

Tema 11. Inducción de Reglas

Pedro Larrañaga, Iñaki Inza, Abdelmalik Moujahid
Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad del País Vasco–Euskal Herriko Unibertsitatea

11.1 Introducción

En este tema vamos a explicar las ideas generales que subyacen al algoritmo *RIPPER* (Cohen, 1995). Este algoritmo es una extensión del denominado *IREP* (Fürnkranz y Widner, 1994).

Tal y como hemos comentado en el tema anterior, las reglas como paradigma modelizador de un problema de clasificación supervisada son además de transparentes y fácilmente comprensibles y aplicables, un paradigma más genérico y flexible que los árboles de clasificación.

11.2 *IREP (Incremental Reduced Error Pruning)*

El conjunto de reglas que se van a considerar en el algoritmo *IREP* (Fürnkranz y Widner, 1994) van a estar en *forma normal disyuntiva*. Conceptos básicos utilizados por *IREP* son los de regla (*rule*), conjunto de reglas (*rule set*) y regla parcial (*partial rule*).

- Una regla (*rule*) en *IREP* se considera que está constituida por una conjunción de literales. Así por ejemplo la regla R_j :

$$R_j \equiv (X_7 = x_7^1) \& (X_{14} = x_{14}^2) \& (X_{24} = x_{24}^1)$$

está constituida por la intersección de los 3 literales siguientes:

$$(X_7 = x_7^1), (X_{14} = x_{14}^2) \text{ y } (X_{24} = x_{24}^1)$$

- Un conjunto de reglas (*rule set*) está formado por una disyunción de reglas. De ahí que podamos decir que *IREP* va a inducir un conjunto de reglas en forma normal disyuntiva. Un ejemplo de *rule set* puede ser el siguiente:

$$R_1 \text{ or } R_2 \text{ or } \dots \text{ or } R_k$$

donde cada R_j está constituido por la intersección de un conjunto de literales.

- Una regla parcial (*partial rule*), R_j^{par} de una determinada regla R_j está constituida por la intersección de un subconjunto de los literales a partir de los cuales se forma R_j .

Por ejemplo

$$R_j^{par} \equiv (X_7 = x_7^1) \& (X_{14} = x_{14}^2)$$

es una regla parcial de la regla R_j definida anteriormente.

Vamos a describir a continuación las ideas fundamentales en las que se basa *IREP*. Consideraremos el caso en el que la variable clase C tome dos posibles valores, los cuales se van a denotar como ejemplos positivos y ejemplos negativos respectivamente.

IREP necesita que el conjunto de casos o patrones etiquetados que denotamos por D se particione en dos. El primero de dichos subconjuntos D_{pos} va a contener el conjunto de patrones positivos, mientras que el segundo subconjunto D_{neg} va a contener el conjunto de patrones negativos. Además de esta partición es necesario subdividir cada uno de los subconjuntos anteriores (D_{pos} y D_{neg}) en otros dos subconjuntos. Es decir, a partir del conjunto de ejemplos positivos D_{pos} obtenemos $D_{grow-pos}$ y $D_{prune-pos}$ subconjuntos relacionados respectivamente con la construcción y el podado de las reglas. De manera análoga, a partir de D_{neg} obtenemos $D_{grow-neg}$ y $D_{prune-neg}$, los cuales también se relacionan respectivamente con la construcción y el podado de las reglas. En concreto se verifica que

$$D = D_{pos} \cup D_{neg} = (D_{grow-pos} \cup D_{prune-pos}) \cup (D_{grow-neg} \cup D_{prune-neg})$$

y además

$$D_{pos} \cap D_{neg} = D_{grow-pos} \cap D_{prune-pos} = D_{grow-neg} \cap D_{prune-neg} = \emptyset$$

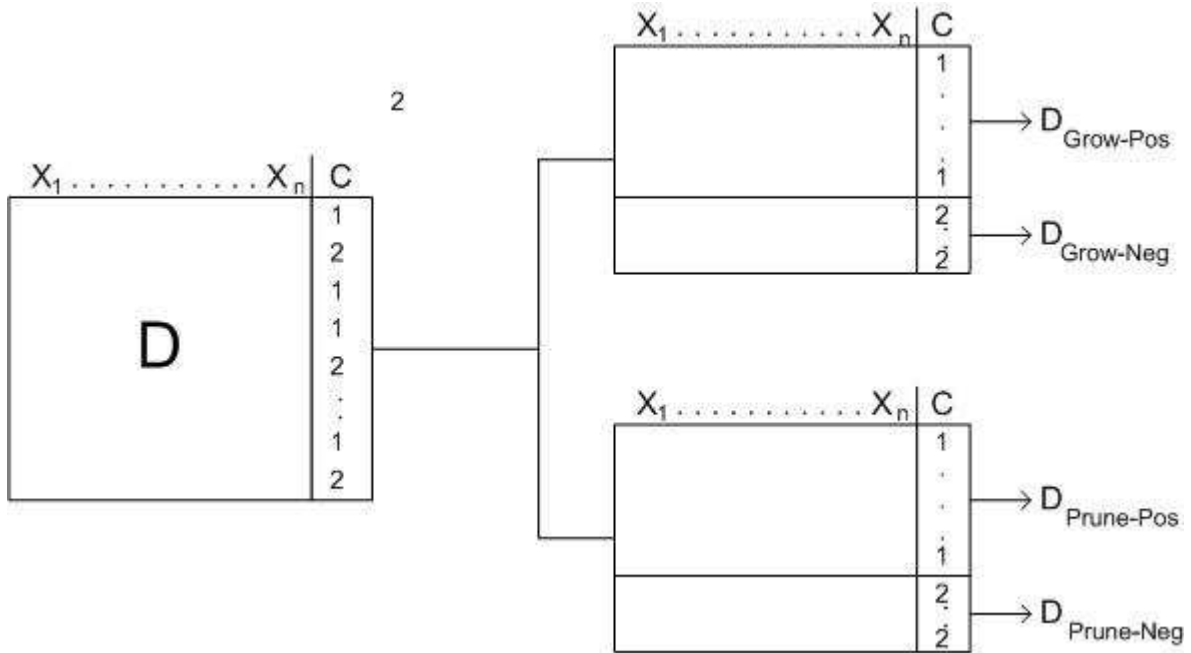


Figura 1: Partición del fichero de casos etiquetados de partida D

IREP va a inducir el conjunto de reglas (*rule set*) de manera voraz, escogiéndose en cada paso añadir el mejor literal a la regla en construcción. Para esta fase de construcción de reglas *IREP* se basa en $D_{grow-pos}$ y $D_{grow-neg}$. En concreto –véase Figura 2– *IREP* utiliza para ello el procedimiento *GrowRule*. Este procedimiento *GrowRule* añade de forma repetida a la regla parcial (*partial rule*) R^{par} el literal que da origen a la regla parcial R'^{par} con mayor valor del criterio siguiente:

$$v(R^{par}, R'^{par}, D_{grow-pos}, D_{grow-neg}) = cu \left[-\log_2 \left(\frac{pos}{pos + neg} \right) + \log_2 \left(\frac{pos'}{pos' + neg'} \right) \right]$$

siendo:

cu porcentaje de ejemplos positivos en $D_{grow-pos}$ que sienta cubiertos por R^{par} están también cubiertos por R'^{par} . Nótese que al ser R'^{par} más específica que R^{par} , no todos los ejemplos positivos cubiertos por R'^{par} lo van a estar por R^{par} .

pos número de ejemplos positivos cubiertos por R^{par} en $D_{grow-pos}$

neg número de ejemplos negativos cubiertos por R^{par} en $D_{grow-neg}$

pos' número de ejemplos positivos cubiertos por R'^{par} en $D_{grow-pos}$

neg' número de ejemplos negativos cubiertos por R'^{par} en $D_{grow-neg}$

Veamos cómo funcionaría el criterio anterior con un simple ejemplo. Supongamos que la regla parcial, R^{par} , que tenemos en un determinado momento cubre 90 ejemplos de los cuales 70 son positivos y 20 negativos. Vamos a tener en cuenta una especialización mala de dicha regla parcial, a la que denominaremos $R^{m,par}$ según la cual se van a cubrir 50 ejemplos positivos y 20 ejemplos negativos. Compararemos dicha mala especialización con una especialización buena, $R^{b,par}$, la cual cubre 70 ejemplos positivos y 10 ejemplos negativos.

Para cada una de las especializaciones anteriores tenemos:

$$v(R^{par}, R^{m,par}, D_{grow-pos}, D_{grow-neg}) = \frac{50}{70} \left[-\log_2 \left(\frac{70}{90} \right) + \log_2 \left(\frac{50}{70} \right) \right]$$

$$v(R^{par}, R^{b,par}, D_{grow-pos}, D_{grow-neg}) = \frac{70}{70} \left[-\log_2 \left(\frac{70}{90} \right) + \log_2 \left(\frac{70}{80} \right) \right]$$

Claramente el valor del criterio para $R^{b,par}$ es superior al correspondiente al $R^{m,par}$. Después de que haya finalizado el proceso de crecimiento de una regla (el cual para

Procedure *IREP*

Begin

Ruleset = \emptyset

while $D_{Pos} = D_{Grow-Pos} \cup D_{Prune-Pos} \neq \emptyset$ **do**

/* Construir y podar una nueva regla */

Dividir D en $D_{Grow-Pos} \cup D_{Grow-Neg} \cup D_{Prune-Pos} \cup D_{Prune-Neg}$

Rule := GrowRule($D_{Grow-Pos} \cup D_{Grow-Neg}$)

Rule := PruneRule(Rule, $D_{Prune-Pos}$, $D_{Prune-Neg}$)

if la tasa de error de Rule en $D_{Prune-Pos} \cup D_{Prune-Neg} > 50\%$

then

return RuleSet

else

Añadir Rule a RuleSet

Borrar ejemplos abiertos por Rule de D

endif

end while

return RuleSet

End

Figura 2: Pseudocódigo del algoritmo *IREP*

cuando no se encuentra ningún literal cuya inclusión en la regla permita que la regla

especializada mejore el criterio $v(R^{par}, R'^{par}, D_{grow-pos}, D_{grow-neg})$ se comienza con el proceso de podado de dicha regla. Dicho proceso de poda se efectúa por medio del procedimiento *PruneRule*. En este procedimiento *PruneRule* se plantea el borrado, de manera secuencial, y empezando por el último literal introducido a la regla en su fase de crecimiento. Se van a ir borrando (podando) literales mientras se mejore el criterio $v(Rule, D_{prune-pos}, D_{prune-neg})$, siendo

$$v(Rule, D_{prune-pos}, D_{prune-neg}) = pos \frac{Neg - neg}{Pos + neg}$$

con:

Pos número de ejemplos en $D_{prune-pos}$

Neg número de ejemplos en $D_{prune-neg}$

pos número de ejemplos positivos en $D_{prune-pos}$ cubiertos por la regla

neg número de ejemplos negativos en $D_{prune-neg}$ cubiertos por la regla

12.2 *RIPPER (Repeated Incremental Pruning Produce Error Reduction)*

RIPPER fue introducido por Cohen (1995) y constituye una mejora del algoritmo *IREP*. Las mejoras introducidas por *RIPPER* pueden resumirse en tres puntos:

1. Métrica alternativa para la fase de poda.

Supongamos que la regla R_1 cubre 2000 ejemplos positivos en $D_{prune-pos}$ y 1000 ejemplos negativos en $D_{prune-neg}$. Igualmente la regla R_2 cubre 1000 ejemplos positivos en $D_{prune-pos}$ y 1 ejemplo negativo en $D_{prune-neg}$. Se puede ver de manera sencilla que independientemente de los valores de Pos y Neg , el criterio de poda utilizado por *IREP* va a preferir R_1 a R_2 , ya que

$$\frac{2000 + (Neg - 1000)}{Pos + Neg} > \frac{1000 + (Neg - 1)}{Pos + Neg}$$

y sin embargo es intuitivo que la R_2 es preferible a R_1 . Para corregir esta carencia *RIPPER* basa su poda en el criterio siguiente:

$$v(Rule, D_{prune-pos}, D_{prune-neg}) = \frac{Pos - neg}{Pos + neg}$$

En tal caso tendríamos

$$\frac{2000 - 1000}{3000} < \frac{1000 - 1}{1001}$$

y R_2 se seleccionaría frente a R_1 .

2. Incorporación de un heurístico para determinar cuándo parar el proceso de añadir reglas.
3. Posteriormente a todo el proceso visto para *IREP*, el algoritmo *RIPPER* efectúa una búsqueda local para optimizar el conjunto de reglas (*rule set*) de dos maneras diferentes:

- a) Reemplazando una regla R_i que forma parte del *rule set* $\{ R_1, \dots, R_{i-1}, R_i, R_{i+1}, \dots, R_k \}$ por R'_i , siempre y cuando el *rule set* correspondiente tenga un menor error en la clasificación en $D_{prune-pos} \cup D_{prune-neg}$
- b) Revisar una determinada regla R_i añadiendo literales para que así se consiga un menor error en $D_{prune-pos} \cup D_{prune-neg}$

Referencias

1. W.W. Cohen (1995). Fast effective rule induction, *Proceedings of the Twelfth International Conference on Machine Learning*
2. J. Fürnkranz, G. Widner (1994). Incremental Reduced Error Pruning, *Proceedings of the Eleventh International Conference on Machine Learning*