para un gen preseleccionado y desde este punto en adelante, el gen se escanea hasta que se encuentra una función. Esta la función se convierte en la primera posición del elemento de RIS. Si no se encuentra ninguna función, el operador no realiza cambios (Ferreira, 2006).

Por último, los individuos de esta nueva generación se someten al mismo flujo del algoritmo.

1.5.2 Programación de Expresiones Genéticas en la tarea de clasificación

El objetivo de la clasificación de datos es predecir el valor de un atributo objetivo, dado un conjunto de atributos predictivos. Por lo general, las reglas están representadas en la forma SI <antecedente> ENTONCES <consecuente>, donde "antecedente" es una combinación lógica de los atributos que predicen y sus valores, y la parte "consecuente" es el valor del atributo objetivo (clase) (Weinert, y otros, 2006).

La representación de un árbol de decisión como un conjunto de reglas es trivial: desde cada camino de la raíz hasta una hoja se obtiene una regla, cuyo antecedente es una conjunción de condiciones referentes a los valores de los atributos ubicados en los nodos internos del árbol y cuyo consecuente es la decisión a la que hace referencia la hoja del árbol, la clasificación efectuada.

El problema de definir funciones de aptitud para medir la calidad de las reglas sigue siendo un tema interesante en la Minería de Datos (Freitas, 2001). En los problemas de clasificación no basta con adivinar bien o mal. Existen diferentes tipos de decisiones equivocadas y diferentes tipos de decisiones correctas, y todos ellos tienen diferentes costos y beneficios. Y todos estos diferentes costos y beneficios pueden ser explorados diseñando una correcta función de aptitud para evolucionar el tipo de modelo (Ferreira, 2006).

Existen cuatro diferentes resultados posibles de una sola predicción para un problema con clases de “1” (positivo) y “0” (negativo):

- Falso positivo (*False Positive*, Fp): cuando el resultado se clasifica incorrectamente como positivo, cuando en realidad es negativo.

- Falso negativo (*False Negative*, Fn): cuando el resultado se clasifica incorrectamente como negativo, cuando en realidad es positivo.
- Verdadero positivo (*True Positive*, Tp): cuando el resultado se clasifica correctamente como positivo.
- Verdadero negativo (*True Negative*, Tn): cuando el resultado se clasifica correctamente como negativo.

Para el cálculo de estos posibles resultados se utiliza la siguiente matriz de confusión:

		Clase Predicha	
		Positivo	Negativo
Clase Real	Positivo	Verdaderos Positivos (T_P)	Falsos Negativos (F_N)
	Negativo	Falsos Positivos (F_P)	Verdaderos Negativos (T_N)

Figura 10: Matriz de confusión (Olmo, 2013).

Buenos resultados corresponden a un gran número abajo de la diagonal principal (Tp+Tn) y para los números pequeños (idealmente cero) de la diagonal (Fp+Fn). Estos dos tipos de errores, falsos positivos y falsos negativos, por lo general tienen diferentes costos. Y obviamente, los dos tipos de clasificaciones correctas, verdaderos positivos y verdaderos negativos, tendrán distintos beneficios (Ferreira, 2006). En cada problema se pueden utilizar estos costos y beneficios para definir una función de aptitud que se adapte al modelo.

Autores como Zhou, Xiao, Tirpak, Nelson (Zhou, y otros, 2003) han expuesto las razones por las cuales utilizar GEP en la tarea de clasificación:

Flexibilidad: GEP es más flexible que las reglas de inducción tradicional y que los algoritmos de árboles de decisión, ya que con más flexibilidad reformula las representaciones subyacentes.

Capacidad: Debido al empleo de poderosos mecanismos evolutivos de búsqueda, se pueden descubrir relaciones en forma de una combinación de atributos y expresarlos matemáticamente.

Eficiencia: Con la ayuda de la representación lineal de los cromosomas con longitud fija y la fácil manipulación genética de operaciones, GEP es más eficiente que la tradicional PG.

1.5.3 Herramientas que utilizan Programación de Expresiones Genéticas en la tarea de clasificación

En la actualidad existen herramientas que utilizan la técnica GEP para hacer frente de manera eficiente a la clasificación de datos. GEPCLASS (Gene-Expression Programming for CLASSification) introducida por Weinert y Lopes (Weinert, y otros, 2006), es una herramienta que ha sido creada para este propósito.

El algoritmo GC-ML (GEPCLASS Multi-label) (Ávila Jiménez, y otros, 2010), es un algoritmo de PG que se basa en GEP para resolver problemas de clasificación multi-etiquetas. Estos problemas tienen como objetivo final, determinar qué patrón de individuos pertenecen a una determinada etiqueta en un contexto multi-etiquetas.

Software propietario GeneXproTools, comercializado por la empresa GepSoft fundada por Candida Ferreira. Es una herramienta de modelado que implementa GEP para la regresión, regresión logística, clasificación, tiempo de predicción de series, y la lógica de síntesis.

El algoritmo MCGEP (*Multi-class with gene expression programming algorithm*) (Guerrero Enamorado, y otros, 2015), es un algoritmo de clasificación con un enfoque Michigan que construye una lista de decisión conformada por los mejores individuos obtenidos por cada clase. Resuelve el problema multi-clase como n problemas binarios en un enfoque uno contra todos. Incluye una competencia por ganar instancias dentro del algoritmo para aumentar la diversidad y eliminar redundancia entre los individuos. Solo clasifican como reglas potenciales para formar el clasificador las que estén por encima de un % cobertura.

1.5.4 Ventajas

- Los cromosomas son entidades simples: lineal, compactos, relativamente pequeño, fácil de manipular genéticamente (replicar, mutar, recombinar, transposición, etc.) (Ferreira, 2006).

- Los árboles de expresión son exclusivamente la representación de sus respectivos cromosomas (Ferreira, 2006).
- Aprovecha el poder de la búsqueda evolutiva para detectar relaciones entre los datos y formularlas como expresiones matemáticas (Zhou, y otros, 2003).
- Provee de una capacidad de búsqueda global que tiende a generar soluciones más cortas que los clasificadores canónicos basados en GP (Zhou, y otros, 2003).
- Extrae conocimiento relevante a partir de colecciones de datos enormes (Ferreira, 2006).
- Es un algoritmo eficiente, por la fácil manipulación genética de sus individuos y la representación lineal de los cromosomas con longitud fija (Zhou, y otros, 2003).
- Es un algoritmo flexible, por la manera en que reformula las representaciones subyacentes (Zhou, y otros, 2003).
- Son capaces de resolver problemas relativamente complejos usando pequeños tamaños de población (Ferreira, 2004).

1.5.5 Desventajas

El método recursivo de construcción de árboles de decisión, conduce a la generación de árboles muy complejos ajustados a los datos del conjunto utilizado para dicha construcción. En resultado harán una clasificación aproximadamente perfecta. Esto, no es óptimo, ya que ajustarse demasiado a los datos de entrenamiento trae consigo que el modelo sea demasiado específico y se comporte mal para nuevos elementos. Si el conjunto de entrenamiento posee ruido, el modelo intentará ajustarse a los errores, perjudicando su comportamiento global. Este es el principal problema de los algoritmos GEP y es conocido como “sobreajuste”.

Otros de los problemas significativos de este tipo de algoritmo son:

- El proceso de evaluación de la función objetivo, puede requerir horas e incluso días en problemas de gran envergadura.
- El problema de definir una función objetivo que sea la más apropiada, para garantizar la convergencia a soluciones óptimas del problema.

- Se necesitan considerables recursos de cómputos para problemas complejos.

1.5.6 Aplicaciones

- Resumen automático de textos (Dehkordi, y otros, 2013).
- Regionalización de la duración del flujo de curva (Hashmi, y otros, 2014).
- Para la estimación total de las emisiones de la mezcla H_2O-CO_2 en la combustión Air-Fuel (Maghbooli, y otros, 2015).
- Identificación de impedancia de circuito (Janeiro, y otros, 2012).
- Predicciones macro-económicas (Jingfeng, y otros, 2009).
- Diagnóstico de enfermedades del corazón (Dai, y otros, 2009).
- Diagnóstico de cáncer de mama (Ferreira, 2006).
- Proyección de crédito (Ferreira, 2006).
- En los sistemas de tuberías de alcantarillado (Ghani, y otros, 2010).
- En Biología (H. Z., y otros, 2006).
- Sistemas de información geográfica (Eldrandaly, 2009).
- Predicción del índice de calidad del agua (Mohammadpour, y otros, 2016).
- Programación automática (Chen, y otros, 2007).
- Clasificación de granizo (Bellini Saibene, y otros, 2015).

1.6 Conclusiones parciales

En este capítulo se expusieron las características principales de los AE. Se describió el paradigma GEP y su desempeño en la tarea de clasificación. Se abordaron las ventajas y desventajas de los algoritmos: PG, AG y GEP. Además se mencionaron un rango amplio de las aplicaciones de estos en el ámbito empresarial. Todos estos elementos permitieron comprender y caracterizar el funcionamiento de un algoritmo evolutivo GEP de clasificación.

Capítulo 2 El principio MDL para la reducción del sobreajuste en la tarea de clasificación

Este capítulo tiene como objetivo caracterizar y comprender el principio MDL y realizar un bosquejo en sus aplicaciones en la tarea de clasificación. Definir el problema de sobreajuste, las causas que lo provocan y la necesidad de reducirlo. Además seleccionar métricas para medir el sobreajuste.

2.1 Principio MDL

El principio de “Longitud de Descripción Mínima” o en inglés *Minimum Description Length* (MDL) fue introducido por Jorma Rissanen en 1978 (Rissanen, 1978). MDL es un método que proporciona una solución genérica a uno de los problemas más importantes de inferencia inductiva y estadística: el de selección de modelo en términos de ganancia de información (Grunwald, 2005). Este principio es una versión formalizada del principio universal de la Navaja de Occam, donde entre diferentes teorías para explicar una determinada evidencia, se debe preferir la teoría más simple; “escoge el modelo que da la descripción más corta de los datos” (*Choose the model that gives the shortest description of data*) (Hansen, y otros, 2006-2007).

El objetivo de la inferencia estadística es tratar de encontrar la regularidad en los datos. La regularidad puede ser expresada como la capacidad de comprimir (Grunwald, 2005). MDL está basado en la idea de la comprensión de los datos (Cui, y otros, 2016). Sin embargo, a pesar de la simplicidad de la idea, representa una vista radicalmente diferente en cuanto a su modelación. Dado que, la clase del modelo tiene que ser tal que sus miembros pueden ser descritos o codificados en términos de un número finito de símbolos, por ejemplo el binario (Rissanen, 2008).

Lo anterior puede extenderse al caso de representar una serie de datos mediante un modelo, en que el que se trata de minimizar la longitud de la descripción del modelo y su

capacidad para representar al conjunto de datos. Nótese que un modelo muy complejo requerirá de una longitud de descripción mayor a un modelo sencillo, mientras que el primero representará de una manera más precisa a los datos; el principio MDL tratará de obtener un equilibrio entre ambos factores: complejidad del modelo y precisión frente a los datos (Arias, 2013).

Esta investigación sigue el método utilizado por (Quinlan, 1993; Zhou, y otros, 2003; Bacardit, 2004) para definir la longitud de descripción de un conjunto de reglas H obtenidas de los datos D:

$$L\,(H) = W^{*} L\_teoría\,(H) + L\_excepciones\,(H) \qquad (2.1)$$

Siendo:

- L_teoría (H) la longitud en bits de la descripción del conjunto de reglas;
- L_excepciones (H) la longitud en bits de los errores producidos por el conjunto de reglas; y
- W el peso que ajusta la relación entre los bits L_teoría (H) y L_excepciones (H).

El mejor modelo para describir D es el más pequeño que contiene el conjunto de reglas seleccionado (Grunwald, 2005).

La idea del principio MDL es buscar un compromiso para cubrir bastantes ejemplos y capturar así las regularidades presentes en los datos (no subajustar) sin llegar a encontrar reglas demasiado específicas (no sobreajustar).

MDL ha sido aplicado como parte de tareas de modelado en muchos campos diferentes por ejemplo, en investigaciones de imágenes (Roman, y otros, 2013; Liu, 2016) en estructuras de redes inalámbricas (Wan, 2016), en análisis de música (Louboutin, y otros, 2016), sistemas médicos (Angelopoulou, y otros, 2015), observaciones de series de tiempo (Kavcic, y otros, 2002; Farahani, y otros, 2015; Hu, y otros, 2014), en el reconocimiento de escritura a mano y brazos robóticos (Gao, y otros, 2000), en la caracterización de señales sísmicas (Jaramillo, y otros, 2014).

2.2 MDL en la tarea de clasificación

Además de las aplicaciones mencionadas en el acápite anterior (Ver **2.1 Principio MDL**), el principio también ha sido ampliamente utilizado en las tareas de clasificación. Entre los ejemplos más significativos se encuentran:

- C4.5 propuesto por Quinlan (Quinlan, 1993) a finales de los años 80. Desde entonces, ha sido uno de los sistemas clasificadores más referenciados en la bibliografía, principalmente debido a su extremada robustez en un gran número de dominios y su bajo coste computacional. C4.5 construye inicialmente un árbol de decisión, luego genera un conjunto de reglas de decisión. Las reglas generalizadas son agrupadas en conjuntos de reglas por clase, cada uno de los cuales cubren a una clase particular. Posteriormente, extrae de cada conjunto de reglas un subconjunto que maximiza la precisión en la predicción para la clase asociada aplicando el principio MDL (Giráldez, 2003).
- Algoritmo propuesto por Zhou (Zhou, y otros, 2003). La idea de este algoritmo se basa en descomponer un problema multi-clase en variados problemas de clasificación binarios y luego realizar una búsqueda evolutiva utilizando GEP (Zhou, y otros, 2003). Utiliza el principio MDL en la etapa de pre-podado para evitar el aprendizaje de reglas complicadas que cubren sólo un pequeño número de ejemplos en la clase que se está analizando.
- GAssist clasificador propuesto por Bacardit (Bacardit, 2004) que reduce el tamaño de los individuos generalizando soluciones y efectúa un aprendizaje incremental evitando usar todos los ejemplos para llevar a cabo la evaluación. Para esto se basa en el modelo Pittsburgh, utilizando un ciclo de funcionamiento de un AG. GAssist emplea una función de aptitud a base del principio MDL para promover soluciones más pequeñas y mejores (Bacardit, 2004).

Otros ejemplos encontrados fueron: en la creación de árboles de decisión (Forsyth, y otros, 1994; Patidar, y otros, 2015), la PG (Sato, y otros, 2015), en Inducción Constructiva (Perner, 2015) y en la creación de Redes Bayesianas (Kim, y otros, 2015).

2.3 El problema del sobreajuste

Generalmente, el método recursivo de un algoritmo GEP conducirá a la generación de reglas muy complejas y excesivamente ajustadas a los datos de entrenamiento. En consecuencia, hará una clasificación bastante perfecta de los datos de entrenamiento. Esto, que en principio puede parecer óptimo, en realidad no lo es, ya que ajustarse demasiado a los datos suele tener como consecuencia que el modelo sea muy específico y se comporte mal para nuevos elementos, especialmente si tenemos en cuenta que el conjunto de entrenamiento puede contener ruido (Ver **Figura 13**) o pueden faltar ejemplos representativos (Ver **Figura 14**), lo que hará que el modelo intente ajustarse a los errores, perjudicando su comportamiento para predecir nuevos resultados. Este es un problema que en general presentan todas las técnicas de aprendizaje de un modelo a partir de un conjunto de datos de entrenamiento (Aprendizaje Automático), al que se conoce como “sobreajuste” (Ver **Figura 11**) (Cawley, y otros, 2010; Díaz, y otros, 2005).

Algunos especialistas definen el sobreajuste como:

“El sobreajuste es un problema de los modelos estadísticos donde se presentan demasiados parámetros.” (Fiszelew, y otros, 2013).

“El sobreajuste: los modelos se ajustan a los datos de forma excesiva, con una expresividad exagerada.” (Lara Torralbo, 2014).

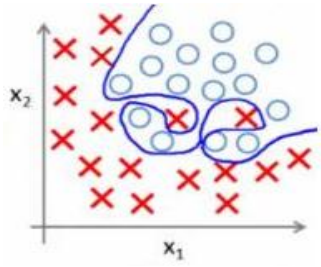

Figura 11: Modelo con sobreajuste (Collová, y otros, 2013).

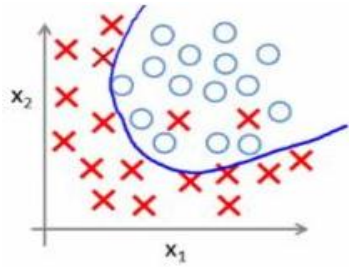

Figura 12: Modelo ajustado (Collová, y otros, 2013).

En la Figura 12 se puede ver cómo el modelo se ajusta a los puntos. La curva (modelo) pasa por muchos puntos, pero no por todos. También se observa que este modelo debería predecir correctamente nuevos puntos incluso fuera del rango observado.

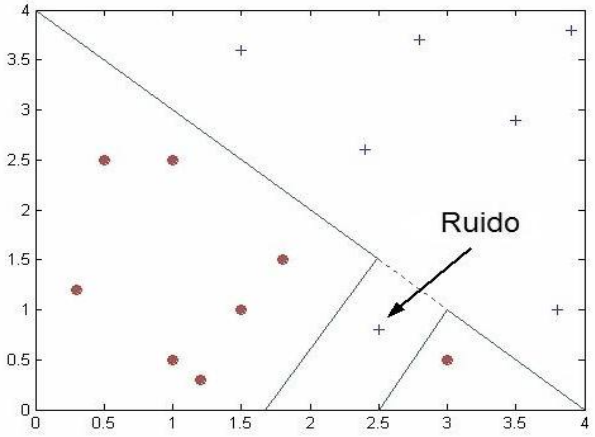

Figura 13: Datos que presentan ruido (Haykin, 2011).

En la Figura 13 se observa como los datos que presentan ruido distorsionan la frontera de decisión.

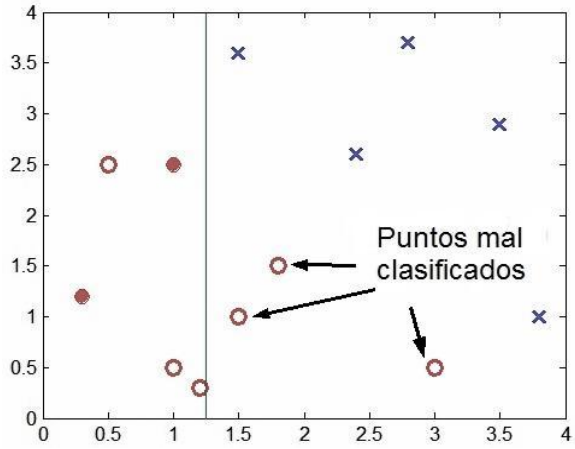

Figura 14: Falta de ejemplos representativos (Haykin, 2011).

La falta de ejemplos representativos que muestra la Figura 14 hace difícil que el modelo realice una predicción acertada en esta región.

Otras de las causas que provocan el sobreajuste son: la presencia de atributos que en los juegos de datos presentan una aparente regularidad pero que no son relevantes en realidad; y entrenar el algoritmo de aprendizaje con conjuntos de datos pequeños.

El modo más frecuente de limitar el sobreajuste en el contexto de los árboles de decisión y conjuntos de reglas consiste en eliminar condiciones de las ramas del árbol o de las reglas. Para lograr este objetivo la presente investigación se basa en el principio MDL para modificar las reglas, dando como resultados modelos más generales.

2.3.1 Medición del sobreajuste

Cuando se evalúa la calidad de un modelo o un ajuste, es importante medir el error en el conjunto de entrenamiento y en la predicción. La utilización exclusiva del error del conjunto de entrenamiento puede conducir a resultados engañosos. Para estos errores se utilizan en esta investigación las medidas:

$$ACC\_dif = \left| ACC_{entrenamiento} - ACC_{predicción} \right| \quad (2.2)$$

$$AUC\_dif = \left| AUC_{entrenamiento} - AUC_{predicción} \right| \quad (2.3)$$

Basadas en las métricas: exactitud (*accuracy* a partir de ahora) y el área bajo la curva ROC (*Area Under the Curve*, AUC).

La *accuracy* representa el porcentaje de ejemplos correctamente clasificados y se calcula (Cantador, 2005):

$$ACC = (Tp+Tn) / (Tp+Fn+Fp+Tn) \quad (2.4)$$

Donde Tp, Tn, Fn, Fp son extraídos de la matriz de confusión (Ver **Figura 10**).

Las curvas ROC (*Receiver Operating Characteristic*) explicadas por Swets en (Swets, 1988) miden el porcentaje de ejemplos positivos reconocidos correctamente en función del porcentaje de ejemplos negativos mal clasificados. El AUC es de hecho una de las métricas de eficacia mejor aceptada (Cantador, 2005). La Figura 15 muestra la representación de 3 curvas ROC. El área bajo ellas es la medida que las caracteriza y que permite definir sus diferentes grados de bondad. Es fácil entender que la curva superior (con más AUC) es la mejor, y que la curva inferior (con menos AUC) es la peor de las tres.

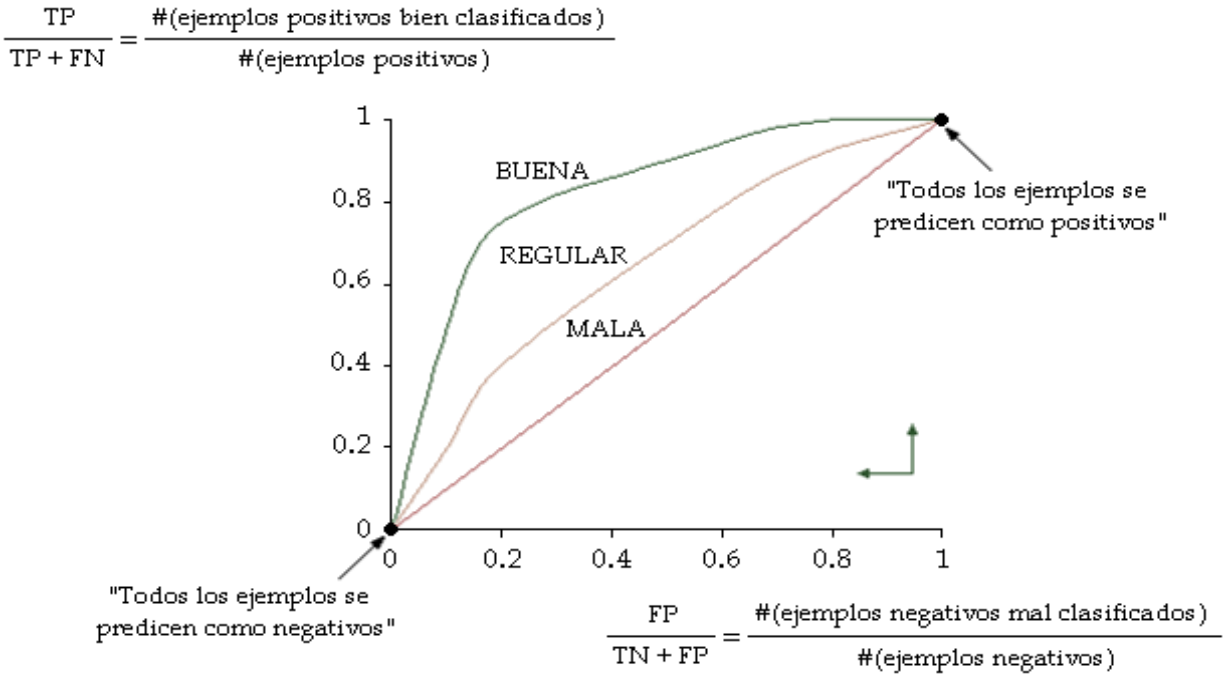

Figura 15: Ejemplo de 3 curvas ROC con diferentes grados de bondad según la métrica AUC (Cantador, 2005).

2.4 Conclusiones parciales

En este capítulo se caracterizó el principio MDL con vista a reducir el sobreajuste. Se realizó un análisis de su aplicación en la tarea de clasificación, seleccionando como referentes para esta investigación los trabajos de Zhou (Zhou, y otros, 2003) y Bacardit (Bacardit, 2004). Se definió el problema del sobreajuste y se explicaron las causas que lo provoca. Además se propusieron las métricas: ACC_dif y AUC_dif para su posterior medición en esta investigación.

Capítulo 3 Descripción de la implementación y resultados experimentales

Este capítulo tiene como objetivo definir el algoritmo GEP al cual se le aplicará el principio MDL y realizar una breve caracterización de su funcionamiento. Se define la propuesta de solución, así como la relación de los estados del nuevo clasificador mediante un diagrama de estados. Se representa el proceso de calcular el principio MDL mediante un diagrama de procesos y se exponen las características de las variantes implementadas del principio MDL. Además se describen las herramientas y tecnologías empleadas para la solución. Y por último se especifican los experimentos a ejecutar y se realiza el análisis estadístico de los resultados obtenidos.

3.1 Algoritmo MCGEP

El principio MDL se implementará en el algoritmo Programación de Expresiones Genéticas Multi Clase (*Multi-class with Gene Expression Programming*, MCGEP) introducido por (Guerrero Enamorado, y otros, 2015).

Ventajas de MCGEP:

- MCGEP es un esquema mucho más simple de entender porque se codifican los individuos exactamente como los define Ferreira (Ferreira, 2001) en su trabajo.
- Se utiliza la sencillez como criterio para el conjunto de funciones MCGEP, sólo empleando cuatro operaciones básicas (*, +, -, /), todas ellos con paridad = 2.
- El algoritmo MCGEP utiliza un método que tiende a eliminar la redundancia de reglas, priorizando las reglas más generales. Por tanto se hace más fácil la manipulación del clasificador generado porque no posee reglas repetidas.
- En comparación con otros Algoritmos Evolutivos ha demostrado estadísticamente una buena precisión a la hora de clasificar nuevos datos.

Desventajas de MCGEP:

- Las observaciones estadísticas realizadas al algoritmo MCGEP, demuestran que todavía presenta sobreajuste en sus resultados.
- MCGEP no cuenta con un método factible para reducir el sobreajuste.

3.1.1 Descripción del algoritmo MCGEP

El proceso evolutivo en MCGEP comienza con la creación de la población inicial. Esta población inicial evoluciona a través de un ciclo hasta que se alcanza una de las condiciones de parada: se obtenga el máximo número de iteraciones definidas o se consiga un individuo con valor de función de aptitud igual a uno.

Por cada iteración se realizan los siguientes pasos:

- Evaluar los individuos con la función de aptitud.
- A raíz de la evaluación realizada, seleccionar un porciento de los mejores candidatos.
- Seleccionar los individuos padres aplicando el operador genético de selección por torneo.
- Aplicar los operadores de mutación (simple), transposición (IS y RIS) y cruzamiento (por un punto y por dos puntos) en la descendencia seleccionada.
- Evaluar la nueva descendencia con la función de aptitud.
- Crear la nueva generación con un porciento de individuos de la población inicial, un porciento de los individuos que evolucionaron y el porciento de los mejores candidatos.
- Eliminar redundancias en la nueva generación.
- Evaluar la nueva generación.

Terminado el ciclo se ordenan por su valor de función de aptitud, las reglas del clasificador obtenido. Y por último se aplica el principio MDL.

3.2 Implementación del principio MDL en el algoritmo evolutivo GEP de clasificación

3.2.1 Propuesta de solución

Utilizando la investigación realizada en los capítulos anteriores, se propone implementar un algoritmo basado en el principio MDL, que reduzca el sobreajuste de los datos en el algoritmo MCGEP, sin colapsar su precisión de clasificación.

3.2.2 Diagrama de estados

En esta investigación se utiliza uno de los diagramas del Lenguaje Unificado de Modelado (UML) (Booch, y otros, 1999): el diagrama de estados. El diagrama de estados es una manera de caracterizar un cambio en un sistema es decir que los objetos que lo componen modificaron su estado como respuesta a los sucesos y al tiempo. Presenta los estados en los que puede encontrarse un objeto junto con las transiciones entre los estados, y muestra los puntos inicial y final de una secuencia de cambio de estado (Schmuller, 2000).

El diagrama de estados se emplea para definir los cambios por los que pasa el nuevo clasificador en un ciclo del algoritmo basado en el principio MDL (Ver **Anexo 1: Diagrama de estados del nuevo clasificador**). A continuación se describen sus estados:

- Creado: en este estado se crea el nuevo clasificador.
- Modificado: se adiciona al clasificador el primer elemento del conjunto de reglas obtenidas por el algoritmo MCGEP.
- Analizado para el cálculo de las instancias cubiertas: se calculan los valores de Tp, Fp, Fn, Tn del nuevo clasificador según la matriz de confusión.
- Analizado para el cálculo del principio MDL: se calcula el principio MDL al nuevo clasificador teniendo en cuenta los tres sub-estados que lo componen:
    - Analizado para el cálculo de la longitud de las excepciones: se calcula la longitud de los errores del nuevo clasificador.

- Analizado para el cálculo de la longitud de la teoría: se calcula la longitud de la teoría del nuevo clasificador.
- Analizado para el cálculo del parámetro w: se calcula el parámetro w del principio MDL teniendo en cuenta variables que dependen del nuevo clasificador.

3.2.3 Diagrama de proceso

Se emplea en esta tesis la Notación de Modelado de Procesos del Negocio o en inglés *Business Process Modeling Notation* (BPMN) (Owen, y otros, 2003), para definir la semántica del diagrama de proceso para el cálculo del principio MDL (Ver **Anexo 2: Diagrama de proceso calcular el principio MDL**). El diagrama de proceso representa gráficamente la secuencia de todas las actividades que ocurren para el cálculo del principio MDL, incluyendo toda la información que se considera necesaria para el análisis.

A continuación se realiza la descripción del diagrama de proceso: Calcular el principio MDL:

El algoritmo basado en el principio MDL crea el nuevo clasificador y solicita el conjunto de reglas generadas por el algoritmo MCGEP. Este conjunto contiene las reglas ordenadas de manera descendente según su valor de función de aptitud. Luego de que el algoritmo reciba el conjunto solicitado verifica si contiene reglas:

- Si el conjunto se encuentra vacío, se detiene la ejecución del algoritmo.
- Si el conjunto no se encuentra vacío, se selecciona su primera regla, se adiciona al nuevo clasificador y se borra del conjunto. Luego se calculan las instancias cubiertas por el nuevo clasificador teniendo en cuenta la matriz de confusión. Se calcula la longitud de las excepciones, la longitud de la teoría y el parámetro w del nuevo clasificador. Se evalúa la longitud de descripción mínima:
  - Si este valor se incrementa, se elimina la última regla agregada al nuevo clasificador y se detiene la ejecución del algoritmo.
  - Si no se incrementa, se toma este valor como la longitud de descripción mínima obtenida hasta el momento y se verifica nuevamente si el conjunto solicitado contiene reglas.

Cómo patrones de flujo o *workflow* utilizados se encuentran:

- Secuencia: este patrón es utilizado para modelar dependencia entre tareas, es decir, una tarea no puede empezar hasta que otra no haya terminado.
- Decisión exclusiva: la decisión exclusiva representa un punto en el proceso donde se debe escoger un solo camino de varios disponibles dependiendo de una decisión o de datos del proceso.

3.2.4 Descripción del algoritmo

Algoritmo: MDL

Entrada: *R* (conjunto de reglas generadas por el algoritmo MCGEP ordenadas de acuerdo a su función de aptitud).

Salida: *H* (clasificador candidato).

Inicializar las variables:

H: =Ø;

L_min: =999999999 (longitud de descripción mínima obtenida hasta el momento);

L_H: =0 (longitud de descripción mínima de H);

L_teoría: =0 (longitud de la teoría en bits);

L_excpeciones: =-1 (longitud de las excepciones en bits);

Tp, Fp, Tn, Fn: =0 (valores de instancias cubiertas según la matriz de confusión);

Repetir

regla = {r |r ϵ *R*};

H: =*H* + *regla*;

R: =*R* - *regla*;

Calcular las instancias cubiertas por *H* según la matriz de confusión;

L_excpeciones (H) = número de bits que codifica las excepciones actuales de *H*;

L_teoría (H) = número de bits que codifica la descripción del conjunto de reglas *H*;

W = valor del parámetro w;

*L_H = W * L_teoría (H) + L_excpeciones (H)*;

Si (*L_H* ≥ *L_min*) Entonces

H: =*H - regla*;

detener el algoritmo;

Si no Entonces

L_min: =*L_H*;

Fin Si

Hasta que *R*=Ø;

El algoritmo basado en el principio MDL comienza su ejecución luego de haber concluido el proceso evolutivo realizado por el algoritmo MCGEP. Cómo se muestra en el **Algoritmo: MDL** se empieza con la inicialización de las variables. Luego emprende el ciclo del algoritmo hasta que se alcance una de estas condiciones de parada: el tamaño del conjunto de reglas generadas por el algoritmo MCGEP es igual a 0 ó el valor de longitud de descripción mínima obtenido hasta el momento se incrementó. En cada corrida del ciclo se le calcula al nuevo clasificador el valor de longitud de descripción mínima hasta el momento. Para esto se utiliza la fórmula (**2.1**). De acuerdo a las investigaciones realizadas en esta tesis se definieron tres variantes, MDL1, MDL2 y MDL3 para definir los parámetros que forman parte de esta fórmula. Por tanto se obtuvieron tres versiones del algoritmo MDL. A continuación se describen cada una de estas variantes y las anotaciones que se tuvieron en cuenta para implementarlas:

MDL1

$$L\_excepciones(H) = \frac{\log_2(tp+fn)}{tp+fn} * fn + \frac{\log_2(tn+fp)}{tn+fp} * fp \quad (3.1)$$

$$L\_teoría(H) = \frac{\log_2 \#Símbolos}{\#Símbolos} * \sum_{i=1}^{s} Longitud(R_i) \quad (3.2)$$

$$W = \frac{log_2(tp+fn)}{tp+fn} * \frac{2*\#Símbolos}{log_2 \#Símbolos * Longitud_{máxima}} \quad (3.3)$$

Dónde:

- tp, fn, tn y fp son los valores de las instancias cubiertas según la matriz de confusión del clasificador en un momento dado (Ver **Figura 10**).
- #Símbolos, es el número de símbolos del algoritmo MCGEP en un problema dado (#Constantes+#Atributos+#Funciones).
- S, es el número de reglas del clasificador H.
- Longitud (R_i), es la longitud de la regla R_i del clasificador H.
- $Longitud_{máxima}$, es 2*Longitud de la cabeza de las reglas +1.
- Longitud de la cabeza, #Constantes, #Atributos, #Funciones, son definidos en el archivo de configuración del algoritmo MCGEP.

Las fórmulas (**3.1**), (**3.2**) y (**3.3**) son de elaboración propia a raíz de los estudios realizados de los trabajos de Zhannon (Shannon, 1948), Zhou (Zhou, y otros, 2003) y Bacardit (Bacardit, 2004).

MDL2

$L\_excepciones(H)$= (3.1)

$L\_teoría(H)$= (3.2)

$$W = \frac{\text{tasa\_inicial\_de\_complejidad} * L\_excepciones(H)}{(1-\text{tasa\_inicial\_de\_complejidad}) * L\_teoría\_ampliada(H)} \quad (3.4)$$

Dónde:

- tasa_inicial_de_complejidad = 0,075.
- L_teoría_ampliada (H)= L_teoría (H)*#Reglas (H)/ #Clases.
- #Reglas (H), es el número de reglas del clasificador H.
- #Clases, es el número de clases del juego de datos utilizado en el algoritmo MCGEP.

Para definir el parámetro w en esta variante del algoritmo se tuvieron en cuenta los criterios propuestos por Zhou (Zhou, y otros, 2003) y Bacardit (Bacardit, 2004):

Zhou expone que el parámetro w=0,5 y que no es sensible al algoritmo.

Los análisis hechos por Bacardit expresan que el parámetro w se vuelve muy sensible. Explica que no existe un método para ajustar este parámetro de forma automática. Sin embargo, se puede tratar de encontrar una manera para que el algoritmo realice "bastante bien" una gama amplia de problemas.

Luego de realizados experimentos previos con el trabajo de Zhou se obtuvo un resultado no deseado: tomar w=0,5 hizo que la población del algoritmo colapsara, deteriorando demasiado su precisión para clasificar nuevos datos.

Por tanto se emplea en esta variante del algoritmo la propuesta de Bacardit (**3.4**). Bacardit introduce el parámetro tasa_inicial_de_complejidad para ajustar w. Para determinar su valor realiza experimentos, llegando a la conclusión que si la tasa_inicial_de_complejidad es demasiada alta la población colapsará. Define que los valores de la tasa_inicial_de_complejidad mayores a 0,1 son peligrosos, obteniendo con las pruebas realizadas los siguientes valores confiables: 0,1, 0,075 y 0,05. Finalmente utilizando el valor 0,075 en su investigación.

En esta variante del algoritmo MDL se emplea como valor de tasa_inicial_de_complejidad el utilizado en el trabajo de Bacardit (Bacardit, 2004) : 0,075.

MDL3

$$L\_excepciones(H) = \log_2\left(\binom{tp*fp}{fp}\right) + \log_2\left(\binom{fn+tn}{fn}\right) \quad (3.5)$$

$L\_teoría(H)$= (3.2)

W= (3.4)

Dónde:

- tp, fn, tn y fp son los valores de las instancias cubiertas según la matriz de confusión del clasificador en un momento dado (Ver **Figura 10**).

En esta variante del algoritmo MDL se utiliza la fórmula propuesta por Zhou (Zhou, y otros, 2003) para el cálculo de la longitud de las excepciones (**3.5**).

3.3 Experimentación

3.3.1 Juegos de datos

Para evaluar el comportamiento del principio implementado se eligieron 10 problemas reales del repositorio UCI (Lichman, 2013). Los juegos de datos que se encuentran en el repositorio UCI son actualmente muy utilizados por la comunidad de aprendizaje automático para el análisis empírico de los AE. Para llevar a cabo los experimentos fueron utilizados los siguientes juegos de datos: Bupa, Cleveland, Ecoli, Glass, Heart, Iris, Sonar, Wine, Wpbc y Zoo. Estos juegos de datos fueron seleccionados por la siguiente razón: constan de pocas instancias, de esta manera se agiliza el proceso de evaluación experimental y no se requiere de gran costo computacional en su ejecución. En total, se tienen 4 problemas binarios, 2 problemas de tres clases y 4 problemas entre 5 y 8 clases. En la Tabla 1 se puede observar un resumen de las características de estos juegos de datos.

Tabla 1: Características de los juegos de datos (Por elaboración propia).

Id	Juego de datos	Instancias	Atributos	#Clases
bup	Bupa	345	6	2
cle	Cleveland	297	13	5
eco	Ecoli	336	7	8
gla	Glass	214	9	7
hea	Heart	270	13	2
iri	Iris	150	4	3
son	Sonar	208	60	2
win	Wine	178	13	3
wpb	Wpbc	194	32	2
zoo	Zoo	101	17	7

3.3.2 Análisis estadísticos

Para evaluar los resultados se utilizó la técnica de validación cruzada o *cross-validation* en 10 partes con 5 semillas aleatorias. Esta técnica permite garantizar la independencia de la partición entre datos de entrenamiento y prueba. La validación cruzada consiste en dividir los datos de muestra en 10 subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y el resto (9) como datos de entrenamiento. El proceso de validación es repetitivo durante 10 iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso puesto que evaluamos a partir de 10 combinaciones de datos de entrenamiento y de prueba, pero aun así tiene una desventaja: es lento desde el punto de vista computacional.

3.3.3 Tecnologías asociadas al desarrollo

Lenguaje de programación

Los lenguajes de programación están diseñados para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras, además de usarse para crear programas que controlen el comportamiento físico y lógico de una computadora.

Java: Es un lenguaje de programación orientado a objetos (Leahy, 2014). La ventaja fundamental de utilizar un lenguaje como Java que ha sido probado, mejorado y ampliado continuamente es el hecho de desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma. Además de integrar muchos frameworks y tecnologías que permiten la construcción de aplicaciones totalmente personalizadas con múltiples propósitos. Java presenta un grupo de características que se enuncian a continuación:

- Los programas Java no se compilan. En su lugar son interpretados por una aplicación conocida como la máquina virtual de Java (JVM). Gracias a ello no tienen por qué incluir todo el código y librerías propias de cada sistema.
- Java es un lenguaje simple, posee una curva de aprendizaje muy rápida.
- Fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.

Framework de programación: Biblioteca de clases Java para la Computación Evolutiva o en inglés *Java Class Library for Evolutionary Computation* (JCLEC) (Ventura, y otros, 2008) en su versión 4. JCLEC es un Framework desarrollado en el lenguaje de programación Java que da soporte a los siguientes paradigmas de la Computación Evolutiva: Algoritmos Genéticos, Programación Evolutiva, Programación Genética, Evolución Gramatical y Programación de Expresiones Genéticas.

JCLEC posee las siguientes características principales (SourceForge, 2005):

- Genericidad: cualquier tipo de algoritmo de Computación Evolutiva puede ser ejecutado usando JCLEC. La única condición necesaria es tener una población de individuos a los que se aplica iterativamente una secuencia de operaciones en evolución. El usuario puede hacer uso de cualquiera de estos marcos especializados, o incluso modificarlos para crear su propio AE personalizado.
- Fácil de usar: JCLEC ofrece diversos mecanismos que ofrecen una interfaz de programación fácil de usar. Siguiendo un estilo de programación de alto nivel, cosa que permite la creación rápida de prototipos de aplicaciones.
- Portabilidad: como el sistema JCLEC ha sido codificado en el lenguaje de programación Java se garantiza su portabilidad entre todas las plataformas que implementan una máquina virtual de Java.
- La robustez: declaraciones de verificación y validación se incrustan en el código para asegurarse de que las operaciones son válidas.
- Fuente abierta: el código fuente de JCLEC es gratuito y está disponible bajo la Licencia Pública General de GNU (GPL). Por lo tanto, se puede distribuir y modificar sin ningún pago.

Entorno de Desarrollo Integrado (IDE)

Un IDE es una aplicación compuesta por un conjunto de herramientas útiles para un programador. Un entorno suele consistir de un editor de código, un compilador, y un constructor de interfaz gráfica GUI (Alegsa, 2010).

Eclipse Java versión Luna Service Release (4.4.1): es un IDE de código abierto basado en Java y multiplataforma. Su fortaleza radica en su grado de extensibilidad. Eclipse es

una superestructura formada por un núcleo y diversos *plugins* que van conformando la funcionalidad final y soporta todo el ciclo de desarrollo (T. E. Foundation, 2010).

Ventajas:

- En Eclipse el concepto de trabajo está basado en las perspectivas, que no es otra cosa que una pre-configuración de ventanas y editores, relacionadas entre sí, y que nos permiten trabajar en un determinado entorno de trabajo de forma óptima.
- Eclipse emplea módulos (*plugins*) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Ejemplo de estos módulos tenemos el JDT (*Java Development Tools*) que viene incluido en dicha distribución del IDE.
- Incluye un potente depurador, de uso fácil e intuitivo, y que visualmente nos ayuda a mejorar nuestro código. Para ello sólo debemos ejecutar el programa en modo depuración (con un simple botón).

Herramienta de modelado: Visual Paradigm (v8.0)

Visual Paradigm es una herramienta para el desarrollo de aplicaciones utilizando modelado UML[1], ideal para ingenieros de software, analistas y arquitectos de sistemas que están interesados en construcción a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. También ofrece navegación intuitiva entre la escritura del código y su visualización; potente generador de informes en formato PDF/HTML[2]. Visual Paradigm presenta un ambiente visualmente superior de modelado y sincronización de código fuente en tiempo real (Gutiérrez, 2009). Entre las principales características de esta herramienta se encuentran:

- Utiliza UML para visualizar y diseñar los elementos de software.
- Permite la generación de código e ingeniería tanto directa como inversa.
- Los desarrolladores pueden diseñar la documentación del sistema con una plantilla de diseño.

[1] UML: Lenguaje Unificado de Modelado (por sus siglas en inglés, Unified Modeling Language).

[2] PDF/HTML: Portable Document Format/HyperText Markup Language.

Ventajas:

- Facilita el modelado de UML, porque proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, pues la misma se ajusta al estándar soportado por la herramienta.
- Corrección sintáctica. Controla que el modelado con UML sea correcto.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- Integración con otras aplicaciones. Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- Coherencia entre diagramas. Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.

Para la extracción de conocimiento basado en el aprendizaje evolutivo se utilizó la herramienta KEEL (*Knowledge Extraction based on Evolutionary Learning*) (v2.0).

KEEL: es un software de código abierto desarrollado en el lenguaje de programación Java, que faculta al usuario evaluar el comportamiento del aprendizaje evolutivo y técnicas basadas en *Soft Computing* para diferentes tipos de problemas: de regresión, clasificación, agrupación, entre otros (Alcalá-Fdez, y otros, 2011).

Ventajas:

- Reduce el trabajo de programación. Incluye una biblioteca con algoritmos de aprendizaje evolutivos basados en diferentes paradigmas (Pittsburgh, Michigan e IRL) y simplifica la integración de estos algoritmos con diferentes técnicas de pre-procesamiento. Por tanto alivia el trabajo de la programación y permite a los investigadores centrarse en el análisis de sus nuevos modelos de aprendizaje en comparación con los existentes.
- Posee una amplia biblioteca de AE que junto con la fácil utilización del software reduce considerablemente el nivel de conocimientos y experiencia necesarios de los investigadores en la Computación Evolutiva. Como resultado, los investigadores con menos conocimientos, al utilizar esta herramienta, serían capaz de aplicar con éxito estos algoritmos a sus problemas.

- Debido al uso de un estricto enfoque orientado a objetos de la biblioteca y del software, éstos se pueden utilizar en cualquier máquina con Java. Como resultado, cualquier investigador puede usar KEEL en su máquina, independientemente del sistema operativo.

Para la realización de las simulaciones se utilizó una PC: procesador Intel Core-i3-2120 con cuatro núcleos, CPU a 3,3 GHz, sistema operativo GNU Linux en su versión de Kernel 4.4.0 y 4 GB de memoria RAM.

3.3.4 Experimento

Las configuraciones utilizadas en los juegos de datos para la simulación del algoritmo se resumen en la Tabla 2. En los experimentos, se han definido el tamaño de los cromosomas y la lista de constantes según las características de los problemas. Dependiendo de la gama de valores de los atributos y de experimentos previos, se eligió la lista de constantes. Para el tamaño de la cabeza de los cromosomas se utilizó el método de ensayo y error, ajustándose con una heurística determinada por anteriores comprobaciones a valores cercanos a diez veces el número de atributos de cada juego de datos.

Tabla 2: Configuración de los juegos de datos (Por elaboración propia).

Juego de datos	Tamaño cabeza	Constantes
Bupa	60	1;0,5
Cleveland	130	1;0,5
Ecoli	70	1;0,5
Glass	90	1;0,5
Heart	130	1;0,5
Iris	40	1;0,5
Sonar	200	1;0,5
Wine	130	1;0,5
Wpbc	160	1;0,5
Zoo	170	1;0,5

3.3.5 Resultados de los experimentos

La efectividad del clasificador MCGEP con las versiones implementadas del algoritmo MDL se evaluó con las métricas *accuracy* y AUC. Para medir el sobreajuste se utilizó las medidas ACC_dif y AUC_dif. Los promedios de los resultados por cada colección se muestran en la Tabla 3 para ACC_dif y en la Tabla 4 para AUC_dif. En las tablas 5 y 6 se muestran los promedios de los resultados para *accuracy* y el AUC. La última fila de cada tabla muestra el ranking promedio de cada algoritmo.

Tabla 3: Resultados de la métrica ACC_dif (Por elaboración propia).

ACC-Dif	**Sin_MDL**	**MDL1**	**MDL2**	**MDL3**
Bupa	0,1074	0,0406	0,0406	0,0029
Cleveland	0,2125	0,0906	0,0906	0,0002
Ecoli	0,0871	0,0572	0,0572	0,0018
Glass	0,1577	0,0808	0,0808	0,0010
Heart	0,0961	0,0529	0,0529	0,0097
Iris	0,0348	0,0147	0,0147	0,0000
Sonar	0,2247	0,1419	0,1419	0,0231
Wine	0,0726	0,0480	0,0480	0,0042
Wpbc	0,1960	0,1279	0,1279	0,0498
Zoo	0,0656	0,0434	0,0434	0,0296
Ranking	**4,0**	**2,5**	**2,5**	**1,0**

Tabla 4: Resultados de la métrica AUC_dif (Por elaboración propia).

AUC-Dif	**Sin_MDL**	**MDL1**	**MDL2**	**MDL3**
Bupa	0,1076	0,0415	0,0415	0,0012
Cleveland	0,1797	0,0520	0,0520	0,0030
Ecoli	0,0098	0,0398	0,0398	0,0374
Glass	0,0393	0,0034	0,0034	0,0033

Heart	0,0972	0,0526	0,0526	0,0124
Iris	0,0174	0,0073	0,0073	0,0000
Sonar	0,2241	0,1422	0,1422	0,0249
Wine	0,0376	0,0231	0,0231	0,0023
Wpbc	0,3091	0,1915	0,1915	0,0329
Zoo	0,0157	0,0103	0,0103	0,0322
Ranking	**3,6**	**2,5**	**2,5**	**1,4**

Tabla 5: Resultados del *accuracy* para los datos de prueba (Por elaboración propia).

ACC	**Sin_MDL**	**MDL1**	**MDL2**	**MDL3**
Bupa	0,6811	0,6927	0,6927	0,4234
Cleveland	0,5289	0,5443	0,5443	0,5389
Ecoli	0,7710	0,7211	0,7211	0,4238
Glass	0,6426	0,6387	0,6387	0,3281
Heart	0,7911	0,7911	0,7911	0,5681
Iris	0,9587	0,9653	0,9653	0,3333
Sonar	0,7353	0,7012	0,7012	0,4412
Wine	0,9219	0,9178	0,9178	0,3273
Wpbc	0,7147	0,7045	0,7045	0,2932
Zoo	0,9344	0,9562	0,9562	0,4355
Ranking	**2,0**	**2,1**	**2,1**	**3,9**

Tabla 6: Resultados del AUC para los datos de prueba (Por elaboración propia).

AUC	**Sin_MDL**	**MDL1**	**MDL2**	**MDL3**
Bupa	0,6707	0,6760	0,6760	0,5013
Cleveland	0,6803	0,7034	0,7034	0,7920

Ecoli	0,9609	0,9446	0,9446	0,9124
Glass	0,9034	0,9008	0,9008	0,8366
Heart	0,7877	0,7878	0,7878	0,5157
Iris	0,9793	0,9827	0,9827	0,6667
Sonar	0,7358	0,6992	0,6992	0,4729
Wine	0,9599	0,9610	0,9610	0,6644
Wpbc	0,5677	0,5706	0,5706	0,5370
Zoo	0,9843	0,9895	0,9895	0,8893
Ranking	**2,5**	**1,9**	**1,9**	**3,7**

3.4 Análisis estadístico de los resultados

Se han seguido las recomendaciones señaladas por Demsar (Demsar, 2006) para llevar a cabo el análisis estadístico de los resultados. Como se indica, se utilizaron pruebas estadísticas no paramétricas de la herramienta KEEL para comparar las métricas ACC-Dif, AUC_dif, ACC y AUC de los modelos construidos por las diferentes variantes del algoritmo MDL. Para comparar dichos algoritmos, primero aplicamos un procedimiento estadístico de múltiple comparación para poner a prueba la hipótesis nula de que no existen diferencias entre los algoritmos (con y sin MDL). Específicamente, se empleó el test de Friedman. El test de Friedman rechazó la hipótesis nula en todos los casos (Ver **Tabla 7**). Por tanto se aplicó el test de Holm para detectar entre cuáles de los algoritmos existen diferencias significativas. En la Tabla 7 se muestra el resumen del análisis estadístico, los valores que se encuentran dentro de los paréntesis indican la variante que ganó en la disputa realizada por el test de Holm.

Métrica ACC_dif: en esta métrica el test de Friedman plantea que existen diferencias significativas en los algoritmos con un valor de p=5,89*10-6 (valor de p<0,05). Ganando en el ranking (Ver **Tabla 3**, última fila) la variante MDL3 y quedando en segundo lugar las variantes MDL1 y MDL2.

El resultado de test de Holm con un valor de α de 0,05 indica diferencias significativas entre todas las variantes, excepto en el algoritmo MDL1 con respecto al MDL2.

Métrica AUC_dif: en esta métrica el test de Friedman revela que hay diferencias significativas en los algoritmos con un valor de $p=2,28*10^{-3}$ (valor de $p<0,05$). Ganando en el ranking (Ver **Tabla 4**, última fila) la variante MDL3 y quedando nuevamente en segundo lugar las variantes MDL1 y MDL2.

El resultado de test de Holm con un valor de α de 0,05 demuestra que hay diferencias significativas entre la variante MDL3 con respecto al Sin_MDL.

Métrica ACC: en esta métrica el test de Friedman muestra que existen diferencias significativas en los algoritmos con un valor de $p=1,31*10^{-3}$ (valor de $p<0,05$). Ganando en el ranking (Ver **Tabla 5**, última fila) la variante Sin_MDL, quedando en segundo lugar las variantes MDL1 y MDL2 y en último lugar MDL3.

El resultado de test de Holm con un valor de α de 0,05 plantea que hay diferencias significativas entre las variantes Sin_MDL, MDL1 y MDL2 con respecto a MDL3.

Métrica AUC: en esta métrica el test de Friedman expone que existen diferencias significativas en los algoritmos con un valor de p=4,72*10-3 (valor de p<0,05). Ganando en el ranking (Ver **Tabla 6**, última fila) las variantes MDL1 y MDL2, teniendo en segundo lugar las variantes MDL0 y en último lugar MDL3.

El resultado de test de Holm con un valor de α de 0,05 muestra diferencias significativas entre las variantes MDL1 y MDL2 con respecto a MDL3.

Tabla 7: Resumen del análisis estadístico (Por elaboración propia).

		ACC_dif	**AUC_dif**	**ACC**	**AUC**
Test de Friedman	**valor de p**	$5,89*10^{-6}$	$2,28*10^{-3}$	$1,31*10^{-3}$	$4,72*10^{-3}$
Test de Holm	**MDL3 vs MDL2**	Si (3)	No (-)	Si (2)	Si (2)
	MDL3 vs MDL1	Si (3)	No (-)	Si (1)	Si (1)
	MDL3 vs Sin_MDL	Si (3)	Si (3)	No (-)	Si (Sin_MDL)
	MDL2 vs MDL1	No (-)	No (-)	No (-)	No (-)
	MDL2 vs Sin_MDL	Si (2)	No (-)	No (-)	No (-)
	MDL1 vs Sin_MDL	Si (1)	No (-)	No (-)	No (-)

3.5 Conclusiones parciales

En este capítulo se abordaron los aspectos fundamentales de la propuesta de solución. Se definieron las herramientas y tecnologías a utilizar, como framework de desarrollo JCLEC v4.0, como IDE Eclipse en su versión Luna Service Release (4.4.1), como lenguaje de programación Java, para el modelado la herramienta case Visual Paradigm v8.0 y para la extracción de conocimiento basado en el aprendizaje evolutivo se utilizó la herramienta KEEL v2.0. Se ejecutaron los experimentos y se realizaron los test estadísticos obteniéndose como resultados:

1. Las variantes implementadas MDL1 y MDL2 redujeron significativamente el sobreajuste (en ACC_dif) y mejoraron la eficiencia (en AUC) en el clasificador MCGEP.
2. El algoritmo MDL3 fue el único que mejoró el sobreajuste en las dos métricas utilizadas (ACC_dif y AUC_dif), pero lo hizo a expensas de deteriorar los valores de ACC y AUC del clasificador MCGEP.

Conclusiones

La investigación desarrollada y los resultados obtenidos permitieron arribar a las siguientes conclusiones:

1. Con la fundamentación de la codificación de algoritmos evolutivos GEP y sus aplicaciones en clasificación, se logró caracterizar y comprender el proceso evolutivo llevado a cabo por el algoritmo de clasificación MCGEP.
2. La fundamentación de las aplicaciones del principio MDL en diferentes algoritmos de clasificación, permitió proponer una nueva propuesta de cálculo del principio y definir su integración con el algoritmo de clasificación MCGEP.
3. Se implementó el principio MDL en el algoritmo de clasificación MCGEP.
4. La experimentación con juegos de datos comprobó el correcto funcionamiento del principio implementado.
5. El método de validación cruzada utilizado en el algoritmo clasificador MCGEP con el principio implementado, garantizó la independencia entre los datos de entrenamiento y prueba.
6. La aplicación de los test estadísticos demostró que:
   - Las variantes implementadas MDL1 y MDL2 redujeron significativamente el sobreajuste (en ACC_dif) y mejoraron la eficiencia (en AUC) en el clasificador MCGEP.
   - El algoritmo MDL3 fue el único que mejoró el sobreajuste en las dos métricas utilizadas (ACC_dif y AUC_dif), pero lo hizo a expensas de deteriorar los valores de ACC y AUC del clasificador MCGEP.
7. Por todo lo anterior las variantes MDL1 y MDL2 son preferibles a la variante MDL3 como solución al problema tratado en esta investigación.

Recomendaciones

A fin de enriquecer el trabajo realizado se recomienda:

- Ampliar la experimentación a un mayor número de colecciones de datos.
- Explorar otras variantes de implementación del principio MDL.
- Incluir en la experimentación otras métricas para medir el sobreajuste.
- Aplicar esta investigación en los sistemas realizados en la Universidad de las Ciencias Informáticas que necesiten clasificar información.

.

Bibliografía

Ahumada, Dania Isabel . 2015. Una aproximación evolucionista para la generación automática de sentencias SQL a partir de ejemplos. [En línea] 2015. https://papyrus.bib.umontreal.ca/xmlui/bitstream/handle/1866/12454/Ahumada_Pardo_Dania_Isabel_2015_memoire.pdf?sequence=2&isAllowed=y.

Alcalá-Fdez, Jesús, y otros. 2011. *KEEL Data-Mining Software Tool: Data Set Repository and Integration of Algorithms and Experimental Analysis Framework.* s.l. : Citeseer, 2011. Vol. 17.

Alegsa, Leandro. 2010. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. [En línea] 2010. [Citado el: 2 de febrero de 2016.] http://www.alegsa.com.ar/Dic/ide.php.

Angelopoulou, A., y otros. 2015. 3D reconstruction of medical images from slices automatically landmarked with growing neural models. [En línea] 2015. http://www.sciencedirect.com/science/article/pii/S0925231214012223.

Arias, Jacinto. 2013. Aprendizaje Exacto de Multiclasificadores Basados en AODE. [En línea] 2013. https://ruidera.uclm.es/xmlui/bitstream/handle/10578/3048/TFM%20Arias%20Mart%C3%ADnez.pdf?sequence=3&isAllowed=y.

Ávila Jiménez, José Luis, Gibaja, Eva y Ventura, Sebastián. 2010. *Evolving Multi-label Classification Rules with Gene Expression Programming: A Preliminary Study.* s.l. : Springer, 2010.

Bacardit, Jaume. 2004. Pittsburgh Genetic-Based Machine Learning in the Data Mining era: Representations,generalization, and run-time. [Tesis Doctoral]. s.l. : Universitat Ramon Llull, 2004.

Back, T. 1996. Evolutionary Algorithms in Theory and Practice. [En línea] 1996. http://hdl.handle.net/961944/14020.

Behnia, Danial, y otros. 2013. Predicting crest settlement in concrete face rockfill dams using adaptive neuro-fuzzy inference system and gene expression programming intelligent methods. [En línea] 2013. http://files.matlabsite.com/docs/papers/springer/fuzzy-logic/10.1631_jzus.a1200301.pdf.

Bellini Saibene, Yanina y Volpacchio, Martín . 2015. Clasificación de granizo en superficie usando técnicas de minería de datos y datos de radar meteorológico. [En línea]

2015. http://sedici.unlp.edu.ar/bitstream/handle/10915/52109/Documento_completo.pdf-PDFA.pdf?sequence=1.

Bikakis, Antonis y Zheng, Xianghan . 2015. *Multi-disciplinary Trends in Artificial Intelligence.* s.l. : Springer, 2015.

Blokdijk, Gerard. 2015. Data Mining - Simple Steps to Win, Insights and Opportunities for Maxing Out Success. [En línea] 2015. http://www.vidyanikethan.edu/engneering/downloads/syllabus/new/MTech_%28SVEC-14%29_Regulations%28CS%29.pdf.

Booch, Grady, Rumbaugh, Jim y Jacobson, Ivar. 1999. *El Lenguaje Unificado de Modelado.* s.l. : Addison-Wesley, 1999.

Booker, L.B., Goldberg, D.E. y Holland, J.H. 1989. Classifier systems and genetic algorithms. Artificial Intelligence. [En línea] 1989. http://www.sciencedirect.com/science/article/pii/0004370289900507.

Bremermann, H.J. 1962. Optimization through evolution and recombination. [En línea] 1962. http://www-public.tem-tsp.eu/~gibson/Teaching/CSC4504/ReadingMaterial/Bremermann62.pdf.

Calero, Coral y Piattini, Mario . 2015. *Green in Software Engineering.* s.l. : Springer, 2015.

Cantador, Iván . 2005. Aplicación de Perceptrones Paralelos y AdaBoost a Problemas de Clasificación de Muestra Extrema. [En línea] 2005. https://www.ia.urjc.es/jspTIN2007/ActasPDF/TIN2004-07676-C02.pdf.

Cárdenas, Miguel. 2015. Algoritmos Evolutivos Multiobjetivos. [En línea] 2015. http://wwwae.ciemat.es/~cardenas/docs/lessons/MOEA.pdf.

Cawley, Gavin C. y Talbot, Nicola L. . 2010. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. [En línea] 2010. http://delivery.acm.org/10.1145/1860000/1859921/11-2079-cawley.pdf?ip=10.8.125.21&id=1859921&acc=PUBLIC&key=223837E73163AEDA.7CA22F5EB1578DD1.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=793265283&CFTOKEN=25051091&__acm__=1464728552_f50011b37ea8f3ddfba87ad11a0d.

Chen, Y., y otros. 2007. An Auto-clustering Algorithm Based on Gene Expression Programming. [En línea] 2007. http://en.cnki.com.cn/Article_en/CJFDTOTAL-SCLH200706022.htm.

Chorianopoulos, Konstantinos , Divitini, Monica y Baalsrud, Jannicke . 2015. *Entertainment Computing - ICEC 2015.* s.l. : Springer, 2015.

Coello Coello, Carlos A. 2011. Algoritmos Genéticos y sus Aplicaciones. [En línea] 2011. http://aisii.azc.uam.mx/mcbc/Cursos/IntCompt/Lectura18.pdf.

Coello, Carlos A. 2015. Introducción a la Computación Evolutiva (Notas de Curso). [En línea] 2015. delta.cs.cinvestav.mx/~ccoello/compevol/apuntes.pdf.gz.

Collová, Francesco y Capaldo, Raffaele. 2013. Machine Learning: Introduction to Neural Networks. [En línea] 2013. http://es.slideshare.net/fcollova/introduction-to-neural-network.

Cui, Wanyun, y otros. 2016. Verb Pattern: A Probabilistic Semantic Representation on Verbs. [En línea] 2016. http://gdm.fudan.edu.cn/GDMWiki/attach/Wanyun%20Cui/verbpattern.pdf.

Dai, W., Zhang, Y. y Gao, X. 2009. The application of gene expression programming in the diagnosis of heart disease. [En línea] 2009. http://europepmc.org/abstract/med/19334550.

Darwin, Charles. 1859. *The Origin of Species.* s.l. : John Murray, 1859.

De Jong, K.A., Spears, W.M. y Gordon, D.F. 1993. *Using genetic algorithms for concept learning.* s.l. : Springer, 1993.

Dehkordi, Pouya Khosravian y Kyoumarsi, Farshad . 2013. *Using Gene Expression Programming in Automatic Text Summarization.* s.l. : Springer, 2013.

Demsar, Janez. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. [En línea] 2006. http://machinelearning.wustl.edu/mlpapers/paper_files/Demsar06.pdf.

Díaz, Zuleyka, Fernández Menéndez, José y Segovia Vargas, Jesús . 2005. *El Algoritmo See5 versus la metodología Rough Set. Una aplicación a la predicción de la insolvencia en empresas Españolas de seguros no-vida.* s.l. : Cuadernos de Estudios Empresariales, 2005.

Dos Santos, Danny. 2016. Sistema Autónomo de Navegación en Robots con reconocimiento de patrones Generales Regulares. [En línea] 2016. revistasenlinea.saber.ucab.edu.ve/temas/index.php/tekhne/article/viewFile/2876/2509.

Eiben, A.E. y Michalewicz, Z. 1998. *Evolutionary Computation.* s.l. : IOS Press, 1998.

Eiben, A.E. y Smith, J.E. 2003. *Introduction to Evolutionary Computing.* s.l. : Springer, 2003.

Eldrandaly, Khalid A. . 2009. Integrating gene expression programming and geographic information systems for solving a multi site land use allocation problem. [En línea] 2009. https://www.researchgate.net/profile/Khalid_Eldrandaly/publication/26625460_Integrating_Gene_Expression_Programming_and_Geographic_Information_Systems_for_Solving_a_Multi_Site_Land_Use_Allocation_Problem/links/02bfe50ec77f80cabe000000.pdf.

Estevez Valencia, P. 2000. Optimización mediante Algoritmos Genéticos. [En línea] 2000.

https://www.researchgate.net/profile/Pablo_Estevez/publication/228708779_Optimizacion_Mediante_Algoritmos_Geneticos/links/0912f51111f82b2a61000000.pdf.

Farahani, R. V. y Penumadu, D. 2015. *Full-scale bridge damage identification using time series analysis of a dense array of geophones excited by drop weight.* s.l. : Structural Control and Health Monitoring, 2015.

Ferreira, C. 2004. Gene Expression Programming and The Evolution of Computer Programs. [En línea] 2004. http://www.gene-expression-programming.com/webpapers/ferreira-bic2004.pdf.

Ferreira, Cândida. 2006. *Gene Expression Programming Mathematical Modeling by an Artificial Intelligence.* s.l. : Springer, 2006.

—. 2001. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. [En línea] 2001. https://arxiv.org/ftp/cs/papers/0102/0102027.pdf.

Fiszelew, A. y García, R. 2013. GENERACIÓN AUTOMÁTICA DE REDES NEURONALES CON AJUSTE DE PARÁMETROS BASADO EN ALGORITMOS GENÉTICOS. [En línea] 2013. http://laboratorios.fi.uba.ar/lsi/R-ITBA-26-rrnn-ags.pdf.

Fogel, L.J., Owens, A.J. y Walsh, M.J. 1966. *Artificial Intelligence through Simulated Evolution.* s.l. : Wiley-IEEE Press, 1966.

—. 1965. *Artificial Intelligence Through Simulation of Evolution.* s.l. : John Wiley & Sons, 1965.

Forsyth, R., Clarke, D. D. y Wright, R. L. 1994. Overfitting revisited: An information-theoretic approach to simplifying discrimination trees. [En línea] 1994. http://www.tandfonline.com/doi/abs/10.1080/09528139408953790.

Freitas, A.A. 2001. *A survey of evolutionary algorithms for data mining and knowledge discovery.* s.l. : Springer, 2001.

Freitas, A.A., Zytkow, J. y Klosgen, W. 2001. *Handbook of Data Mining and Knowledge Discovery.* s.l. : Oxford University Press, 2001.

Gandomi, Amir H. , Alavi, Amir H. y Ryan, Conor . 2015. *Handbook of Genetic Programming Applications.* s.l. : Springer, 2015.

Gao, Q., Li, M. y Vitányi, P. 2000. *Applying mdl to learn best model granularity.* s.l. : Elsevier, 2000.

García Gutiérrez, José Alberto . 1980. Análisis e implementación de Algoritmos Evolutivos para la optimización de simulaciones en Ingeniería Civil. [En línea] 1980. https://arxiv.org/ftp/arxiv/papers/1401/1401.5054.pdf.

Gendreau, M. y Potvin, J.Y. 2010. *Handbook of Metaheuristics.* s.l. : Springer Science & Business Media, 2010.

Gestal, Marcos, y otros. 2010. *Introducción a los Algoritmos Genéticos y la Programación Genética.* s.l. : Universidade da Coruña, Servicio de Publicación, 2010.

Ghani, Aminuddin y Azamathulla , H. 2010. Gene-expression programming for sediment transport in sewer pipe systems. [En línea] 2010. http://redac.eng.usm.my/html/publish/2010_19.pdf.

Giráldez, Raúl. 2003. Mejoras en Eficiencia y Eficacia de Algoritmos Evolutivos para Aprendizaje Supervisado. [En línea] 2003. http://fondosdigitales.us.es/tesis/tesis/2077/mejoras-en-eficiencia-y-eficacia-de-algoritmos-evolutivos-para-aprendizaje-supervisado/.

González, S.L., y otros. 2015. Algoritmos de clasificación y redes neuronales en la observación automatizada de registros. [En línea] 2015. revistas.um.es/cpd/article/download/223051/173291.

Graff, Mario. 2010. Modelado de Algoritmos Evolutivos: Un enfoque práctico. [En línea] 2010. http://dsc.itmorelia.edu.mx/~jcolivares/documents/modalgogen.pdf.

Grunwald, Peter. 2005. A Tutorial Introduction to the Minimum Description Length Principle. [En línea] 2005. sites.google.com/site/mkrishnarhul/IntroMDL.pdf.

Guerrero Enamorado, Alain, Morell, Carlos y Ventura, Sebastián. 2015. An algorithm evaluation for classification rules discovering with gene expression programming. [En línea] 2015. http://www.tandfonline.com/doi/pdf/10.1080/18756891.2016.1150000?redirect=1.

Gutiérrez, Demián. 2009. UML Diagramas de Paquetes (UML ilustrado). [En línea] 2009. http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

H. Z., Si, y otros. 2006. *QSAR study of 1, 4-dihydropyridine calcium channel antagonists based on gene expression programming.* s.l. : Elsevier, 2006.

Hansen, Mark H. y Yu, Bin. 2006-2007. Model Selection and the Principle of Minimum Description Length. [En línea] 2006-2007. http://www.exercicescorriges.com/i_214326.pdf.

Hashmi, Muhammad Z. y Shamseldin, Asaad Y. 2014. *Use of Gene Expression Programming in regionalization of flow duration curve.* s.l. : Elsevier, 2014.

Haykin, Simon. 2011. Neural Networks: A Comprehensive Foundation. [En línea] 2011. http://tocs.ulb.tu-darmstadt.de/71836187.pdf.

Holland, J.H. 1975. *Adaption in Natural and Artificial Systems.* s.l. : London: The MIT Press, 1975.

—. 1973. *Genetic algorithms and the optimal allocation of trials.* s.l. : SIAM, 1973.

Hu, Bing , y otros. 2014. *Using the minimum description length to discover the intrinsic cardinality and dimensionality of time series.* s.l. : Springer, 2014.

Janeiro, Fernando M. y Ramos, Pedro M. 2012. *Gene expression programming and genetic algorithms in impedance circuit identification.* s.l. : IMEKO, 2012.

Jaramillo, Carolina E. , y otros. 2014. *Caracterización de señales sísmicas del Volcán Cotopaxi utilizando estimadores espectrales clásicos y de máxima entropía.* s.l. : Maskana, 2014.

Jingfeng, Yan y Guoqing, Li. 2009. *The Application of Macro-Economic Prediction Based on Improved Gene Expression Programming.* s.l. : IEEE, 2009.

Kavcic, A. y Srinivasan, M. 2002. *The minimum description length principle for modeling.* s.l. : IEEE , 2002.

Kim, K. J. y Cho, S. B. 2015. *Ensemble bayesian networks evolved with speciation for high-performance prediction in data mining.* s.l. : Springer, 2015.

Koza, J.R. 1992. *Genetic Programming.* s.l. : MIT press, 1992.

—. 1994. *Genetic Programming II.* s.l. : MIT Press, 1994.

Langdon, W.B. y Poli, R. 2002. *Foundations of Genetic Programming.* s.l. : MIT Press, 2002.

Lara Torralbo, Juan Alfonso . 2014. Fundamentos y Aplicaciones Prácticas del Descubrimiento de Conocimiento en Bases de Datos. [En línea] 2014. http://repositorio.cedia.org.ec/handle/123456789/965.

Lara, Juan Alfonso. 2010. Marco de Descubrimiento de Conocimiento para Datos Estructuralmente Complejos con Énfasis en el Análisis de Eventos en Series Temporales. [En línea] 2010. http://oa.upm.es/5729/.

Leahy, Paul. 2014. What Is the Java Computer Programming Language? [En línea] Diciembre de 2014. [Citado el: 2 de Abril de 2016.] http://java.about.com/od/gettingstarted/a/whatisjava.htm.

Lichman, M. 2013. UCI Machine Learning Repository. [En línea] 2013. [Citado el: 3 de 12 de 2015.] https://archive.ics.uci.edu/ml/citation_policy.html.

Liu, Lijuan. 2016. *Discriminative Feature Learning with an Optimal Pattern Model for Image Classification.* s.l. : Springer, 2016.

Louboutin, Corentin y Meredith, David. 2016. Using general-purpose compression algorithms for music analysis. [En línea] 2016. http://www.chromamorph.net/papers/public/rapport.pdf.

Maghbooli, Babak, Najafi, Hamidreza y Sobati, Mohammad Amin. 2015. Application of Gene Expression Programming for estimating total emissivity of H2O−CO2 mixtures in Air−Fuel combustion without soot formation. [En línea] 2015. www.dl.begellhouse.com/journals/648192910890cd0e,776515bc70e6a5e8,19e0e68538e3c7e3.html.

Martínez Páez, José J. 2001. Conceptos básicos de programación genética. [En línea] 2001. http://www.revistas.unal.edu.co/index.php/revfacmed/article/viewFile/19750/20851.

Mazuera, Cristina y Henao, Yenifer. 2013. Programación de turnos para los tecnólogos en radiología e imágenes diagnóstica en las sedes de catrago de la empresa radiólogos asociados S.A.S usando Algoritmos de Inteligencia Artificial. [En línea] 2013. http://repositorio.utp.edu.co/dspace/bitstream/11059/3592/1/0063M476.pdf.

Melin, Patricia , Castillo, Oscar y Kacprzyk, Janusz. 2015. *Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization.* s.l. : Springer, 2015.

Michalewicz, Zbigniew. 1996. *Genetic algorithms + data structures = evolution programs.* s.l. : Springer Science & Business Media, 1996.

Mohammadpour, R., y otros. 2016. *Prediction of water quality index in free surface constructed wetlands.* s.l. : Springer, 2016.

Molina, Carlos A. 2015. Reconstrucción 3D de bayas de racimos de uva de vinificación basada en detección de elipses sobre imágenes de estéreo. [En línea] 2015. https://riunet.upv.es/bitstream/handle/10251/51923/Tesina_Carlos_Molina.pdf?sequence=1.

Muñoz, Mario A., López, Jesús A. y Caicedo, Eduardo F. . 2008. Inteligencia de enjambres: sociedades para la solución de problemas. [En línea] 2008. http://www.scielo.org.co/pdf/iei/v28n2/v28n2a15.pdf.

Muñoz, Mario A., López, Jesús A. y Caicedo, Eduardo F. 2008. Swarm intelligence: problem-solving societies. [En línea] 2008. http://revistas.unal.edu.co/index.php/ingeinv/article/view/14901.

Nandi, Asoke K., Fa, Rui y Abu, Basel . 2015. *Integrative Cluster Analysis in Bioinformatics.* s.l. : John Wiley & Sons, 2015.

Olmo, Juan Luis. 2013. Minería de Datos mediante Programación Automática con Colonias de Hormigas. [En línea] 2013. http://helvia.uco.es/xmlui/bitstream/handle/10396/9498/2013000000722.pdf?sequence=1.

Ortuño, Francisco y Rojas, Ignacio . 2015. *Bioinformatics and Biomedical Engineering.* s.l. : Springer, 2015.

Owen, Martin y Raj, Jog . 2003. *BPMN and Business Process Management Introduction to the New Business Process Modeling Standard.* s.l. : Citeseer, 2003.

Patidar, P., Dangra, J. y Rawar, M. K. 2015. Decision Tree C4. 5 algorithm and its enhanced approach for Educational Data Mining. [En línea] 2015. http://eusrm.com/PublishedPaper/7Vol/Issue2/2015EUSRM2201521205-62dada95-d654-42bf-a5a0-c8c58891f30349892.pdf.

Pedemonte, Martín, N. S. 2003. Estudio Empírico de Operadores de Cruzamiento en un Algoritmo Genético. [En línea] 2003. http://sedici.unlp.edu.ar/bitstream/handle/10915/22719/217.pdf?sequence=1.

Perner, P. 2015. *Decision tree induction methods and their application to big data. In Modeling and Processing for Next-Generation Big-Data Technologies.* s.l. : Springer, 2015.

Quinlan, J. R. 1993. *C4.5: programs for machine learning.* s.l. : Elsevier, 1993.

Real Academia de Ingeniería. 2012. Real Academia de Ingeniería. *Real Academia de Ingeniería.* [En línea] Registrado en el Registro de la Propiedad Intelectual. España, 2012. [Citado el: 6 de enero de 2016.] http://www.raing.es.

Rechenberg, I. 1973. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution.* s.l. : JSTOR, 1973.

Resta, Marina . 2015. *Computational Intelligence Paradigms in Economic and Financial Decision Making.* s.l. : Springer, 2015.

Rissanen, J. 2008. An Introduction to the MDL Principle. [En línea] 2008. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.2499&rep=rep1&type=pdf.

Rissanen, Jorma. 1978. *Modeling by shortest data description.* s.l. : Elsevier, 1978.

Rivas, Joel y Perozo, Freddy. 2016. Una herramienta de meta-evolución paralela para la entonación de programas evolutivo. [En línea] 2016. http://revistasenlinea.saber.ucab.edu.ve/temas/index.php/tekhne/article/viewFile/2656/2332.

Roman, Avid, Reynaga , Camilo J. y Ganvini, Cristhian. 2013. A General Method Based on Data Compression for Manipulated Image Detection. [En línea] 2013. https://dialnet.unirioja.es/descarga/articulo/4814385.pdf.

Rosales, Alejandro . 2016. *Surrogate-Assisted Evolutionary.* s.l. : Elsevier, 2016.

Sato, H., y otros. 2015. *Improved sampling using loopy belief propagation for probabilistic model building genetic programming.* s.l. : Elsevier, 2015.

Schmuller, Joseph. 2000. *Aprendiendo UML en 24 Horas.* s.l. : Pearson educación, 2000.

Schwefel, H.-P. 1977. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie.* s.l. : Basel, 1977.

Shannon, C. E. 1948. A mathematical theory of communication. [En línea] 1948. http://lanethames.com/dataStore/ECE/InfoTheory/shannon.pdf.

Smith, S.F. 1980. A learning system based on genetic adaptive algorithms. [En línea] 1980. http://dl.acm.org/citation.cfm?id=909835.

—. 1983. *Flexible learning of problem solving heuristics through adaptive search.* s.l. : JCA, 1983.

Soule, T. y Foster, J.A. 1997. Code Size and Depth Flows in Genetic Programming. [En línea] 1997. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.6051&rep=rep1&type=pdf.

SourceForge. 2005. JCLEC. *JCLEC.* [En línea] SourceForge, 2005. [Citado el: 2 de febrero de 2016.] http://jclec.sourceforge.net/.

Swets, J. 1988. Measuring the Accuracy of Diagnostic Systems. [En línea] 1988. http://science.sciencemag.org/content/240/4857/1285.full.pdf+html.

T. E. Foundation. 2010. Eclipse. *What is eclipse?* [En línea] 2010. [Citado el: 3 de Abril de 2016.] https://eclipse.org/org/#about.

Teodorescu, L. y Sherwood, D. 2008. *High energy physics event selection with gene expression programming.* s.l. : Elsevier, 2008.

Ventura, Sebastián, y otros. 2008. *JCLEC: A Java Framework for Evolutionary Computation.* s.l. : Springer, 2008.

Villagra, A., y otros. 2006. Algoritmos Evolutivos y su Aplicabilidad en la tarea de Clasificación. [En línea] 2006. http://sedici.unlp.edu.ar/bitstream/handle/10915/20711/627%201.pdf?sequence=1.

Wan, Liangtian. 2016. *Distributed DOA Estimation for Arbitrary Topology Structure of Mobile Wireless Sensor Network Using Cognitive Radio.* s.l. : Springer, 2016.

Weinert, Wagner R. y Lopes, Heitor S. 2006. *GEPCLASS: A Classification Rule Discovery Tool Using Gene Expression Programming.* s.l. : Springer, 2006.

Witten, E. y Frank, I.H. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* s.l. : Morgan Kaufmann, 1999.

Zhou, Chi, y otros. 2003. Evolving Accurate and Compact Classification Rules With Gene Expression Programming. [En línea] 2003. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.205.5251&rep=rep1&type=pdf.

Glosario de términos

Algoritmo: grupo finito de operaciones organizadas de manera lógica y ordenada que permite solucionar un determinado problema. Método y notación en las distintas formas del cálculo.

AE: Algoritmos Evolutivos.

PE: Programación Evolutiva.

EE: Estrategias de Evolutivas.

AG: Algoritmos Genéticos.

PG: Programación Genética.

GEP: Programación de Expresiones Genéticas.

ET: Árboles de expresión (por sus siglas en inglés Expression Trees).

MDL: Principio de Longitud de Descripción Mínima: Minimum Description Length.

Datos: del latín datum (“lo que se da”), un dato es un documento, una información o un testimonio que permite llegar al conocimiento de algo o deducir las consecuencias legítimas de un hecho.

ROC: Característica de funcionamiento del receptor: Receiver Operating Characteristic.

AUC: Área bajo la curva ROC: Area Under the Curve.

Dataset: juego de datos.

Framework: Marco de trabajo.

GNU: es un sistema operativo de tipo Unix desarrollado por y para el Proyecto GNU. GNU es un acrónimo recursivo de "GNU's Not Unix!" (En español: GNU no es Unix).

GB: Gigabyte, es una unidad de almacenamiento.

RAM: Memoria de Acceso Aleatorio: Random Access Memory.

UML: Lenguaje Unificado de Modelado (por sus siglas en inglés, Unified Modeling Language).

VPMN: Notación de Modelado de Procesos del Negocio (por sus siglas en inglés, *Business Process Modeling Notation*).

URL: Localizador Uniforme de Recursos.

Especificabilidad: cualidad de ser determinable.

Metaheurísticas: Una metaheurística es un método heurístico para resolver un tipo de problema computacional general, usando los parámetros dados por el usuario sobre unos procedimientos genéricos y abstractos de una manera que se espera eficiente.

Estocásticos: Proceso estadístico en el que la ley de probabilidad que da la evolución de un sistema depende del tiempo.

Plugin: Programa que puede anexarse a otro para aumentar sus funcionalidades (generalmente sin afectar otras funciones ni afectar la aplicación principal). No se trata de un parche ni de una actualización, es un módulo aparte que se incluye opcionalmente en una aplicación.

Soft Computing: es una rama de la Inteligencia Artificial que engloba diversas técnicas empleadas para solucionar problemas que manejan información incompleta, con incertidumbre y/o inexacta.

Anexos

Anexo 1: Diagrama de estados del nuevo clasificador.

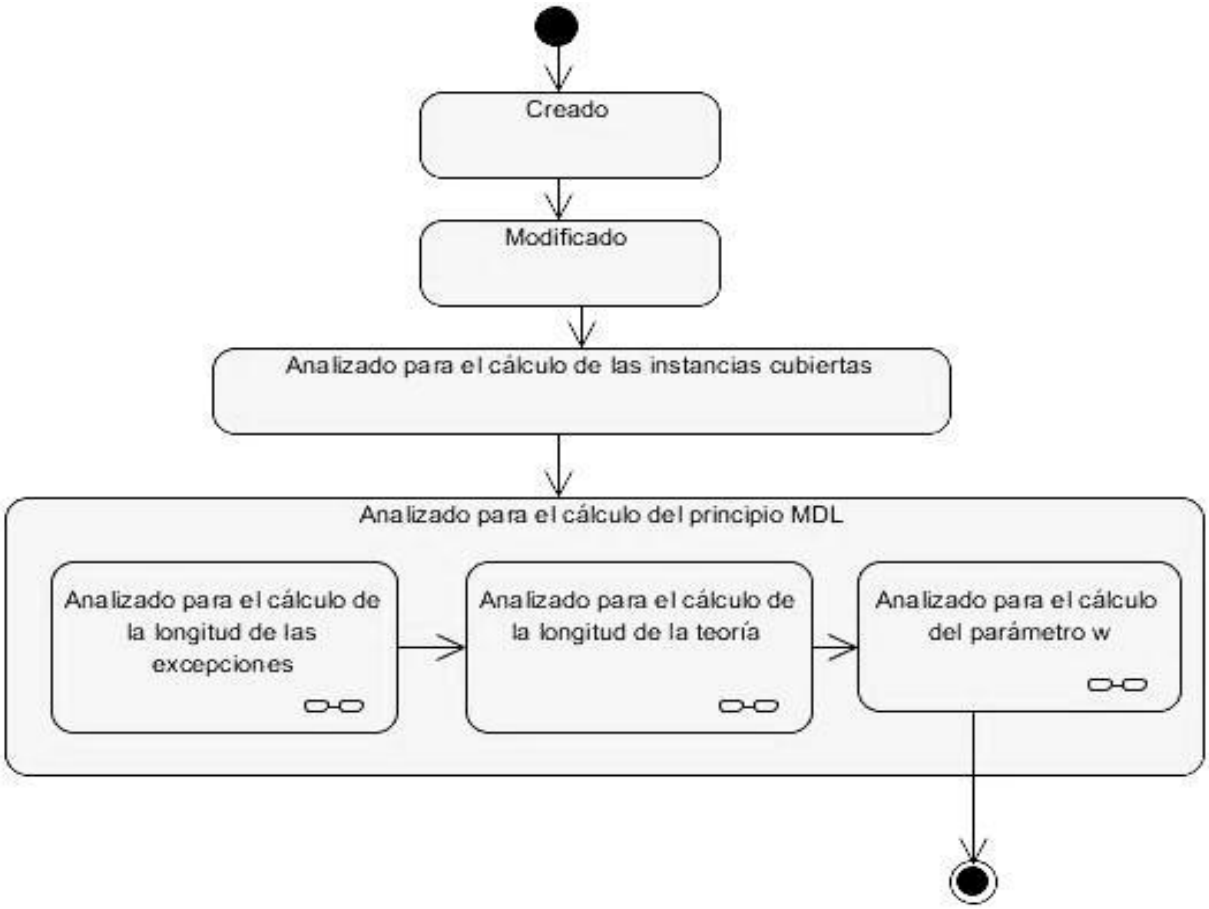

Figura 16: Diagrama de estados del nuevo clasificador (Por elaboración propia).

Anexo 2: Diagrama de proceso calcular el principio MDL.

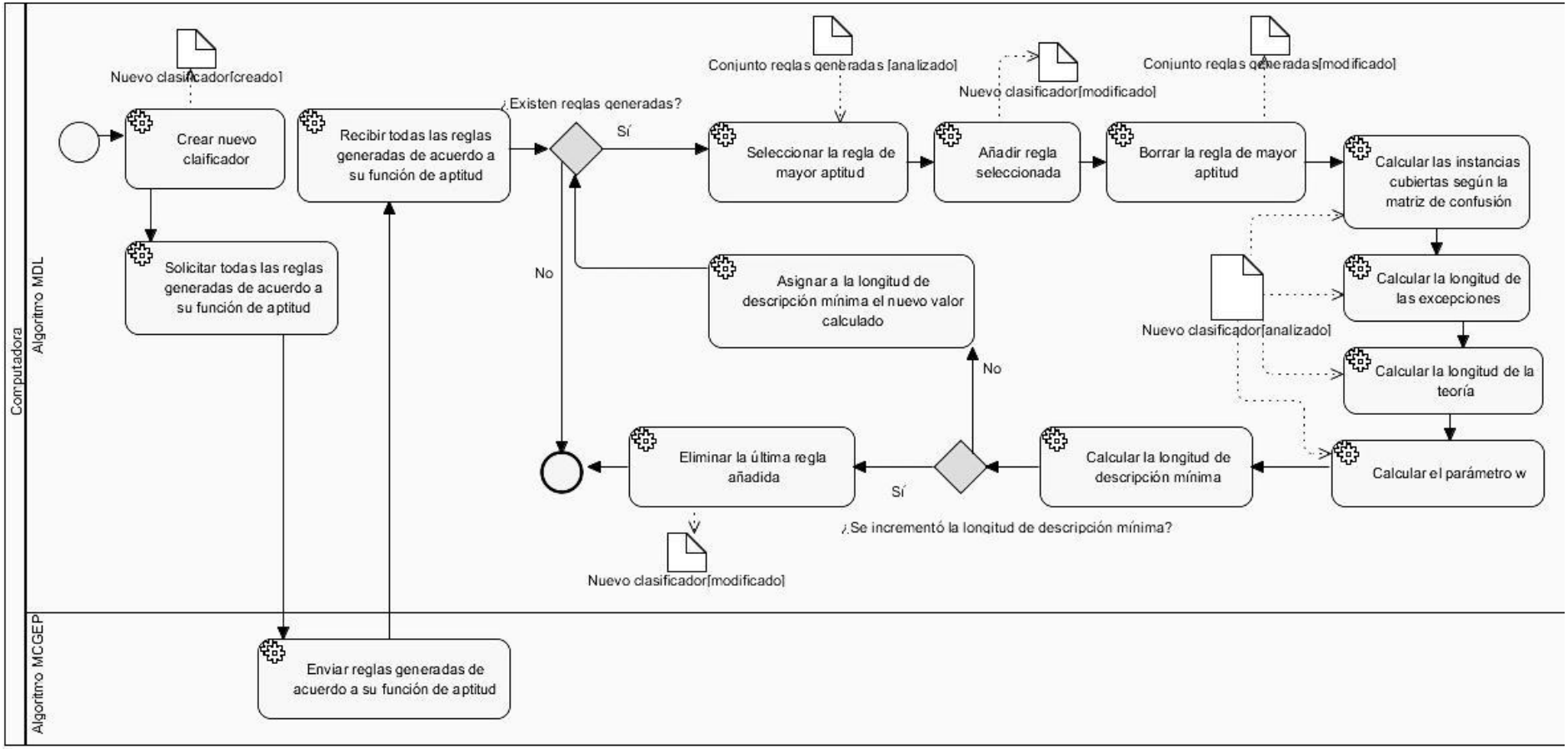

Figura 17: Diagrama de proceso: Calcular el principio MDL (Por elaboración propia).