Árboles de decisión utilizando recocido simulado

Ángel García Baez[1], Guillermo Alfredo Gacía Manjarres[1], and Eva Jiménez Flora[1]

Maestría en Inteligencia Artificial, Universidad Veracruzana, México

Resumen. Este artículo presenta un análisis comparativo del comportamiento del árbol de decisión creado utilizando el algoritmo C 4.5 y uno creado utilizando recocido simulado donde se propone un método de generación de soluciones basado en el intercambio de los nodos del árbol. Los resultados obtenidos sobre un conjunto de datos de prueba, indican que si bien los árboles de decisión producidos por este algoritmo para datasets sencillos son buenos, los resultados con datasets más complejos son subóptimos.

Abstract. This aricle presents a comparative analysis of the behaviours of a decision tree developed using the C 4.5 algorithm and simulated annealing, proposing a method for the generation of solutions based on the interchange of the tree nodes. The results obtained over 3 datasets indicate that even though the decision trees generated with the simulated annealing algorithm for simple datasets are good, the results for more complex datasets are suboptimal.

Keywords: árboles de decisión · C4.5 · metahurísticas · optimización · recocido simulado.

1 Introducción

Hoy en día, con la creciente necesidad del análisis de datos en las diversas disciplinas como el aprendizaje automático, ciencia de datos, big data, entre otras, es necesaria la búsqueda de mejorar los métodos tradicionales con el fin de obtener nuevas soluciones o técnicas más efectivas.

El aprendizaje automático es una disciplina de la Inteligencia Artificial cuyo objetivo principal es que un agente artificial mejore su rendimiento a partir de resultados previamente obtenidos [6]. Encontrar un clasificador adecuado es una tarea desafiante, ya que no existe una solución universal en la minería de datos o el aprendizaje automático. Existen varios enfoques de clasificación, como lo son: reglas, redes neuronales, árboles de decisión, por mencionar algunos [10]. Estos a su vez se pueden clasificar en dos grandes tipos: aprendizaje supervisado y no supervisado [2].

Los árboles de decisión es un método de aprendizaje supervisado, los cuales son generados a partir de datos que cuentan con una estructura jerárquica similar a la de un árbol biológico, compuesto por ramas y hojas [12]. El cual utiliza datos

de entrenamiento ya clasificados, analiza los datos de entrenamiento y produce un clasificador junto con una regla de inferencia se puede usar para clasificar nuevos datos no vistos anteriormente [10]. La mayoría de los algoritmos de inducción inducen divisiones de forma recursiva mediante la optimización local de una función de división predefinida [8].

Los árboles de decisión son extremadamente populares debido a su representación gráfica, eficiencia (pueden manejar el ruido en los datos, datos con valores atípicos, o irrelevantes, y aun así, hacer predicciones correctas) [4], rendimiento subóptimo y capacidad para explicar fácilmente sus aspectos internos a cualquier observador [10]. En la actualidad tiene un gran campo de aplicaciones, como lo son en el diagnóstico de enfermedades médicas, resolución de problemas eléctricos, reconocimiento de imágenes, procesamiento de señales, etc. [4].

Las técnicas tradicionales para inducir árboles de decisión están basados en una estrategia agresiva de posicionamiento recursivo para el crecimiento del árbol. Usan diferentes variantes de medida de impureza como ganancia de información, índice de Gini, entre otros, para seleccionar el atributo de entrada que será asociado a un nodo [12].

Entre sus desventajas, se puede mencionar que guía a soluciones subóptimas, debido a que al posicionamiento del conjunto de datos de forma recursiva, puede resultar en subconjuntos muy pequeños, lo cual puede llevar a un sobre ajuste de datos. Estas técnicas realizan una búsqueda local en cada nodo. Para evitar que el árbol de decisión se ajuste en exceso, se ha recorrido imponiendo algunas restricciones a los árboles o validando los árboles con un conjunto de datos de validación independiente [6].

Por lo cual, han surgido propuestas de modelos híbridos, en los cuales se incluyen métodos estocásticos con el objetivo de resolver problemas de optimización global [9]. Algunos de los más conocidos son: mínimos cuadrados; algoritmo genético; recocido simulado, búsqueda tabú, por mencionar algunos [9] [14].

Recientemente, los métodos evolutivos, especialmente la Programación Genética, se han utilizado para evolucionar árboles de decisión [14], [3], [5]. Los algoritmos evolutivos se inspiran en el proceso de la evolución natural. Esta metáfora de evolución natural relaciona al cómputo evolutivo con un estilo muy particular de resolución de problemas, donde existe una colección de posibles soluciones y con las técnicas correctas estas pueden ser seleccionadas para competir para sobrevivir y reproducirse, tal como los seres vivos, usando la técnica de prueba y error [13].

En este caso, los árboles de decisión evolucionan mediante operaciones explícitas de árboles de decisión candidatos [5]. Donde los algoritmos evolutivos realizan una búsqueda robusta global, es decir, buscan un equilibrio tanto en la exploración, las como en la explotación en el espacio de búsqueda de las soluciones candidatas [6]. Generando así árboles subóptimos no codiciosos. Los algoritmos de inducción genética codifican los árboles en representaciones predefinidas, que a su vez se optimizan [2]. Al mismo tiempo que busca construir árboles de decisión cortos y casi óptimos.

Otra metaheurística que se ha empleado para evitar el sobre ajuste de los árboles de decisión es el recocido simulado, el cual permite solucionar problemas de optimización global. Debido a su naturaleza estocástica, este método obtiene buenas soluciones incluso si existen múltiples óptimos locales. Su principal desventaja es que el tiempo de búsqueda puede tornarse infinito [11].

El recocido simulado es un método probabilístico propuesto en Kirkpatrick, Gelett y Vecchi (1983) y Cerny (1985) para encontrar el mínimo global de una función de costo que puede poseer varios mínimos locales. Funciona emulando el proceso físico por el cual un sólido se enfría lentamente de modo que cuando finalmente su estructura se "congela", esto sucede con una configuración de energía mínima [1]. El recocido simulado es especialmente efectivo para resolver problemas de gran escala (multidimensionales) en los que el óptimo global se encuentra escondido detrás de muchos óptimos locales [7].

La idea central del recocido simulado es imitar al proceso de recocido físico. En este último, un metal es calentado hasta una temperatura mayor a la de su punto de fusión para inmediatamente después ser gradualmente enfriado hasta llegar a una temperatura muy baja (único estado físico permitido). Si el proceso de enfriamiento es lo suficientemente lento, los átomos del metal formarán una estructura libre de defectos o estructura estable. En contrapartida, si el proceso de enfriamiento se realiza con rapidez, los átomos del metal formarán una estructura defectuosa, llena de irregularidades e imperfecciones, o estructura metaestable [14].

A continuación, se encuentra el desarrollo de la implementación de un árbol de decisión utilizando recocido simulado, con el objetivo de proponer una alternativa de mejora en el desempeño y precisión de los de los árboles tradicionales en este caso el algoritmo C4.5, para lo cual, se utilizó 3 dataset obtenidos de "UC Irvine Machine Learning Repository" con el fin de comparar el desempeño de los mismos.

2 Desarrollo

El diseño de árboles de decisión, utilizando un algoritmo de recocido simulado, provienen del principio de enfriamiento de materiales, donde dependiendo en un nivel de energía alto, su estructura puede cambiar hasta que en un nivel de energía mínimo su estructura se congela.

Para el algoritmo del recocido simulado se utiliza una variable llamada temperatura. El algoritmo comienza con un valor alto para la temperatura y se crea una solución inicial aleatoria. A partir de la solución inicial se crean nuevas soluciones y mientras la temperatura sea alta, la probabilidad de aceptar soluciones peores es mayor. Cada vez que se acepta una solución peor, la temperatura disminuye hasta que esta llega a 0, se realiza un número máximo de iteraciones o se encuentra una solución adecuada.

Para determinar la viabilidad de una solución se utiliza una heurística, en este caso la precisión de la clasificación sobre un conjunto de datos de entrenamiento.

Para la solución inicial se escogen los atributos para cada nodo del árbol de forma aleatoria utilizando división de una sola variable, por lo cual los atributos de los datos solo se utilizan una vez en todo el árbol.

Para la creación de nuevas soluciones se intercambian nodos del árbol que estén en diferentes ramas. Esto permite buscar diferentes configuraciones sin cambiar el tamaño del árbol, que es directamente proporcional al número de atributos de los datos y el número de valores que puede tomar cada atributo. Para dar variabilidad al nodo raíz hay una pequeña posibilidad de recrear el árbol por completo.

De igual forma que la probabilidad de aceptar una solución peor aumenta con la temperatura, también la cantidad de modificaciones realizadas al árbol es proporcional a la temperatura, mientras mayor temperatura la probabilidad de realizar un mayor número de modificaciones por iteración es mayor.

A continuación, se muestra el pseudocódigo del algoritmo implementado.

Algorithm 1: Algoritmo del árbol propuesto

Result: Mejor árbol encontrado con su precisión final.
Data: Datos de entrada: atributos, clases y numero maximo de iteraciones.
Input: Datos de entrada (atributos-clase)
Output: Mejor árbol de decisión y su precisión final

Paso 1: Inicializar la temperatura con un valor alto.;
Inicializar el árbol de decisión con los datos de entrada (atributos-clase).;
Crear primera solución de forma aleatoria utilizando cada atributo solo una vez.;
Evaluar el rendimiento del árbol.;
Paso 2: Generar un número aleatorio de modificaciones al árbol basado en la temperatura.;
Paso 3: Evaluar el rendimiento del árbol con los datos de entrada.;
Si el rendimiento del árbol es mayor al árbol previo, ir a paso 5, si no ir a paso 4.;
Paso 4: Se puede aceptar el árbol con una probabilidad proporcional a la temperatura, si es el caso, reducir la temperatura.;
Ir a paso 6.;
Paso 5: Aceptar el nuevo árbol.;
Paso 6: Si la temperatura es igual o menor a 0 o si el número de iteraciones es mayor al número máximo de iteraciones ir al paso 7, si no ir al paso 2.;
Paso 8: Mostrar el árbol final y su precisión.;

3 Experimentos y resultados

Para realizar la comparación se utilizaron 3 datasets obtenidos de "UC Irvine Machine Learning Repository", los cuales se muestran sus características principales en la tabla 1. Cabe mencionar que estas presentan información heterogénea (Al incluir en sus atributos numéricos y nominales). Previamente las dataset fueron sometidas a preprocesamiento con el algoritmo de CAIM.

Table 1. Descripción de los datasets usados en el estudio experimental.

Dataset	Instancias	Atributos	Clases
Iris	150	4	3
Zoo	101	17	7
tic-tac-toe	958	11	2

En la tabla 2 y 3 se muestran las características de los árboles de decisión obtenidos en los experimentos realizados con los algoritmos de C4.5 y recocido simulado. Los resultados se obtuvieron a partir de aplicar cross-validation (K=5) a los algoritmos y son los promedios de los árboles generados. Entre comillas se muestra la desviación estándar de igual manera.

Las variantes de los árboles C4.5 se implementaron en en el lenguaje de R, utilizando las bibliotecas Rweka y JMetal, y e la variante de recocido simulado fue implementado en lenguaje de python utilizandpo la biblioteca de pandas.

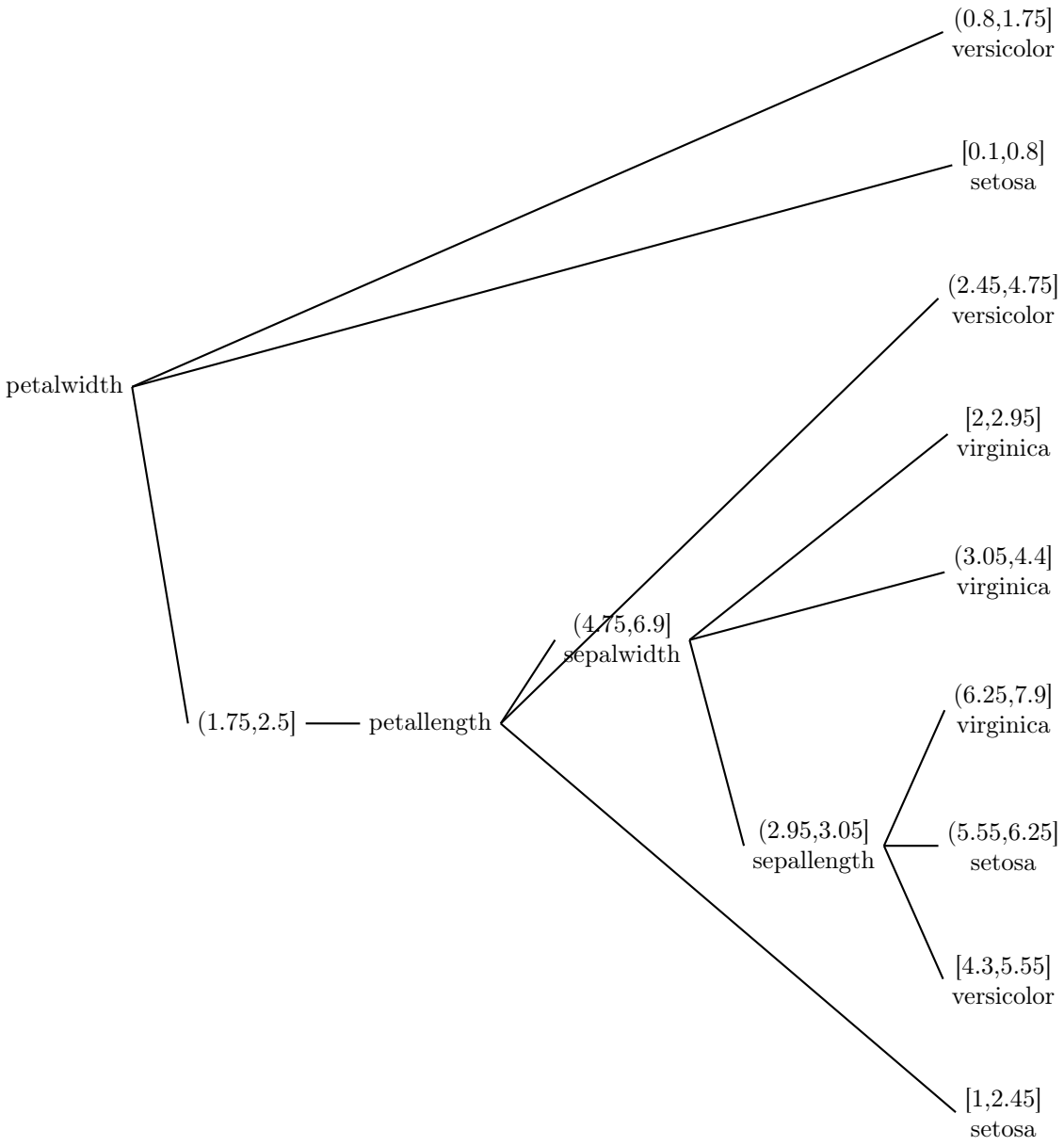
Table 2. Resultados del método modificado C4.5.

Dataset	Precisión	Tiempo	Tamaño promedio	Hojas promedio
Iris	93.3 (4.4)	0.22	5.2 (3.8)	3.8 (1.68)
Tic-tac-toe	84.5 (2.37)	0.28	133.3 (8.77)	89.2 (5.84)
Zoo	93.3 (8.7)	0.67	18 (3.91)	10.7 (2.98)

Table 3. Resultados del método modificado de recocido simulado.

Dataset	Precisión	Tiempo	Tamaño promedio	Hojas promedio
Iris	90.6 (3.6)	5.21	13 (0)	9 (0)
Tic-tac-toe	67.1 (4.5)	13.61	28 (0)	19 (0)
Zoo	59.0 (8.2)	23.76	110.2 (59.926)	96.4 (52.77)

Arbol de decision obtenido con el metodo de recocido simulado para iris.



4 Conclusiones y trabajo futuro

Si bien los árboles de decisión obtenidos con el método de recocido simulado son adecuados para conjuntos de datos sencillos como iris, con conjuntos de datos más complejos como zoo se generan árboles subóptimos, esto aunado al costo computacional requerido para generar estos árboles lo hace un algoritmo no viable en su estado actual.

Dicho esto, este algoritmo usualmente genera árboles más compactos que C 4.5 lo cual es una ventaja, así como utilizar relativamente poca memoria (112 bytes para dos árboles). Como trabajo futuro queda la implementación de un método más robusto para la selección de atributos en la generación inicial, así como un sistema más completo para generar nuevas soluciones que permitan hacer una búsqueda más completa del conjunto de soluciones como ser capaz de cambiar nodos de atributos por nodos de clases y viceversa. También el realizar recortes al árbol cuando las ramas sean redundantes.

5 Referencias

References

[1] Dimitris Bertsimas and John Tsitsiklis. "Simulated Annealing". In: *Statistical Science* 8.1 (Feb. 1993). Report from the Committee on Applied and Theoretical Statistics of the National Research Council on Probability and Algorithms, pp. 10–15.

[2] M. Binkhonain and L. Zhao. "A Review of Machine Learning Algorithms for Identification and Classification of Non-Functional Requirements". In: *Expert Systems with Applications* X (2019), p. 100001.

[3] Martijn CJ Bot and William B Langdon. *Application of genetic programming to induction of linear classification trees*. Springer, 2000.

[4] Sung-Hyuk Cha and Charles Tapperts. "Constructing Binary Decision Trees Using Genetic Algorithms". In: *Proceedings of the 2008 International Conference on Genetic and Evolutionary Methods (GEM 2008)*. Las Vegas, Nevada, USA, July 2008, N/A.

[5] Jeroen Eggermont, Joost N. Kok, and Walter A. Kosters. "Genetic Programming for Data Classification: Partitioning the Search Space". In: (2019). N/A.

[6] Camilo Flores Rodríguez. "*Generación de árboles de decisión usando un algoritmo inspirado en la física*". MA thesis. Centro de Investigación en Inteligencia Artificial, Universidad Veracruzana, 2021.

[7] D.E. Golberg. *Genetic Algorithms in Search, Optimization, & Machine Learning*. 1953.

[8] Riccardo Guidotti et al. "Generative Model for Decision Trees". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 19. 2024, pp. 21116–21124. DOI: 10.1609/aaai.v38i19.30104.

[9] M.A. Hafeez et al. "Performance Improvement of Decision Tree: A Robust Classifier Using Tabu Search Algorithm". In: *Applied Sciences* 11 (2021), p. 6728. URL: https://doi.org/10.3390/app11156728.

[10] Cezary Z. Janikow. "Applications of Genetic Programming to Building Decision Trees". In: *University of Missouri-St. Louis* (2011). URL: https://www.umsl.edu/cmpsci/faculty-sites/Janikow/publications/2011/Applications%20of%20Genetic%20Programming%20to%20Building%20Decision%20Trees/paper.pdf.

[11] J.-A. Mejía-de Dios. "A Metaheuristic Based on the Center of Mass". MA thesis. Centro de Investigación en Inteligencia Artificial, Universidad Veracruzana, 2018.

[12] Miguel Ángel Morales-Hernández, Rafael Rivera-López, and Efrén Mezura-Montes. "Inducción de árboles de decisión oblicuos utilizando dos variantes de evolución diferencial adaptiva". In: *Research in Computing Science* 152.6 (2023). Recibido el 17 de abril de 2023; aceptado el 1 de mayo de 2023, pp. 287–298.

[13] C. Tsallis and D.A. Stariolo. "Generalized Simulated Annealing". In: *Physica A: Statistical Mechanics and its Applications* 233.1-2 (1996), pp. 395–406. DOI: 10.1016/s0378-4371(96)00271-3.

[14] Jorge Homero Wilches-Visbal and Alessandro Martins Da Costa. "Algoritmo de recocido simulado generalizado para Matlab". In: *Ingeniería y Ciencia* 15.30 (2019), pp. 117–140.