Árboles de decisión

Isaí García Mendoza

1. Introducción

Los árboles de decisión son una colección de algoritmos útiles para encontrar patrones en conjuntos de datos (de ahora en adelante DS por sus siglas en inglés **Data Set**) que en primera instancia son difíciles de caracterizar. Usados para tareas de clasificación y de regresión, los árboles de decisión (DT, **Decision Tree**) ofrecen muchas ventajas respecto de algoritmos más básicos como regresión logística y regresión lineal, como tratar problemas que estos dos últimos no pueden resolver: la no linealidad presente en el DS.
La idea fundamental detrás de DT es particionar el DS de entrenamiento en subconjuntos más pequeños dando lugar a una estructura de tipo árbol, de ahí su nombre, donde todas las posible rutas hacia los nodos hojas (nodos que no particionan más el DS) se pueden ver como reglas ***if-else***.
Por ejemplo, dado el siguiente DS:

Pronóstico	**Temperatura**	**Humedad**	**Viento**	**Jugar tenis**
Soleado	Caliente	Alta	Débil	No
Soleado	Caliente	Alta	Fuerte	No
Nublado	Caliente	Alta	Débil	Sí
⋮				
Nublado	Templado	Alta	Fuerte	Sí
Nublado	Caliente	Normal	Débil	Sí
Lluvia	Templado	Alta	Fuerte	No

Cuadro 1: Condiciones climáticas y la decisión de jugar tenis.

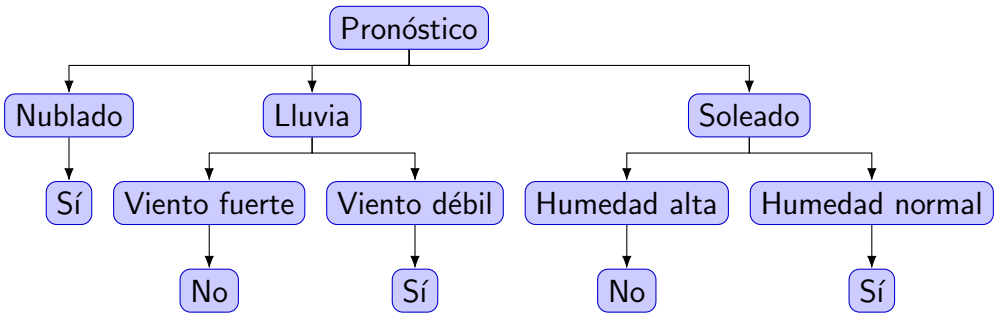

Figura 1: Árbol resultante

Se puede ejecutar una variante de DT, que se revisará más adelante, obteniendo el árbol de la figura 1, el cual clasifica correctamente todo el DS. El nodo raíz, etiquetado con

el atributo pronóstico, es el que mejor divide el DS original. Este, a su vez, tiene tres hijos que dividen todavía más el DS, esto sigue sucediendo hasta que se cumplen ciertas condiciones de parada que impiden que el árbol siga creciendo indefinidamente. A primera vista se pueden inferir algunas reglas:

- Si el pronóstico es nublado, entonces sí se puede jugar tenis.
- Si el pronóstico es lluvia y hay viento fuerte, entonces no se puede jugar tenis.
- Si el pronóstico es soleado y la humedad normal, entonces sí se puede jugar tenis.

Esta es una posible aplicación de los DT: generar reglas que puedan usarse en la toma de decisiones. Esto es un ejemplo de extracción de conocimiento (**Knowledge Discovery**), algoritmos utilizados en minería de datos para analizar conjuntos de datos tanto no estructurados como estructurados.
En este ensayo se repasarán los distintos algoritmos y métricas comunes a la hora de construir un árbol de decisión tomando como referencia Maimon y Rokach [1].

2. Desarrollo

2.1. Terminología

Antes de ahondar más en detalle sobre DT, es necesario aclarar la terminología que se usará a lo largo del texto.

- Muestra: una fila del DS.
- Atributo: una columna del DS, puede ser numérica o categórica.
- Nodo: puede ser de decisión (divide el DS) o un nodo hoja (no tiene más hijos, tiene asociado el valor que se predice).
- Clase: la variable que se desea predecir a partir de los demás atributos.
- Modelo: algoritmo matemático que ya ha sido entrenado y puede realizar nuevas predicciones.

2.2. Métricas

Centrándose en la tarea de emplear DT para clasificar el mismo DS presentado al principio, se debe encontrar el mejor atributo que lo divide, para lo cual se pueden usar distintas métricas. Una de las más sencillas de entender e implementar es la ganancia de información (IG, **Information Gain**) que combina la entropía del nodo padre y la entropía de los nodos hijos (resultados de dividir el nodo padre de acuerdo con el atributo en cuestión y con todos los valores posibles que puede tomar este). La entropía es una medida de impureza que cuantifica el desorden en un DS, así un DS homogéneo tendrá un entropía menor que uno heterogéneo. Se calcula con la siguiente fórmula:

$$E(S) = \sum_{i=1}^{C} -p_i \log_2 p_i \tag{1}$$

Donde S es el DS en cuestión, C es la cantidad clases únicas, p_i es la proporción entre el número de muestras que pertenecen a la clase i-ésima y el tamaño total de S. p_i se puede considerar como la probabilidad de que una muestra dada pertenezca a la clase i-ésima. La entropía será nula si todas las muestras pertenecen a la misma clase (conjunto perfectamente homogéneo) y será máxima si todas las clases tienen la misma probabilidad de ocurrir.

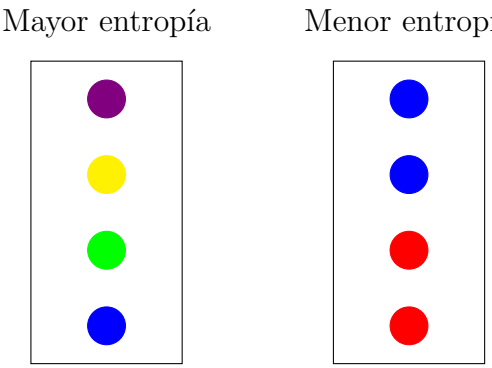

Figura 2: Entropía como medida de desorden

La ganancia de información se calcula con:

$$IG = E(S) - \sum_{i=1}^{k} \frac{n_i}{n} E(i) \tag{2}$$

Donde k es la cantidad de valores únicos que toma el atributo del que se está calculando su IG. En el caso del atributo pronóstico, que es el primero que escoge el algoritmo, este número sería 3 (soleado, nublado y lluvia). n_i es el tamaño del subconjunto que resulta de filtrar todas la muestras de S que tienen presente el valor k-ésimo del atributo. Por ejemplo, si se observa cuáles muestras tienen el valor soleado, se forma un subconjunto de 5 elementos. A este proceso de filtrado también se le llama dividir el nodo o particionarlo. En la ecuación 2 se usa 1, pero también es posible usar **Gini Index** (GI), se calcula con:

$$GI = 1 - \sum_{i=1}^{C} p_i^2 \tag{3}$$

Donde C y p_i tienen el mismo significado que en 1. GI cuantifica la frecuencia con que se clasificará mal una muestra al azar de S. Será nula cuando solo haya una clase y máxima cuando todas las clases tengan la misma probabilidad.
Para problemas de clasificación binaria se puede usar la métrica DKM, cuya fórmula es:

$$DKM(y, S) = 2 \cdot \sqrt{\left(\frac{|\sigma_{y=c_1} S|}{|S|}\right) \cdot \left(\frac{|\sigma_{y=c_2} S|}{|S|}\right)} \tag{4}$$

Otro métrica importante para determinar si un atributo está relacionada de forma significativa con la clase es **Chi-squared test**. Se calcula con:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{5}$$

Donde O_i es la frecuencia observada de la categoría i y E_i es la frecuencia esperada. Las métricas mencionadas hasta el momento tienen un problema y es que favorecen los atributos con muchos valores únicos, lo que resultará en una pobre generalización del modelo. Aquí es donde entran en juego las métricas normalizadas, que además ayudan a el resolver el problema de tratar atributos numéricos, como se presentará más adelante. Una de tales es el **Gain Ratio** (GR) que puede calcularse con:

$$GR = \frac{InformationGain(a_i, S)}{SplitInformation(a_i, S)} \tag{6}$$

Donde a_i es el atributo del cual se quiere calcular su GI y $SplitInformation$ cuantifica el desorden introducido por dividir S de acuerdo con los valores específicos de a_i, se calcula con:

$$SplitInformation(a_i, S) = -\sum_{i=1}^{k} \frac{|S_i|}{|S|} \log_2 \left(\frac{|S_i|}{|S|} \right) \tag{7}$$

Por lo tanto, GR penaliza los atributos con muchos valores, pues $SplitInformation$ tenderá a aumentar en tal caso.
En Maimon y Rokach [1] se mencionan varias métricas más como **Distance Measure**, **Binary Criteria**, **Orthogonal (ORT) Criterion**, **Kolmogorov–Smirnov Criterion**, **AUC–Splitting Criteria**. Sin embargo, se decidió omitirlas, pues no se encontró respaldo suficiente de que dichas métricas realmente se usen en la implementación de los algoritmos de creación de árboles de decisión.

2.3. Criterios de parada

Las métricas mencionadas en el apartado anterior son usadas para construir el DT hasta que se cumple algunas de las siguientes condiciones:

- Se ha llegado a un nodo donde todas las muestras pertenecen a la misma clase.
- Se ha llegado a la profundidad máxima (hiperpárametro) a la que se desea construir el árbol.
- El nodo ya no se puede dividir más, pues la cantidad de muestras que tiene es menor que un número fijado (hiperpárametro).
- La métrica que se usó para escoger la mejor división del nodo no supera un cierto umbral.

Una vez se ha construido el árbol, este puede tener muchas ramas que no aportan a su precisión y además lo sobreajustan, es decir, no es capaz de generalizar a muestras que no vio durante la etapa de construcción. En este punto se pueden emplear métodos de poda que van a eliminar tales ramas.

2.4. Métodos de poda

Existen dos tipos aproximaciones a este problema, una consiste en aplicar una pre poda ajustando los hiperpárametros que controlan cuándo el árbol dejará de crecer, para lo cual se pueden usar tecnicás de optimización de hiperpárametros como **Grid Search**. O bien dejando que el árbol crezca permitiendo el sobreajuste y después eliminar las ramas

que no sean significativas. Uno de los métodos más sencillos para esta última tarea es **Reduced Error Pruning** (REP), el cual procesará el árbol de abajo a arriba eliminando nodos de decisión y sustituyéndolos por nodo hojas donde la clase predicha es la clase más repetida en sus nodos hijos, esto se realiza siempre que este proceso no reduzca la precisión del modelo en el DS de validación. Este método conlleva inconvenientes como que tiene una alta complejidad computacional para DS grandes. Otro método bastante empleado en la práctica es **Cost Complexity Pruning**, el cual asocia a cada subárbol una medida de costo-complejidad de tal forma que los subárboles que la reduzcan se reemplazan por una hoja, este proceso genera una secuencia de árboles, aquí se puede usar validación cruzada para seleccionar el mejor. La fórmula en que se basa es:

$$R_\alpha(T) = R(T) + \alpha \cdot |T| \tag{8}$$

Donde $R(T)$ es el error del subárbol, $|T|$ la cantidad de nodos hoja que tiene y α es un parámetro que debe ajustarse. Este método tiene desventajas como que es díficil de implementar y conlleva una alta carga computacional.
Otro método importante es **Error Based Pruning**, el cual funciona de forma parecida a REP pero no depende de un DS de validación y además utiliza una medida de error que ayudará a determinar qué nodos simplificar. Como se menciona en Maimon y Rokach [1], por el teorema de *no free lunch*, no hay ningún método que sea mejor que los demás en todos los casos, y por tanto, el uso de uno u otro depende del problema que se esté tratando.
Hasta ahora se ha repasado las métricas que se usan para seleccionar los mejores atributos que dividen un nodo y algunas técnicas para evitar el sobreajuste, pero no se ha visto cuáles pasos hay que seguir para construir un DT desde el DS. Para ello existe lo que en la literatura se conoce como **Tree Inducers**. A continuación se presentarán algunos.

2.5. Algoritmos de generación de árboles

Uno de los algoritmos más accesibles de implementar es **ID3**, desarrollado por Ross Quinlan, el cual funciona únicamente para atributos categóricos, este es uno de sus inconvenientes. Usa la ganancia de información y la entropía para seleccionar el mejor atributo en cada iteración de forma *greedy*. Otro algoritmo del mismo autor, **C4.5**, es una mejora a ID3, pues acepta atributos numéricos, que emplea *gain ratio* y *error based pruning*. Estos dos algoritmos construyen árboles de clasificación, es decir, aquellos en que el atributo objetivo es categórico. Para poder predecir atributos numéricos se puede hacer uso del algoritmo **CART** que, a diferencia de los demás, construye un árbol binario. Otro algoritmo popular en la práctica es **CHAID**, el cual emplea *chi-square test* como método para elegir el mejor atributo, además considera los valores en faltantes en las muestras como una categoría aparte, pero no emplea ningún método de poda, por lo que es susceptible al sobreajuste.
No existe un mejor algoritmo para construir un DT, y el uso de uno u otro depende de la naturaleza específica del problema a resolver.

2.6. Ventajas y desventajas

Una de las principales ventajas de los árboles de decisión con respecto a otros algoritmos como las redes neuronales es que son fáciles de interpretar, pueden manejar DS con

valores faltantes o errores, de tipos numérico o categórico, y además es un modelo no paramétrico. Dentro de sus desventajas se encuentra que son muy sensibles a los cambios en los valores del DS, así como al ruido presente en este, implican alto coste computacional en DS grandes, y pueden generalizar mal a muestras no vistas.

3. Conclusión

Los árboles de decisión son algoritmos buenos para clasificación y regresión, y aunque no son perfectos, son la base de algoritmos más potentes como **Random Forest**, que pueden aumentar sus capacidades. Por lo tanto, constituyen una buena herramienta a poseer para todo científico de datos.

4. Referencias

[1] Oded Maimon y Lior Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer, 2010.