Introducción a Weka

García Prado, Sergio

27 de febrero de 2017

Resumen

En este documento se ha analizado el comportamiento de los algoritmos ID3 *y* J48 *a través de su implementación en la herramienta de minería de datos* Weka*[4]. Como caso de prueba se ha entrenado a partir de la tabla de verdad de una función lógica a los distintos algoritmos. Como resultados de dicho análisis se ha comprobado el sobreajuste producido por* ID3*, que en este caso proporciona mejores resultados que* J48 *debido al carácter finito de la función así como la estrategia de entrenamiento y test seguida para la realización de la comparativa.*

1. Tómese la siguiente función lógica y obténgase el árbol de decisión correspondiente usando WEKA[4]

$$\neg(A \wedge B) \vee \neg(C \wedge D) \oplus E \tag{1}$$

En esta práctica se ha generado la tabla de verdad de la función descrita en la ecuación (1). Por lo tanto, el conjunto de datos que se ha utilizado en este caso sigue la siguiente estructura: *a*) los atributos de cada dato se corresponden con una determinada combinación de las variables de entrada de (1), *b*) mientras que el valor de la clase será el valor de verdad de dicha combinación.

El conjunto de datos está compuesto por tanto, por 5 atributos de tipo booleano (discretos) al igual que la clase, que también es de tipo booleano. Puesto que la función está formada por 5 argumentos y cada uno de ellos puede tomar 2 valores, nuestro conjunto de datos está formado por $2^5 = 32$ datos. Todos ellos han sido recogidos en la tabla 3. Dicha tabla de verdad ha sido generada a partir de la clase *LogicDataSet* codificada en el lenguaje *Python*, a la cual se puede acceder a partir de `https://github.com/garciparedes/python-examples/blob/master/data_sets/logic_data_set.py`[2].

El propósito de esta práctica es analizar el comportamiento de las estrategias de aprendizaje automático básadas en estructuras jerárquicas (árboles). Para ello se ha utilizado la herramienta *Weka*[4], que permite la realización de diversas tareas relacionadas con el aprendizaje automático de manera simple y de manera gráfica sobre conjuntos de datos. En este caso se ha realizado una comparativa entre los algoritmos de clasificación basada en árboles mediante aprendizaje supervisado.

Debido a la naturaleza intrínseca de un estructura en forma de árbol, esto permiten la representación de funciones lógicas con un alto grado de acierto. Para comprobar dicha cualidad, se ha realizado un ánalisis de resultados de los algoritmos *ID3* y *J48*, cuyas diferencias son las siguientes:

- **ID3**: Es el algoritmo básico de generación de árboles de decisión a partir de un conjunto de datos formado tanto por atributos como valores de clase de carácter discreto. Utiliza heurísticas basadas en la Teoría de Información. Debido a sus simplicidad, está muy condicionado al conjunto de datos de entrenamiento.

- **J48**: Es la versión implementada en el lenguaje *Java* del algoritmo **C4.5**, una versión con numerosas mejoras respecto de *ID3*, tales como: *a*) la capacidad de procesar atributos continuos mediante la generación, *b*) manejo de valores desconocidos, *c*) tratamiento de mismos atributos de entrada para distintos valores para la clase de destino y *d*) poda para tratar de evitar el sobreajuste.

Para los casos de prueba, con los dos algoritmos se han utilizado los parámetros por defecto de la herramienta *Weka*[4]. Tras utilizar todo el conjunto de datos tanto para entrenamiento como para test, los árboles generados por el algoritmo **ID3** y **J48** se muestran en las figuras 1 y 2. Dichas figuras han sido generadas a partir de la herramienta *treetograph*[3].

1.1. Resultados obtenidos a partir de *ID3*

El algoritmo *ID3*, tal y como se ha dicho en el párrafo anterior, ha generado el árbol de la figura 1. Tal y como se puede apreciar, este ha tenido en cuenta todos los datos para generar el árbol, por lo cuál tiene la misma profundidad que el número de atributos del conjunto de datos. Esto es equivalente al número de variables que tiene la función lógica (1) que se ha querido representar.

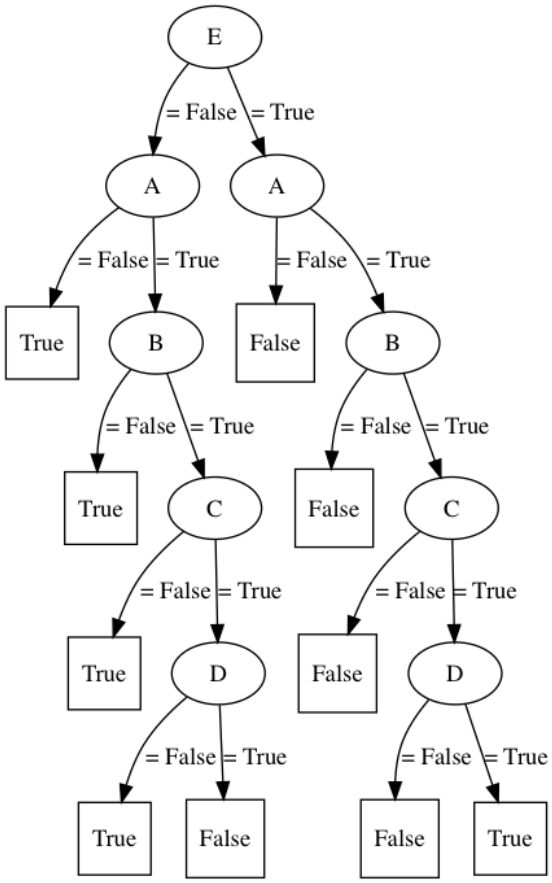

Figura 1: *Árbol de decisión generado a partir del algoritmo* ID3

Como consecuencia de dicho tamaño, con esta alternativa se consigue la matriz de confusión de la tabla 1. Como vemos, esta muestra que la tasa de acierto del árbol es del 100 %, ya que este captura todas las posibles combinaciónes de la función lógica (1). Los valores p_i y π_j representan

la proporción de aciertos de cada clase y la proporción de muestras de cada clase del conjunto de test respectivamente.

		Valor Real		
		Positivo	Negativo	p_j
Valor Predicho	Positivo	16	0	1
	Negativo	0	16	1
	π_j	0,5	0,5	$N = 32$

Tabla 1: *Matriz de confusión resultante de del conjunto de datos generado a partir de la ecuación* (1) *por el algortimo* ID3

Los resultados obtenidos han sido tan satisfactorios debido a los siguientes factores: *a*) la naturaleza lógica de la función utilizada para generar el conjunto de datos, *b*) la utilización de todo el conjunto de datos tanto para entrenamiento como para test y *c*) la estrategia utilizada por el algoritmo *ID3*, que no restringe el tamaño del árbol generado y captura todas las alternativas que obtiene a partir del conjunto de entrenamiento.

1.2. Resultados obtenidos a partir de *J48*

El algortimo *J48* se diferencia de su versión anterior en que proporciona un mayor número de características, que le hacen más versátil y eficiente a la hora de clasificar conjuntos de datos (a pesar de que en este ejemplo se haya producido el fenómeno contrario). En este caso, el algoritmo ha generado un árbol de decisión de altura 1, es decir, tan solo basará la clasificación en el valor de 1 único atributo del conjunto de datos, o lo que es lo mismo, 1 de las variables de la función lógica (1). Esto se ilustra gráficamente en la figura 2

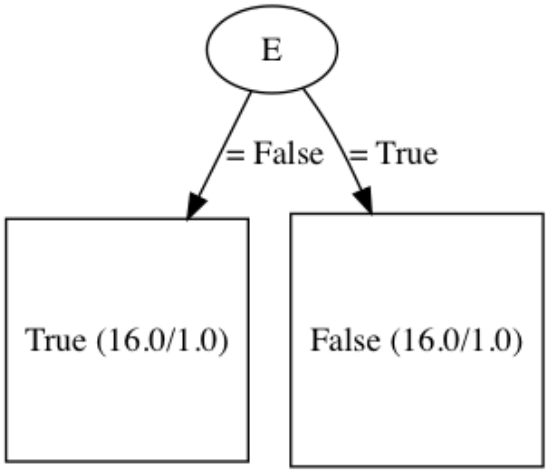

Figura 2: *Árbol de decisión generado a partir del algoritmo* J48

La razón por la cual se ha dado este suceso es debido a que el algoritmo *J48* realiza una poda al árbol que previamente genera siguiendo la misma estrategia que *ID3*. Esta poda consiste en eliminar nodos o reducir la altura del árbol basandose en un umbral admisible de pérdida de precisión a la hora de clasificar los datos. La razón por la cual utiliza dicha técnica es para tratar de evitar el sobre-ajuste.

El sobre-ajuste consiste en un fenómeno que consiste en la sobre adaptación del algoritmo de aprendizaje a los valores de entrenamiento. Lo que conlleva un peor comportamiento cuando se le presentan valores desconocidos (con los cuales no ha sido entrenado). El uso de esta técnica conlleva situaciones como lo ocurrido en este caso, es decir, una mayor tasa de fallo. Esto se puede comprobar a partir de la matriz de confusión de la tabla 2, obtenida tras volver a clasificar los mismos datos con los que previamente ha sido entrenado el algoritmo.

		Valor Real		
		Positivo	Negativo	p_j
Valor Predicho	Positivo	15	1	0,9375
	Negativo	1	15	0,9375
	π_j	0,5	0,5	$N = 32$

Tabla 2: *Matriz de confusión resultante de del conjunto de datos generado a partir de la ecuación* (1) *por el algortimo* J48

En la matriz de confusión de la tabla 2 se aprecia que el árbol generado no tiene un comportamiento tan "buenoçomo el anterior, produciendo una tasa de acierto de $p = 0{,}9375$ en ambos casos. Lo cual es debido a la poda del árbol de decisión.

1.3. Conclusiones

Mediante el ánalisis de resultados de los distintos algoritmos sobre el conjunto de datos generado por una función lógica se ha comprobado que la selección del mejor algoritmo para clasificarlo depende de muchos factores.

Por tanto, la selección de uno sobre otro muchas veces dependerá del problema que se pretenda resolver, así como el tamaño del conjunto de datos disponible para entrenamiento y la naturaleza del mismo. Además pueden intervenir otros factores como la tasa de acierto que se espera conseguir o las restricciones de carácter computacional que se pretendan imponer (lo cual requiere árboles de decisión de menor tamaño).

A	B	C	D	E	Result
False	False	False	False	False	True
True	False	False	False	False	True
False	True	False	False	False	True
True	True	False	False	False	True
False	False	True	False	False	True
True	False	True	False	False	True
False	True	True	False	False	True
True	True	True	False	False	True
False	False	False	True	False	True
True	False	False	True	False	True
False	True	False	True	False	True
True	True	False	True	False	True
False	False	True	True	False	True
True	False	True	True	False	True
False	True	True	True	False	True
True	True	True	True	False	False
False	False	False	False	True	False
True	False	False	False	True	False
False	True	False	False	True	False
True	True	False	False	True	False
False	False	True	False	True	False
True	False	True	False	True	False
False	True	True	False	True	False
True	True	True	False	True	False
False	False	False	True	True	False
True	False	False	True	True	False
False	True	False	True	True	False
True	True	False	True	True	False
False	False	True	True	True	False
True	False	True	True	True	False
False	True	True	True	True	False
True	True	True	True	True	True

Tabla 3: *Tabla de verdad de la ecuación 1*

Referencias

[1] Calonge Cano, T., and Alonso GonzáLez, C. J. Técnicas de Aprendizaje Autómatico, 2016/17.

[2] García Prado, S. Python Examples. `https://github.com/garciparedes/python-examples`.

[3] Taboada Rodero, I. J. treetograph. `https://github.com/ismtabo/treetograph`.

[4] The University of Waikato. Weka. `http://www.cs.waikato.ac.nz/ml/weka/`.