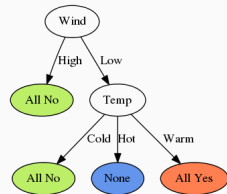


ÁRBOLES DE CLASIFICACIÓN

C4.5, C5.0 Y RANDOM FOREST



Luis María Costero Valero

Jesús Javier Doménech Arellano

Enero 2016

1. ID3

2. C4.5

3. C5.0

4. Random Forest

5. Bibliografía

ID3

TDIDT

```
tdidt( C:conjunto_datos, l:atributos_candidatos)
  cp := clase que aparece más veces en C

  si todas las instancias en C son de clase cj
  entonces
    return new Hoja(cj);

  si l es vacía entonces
    return new Hoja(cp);

  a := selecciona_atributo(C,l);
  n := new NodoInterno(a);
  para cada valor aj del atributo a
    Cj := particion(C,a,aj);
    si Cj =  $\emptyset$  entonces n' := new Hoja(cp);
    si no n' := tdidt(Cj,l\{a\});
    n.añadeHijo(n',aj);
  return n;
```

¹Diapositiva sacada del temario de SGDI

Ganancia de información

- Lo más interesante es realizar la selección del atributo teniendo en cuenta la calidad del particionado que genera: nos interesa que los subconjuntos resultantes tengan la mínima variedad de clases posibles.
- Para ello utilizaremos la medida de **ganancia de información** (o reducción de la **entropía**), lo que da lugar al algoritmo **ID3**.

²Diapositiva sacada del temario de SGDI

C4.5

El algoritmo ID3 es mejorado por el C4.5. Esta mejora, aparte de la optimización de partes de código, incluye³:

- Permitir atributos continuos.
- Permitir dar un peso diferente a cada atributo.
- Permitir a una instancia no tener definido un valor en sus atributos.
- Mejorar la selección del atributo clasificador.
- Realizar una poda del árbol después de la creación.

³Artículo con las bases del algoritmo [Quinlan, 1986]

Split y Funciones Test: Para dividir en subconjuntos las instancias test se divide el dominio donde haya mayor ganancia de información. Esta división se traduce en funciones test de tipo $x > 40$ ó $x < 4$, y para atributos discretos $x = \textit{soleado}$.

⁴Obtenido del libro [Quinlan, 1993] y su review [Salzberg, 1994]

Split y Funciones Test: Para dividir en subconjuntos las instancias test se divide el dominio donde haya mayor ganancia de información. Esta división se traduce en funciones test de tipo $x > 40$ ó $x < 4$, y para atributos discretos $x = \textit{soleado}$.

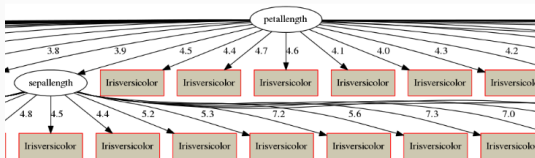
Selección del atributo: La selección del atributo por el que dividir consiste en escoger el atributo con mayor ganancia de información normalizado y ponderado.

Para ello, se tiene en cuenta la proporción de instancias que queda en cada rama para el atributo candidato y el peso de importancia dado a dicho atributo. Siendo D el conjunto de instancias, la fórmula para el atributo i queda:

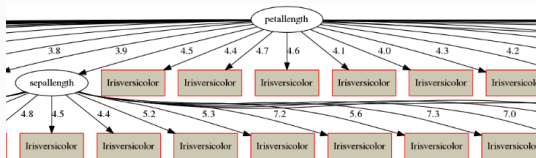
$$Ganancia_i = Peso_i * \sum_j \frac{|D_j|}{|D|} * Info_j$$

⁴Obtenido del libro [Quinlan, 1993] y su review [Salzberg, 1994]

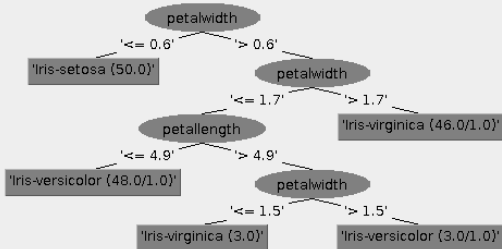
EJEMPLO – ID3 vs C4.5 (ATRIBUTOS CONTINUOS)



EJEMPLO – ID3 vs C4.5 (ATRIBUTOS CONTINUOS)



Tree View



C4.8 (J48): MEJORAS A C4.5

Problemas:

“The accuracy of T2’s trees rivalled or surpassed C4.5’s on 8 of the 15 datasets, including all but one of the datasets having only continuous attributes.” [Auer et al., 1995]

“C4.5’s performance was significantly improved on two data sets [...] using the entropy discretization method and did not significantly degrade on any datasets [...] We conjecture that the C4.5 induction algorithm is not taking full advantage of possible local discretization.” [Dougherty et al., 1995]

C4.8 (J48): MEJORAS A C4.5

Problemas:

“The accuracy of T2’s trees rivalled or surpassed C4.5’s on 8 of the 15 datasets, including all but one of the datasets having only continuous attributes.” [Auer et al., 1995]

“C4.5’s performance was significantly improved on two data sets [...] using the entropy discretization method and did not significantly degrade on any datasets [...] We conjecture that the C4.5 induction algorithm is not taking full advantage of possible local discretization.” [Dougherty et al., 1995]

Solución [Quinlan, 1996]:

- Aumentar el coste de usar atributos continuos disminuyendo el peso de importancia del atributo.
- Simplificar la división de un atributo continuo, en lugar de maximizar la ganancia, basta con superar un límite.
- Se añade el generador de reglas.

C5.0

C5.0 tiene licencia comercial y licencia GPL para un solo proceso.

Mejoras ⁵:

- Aumenta el rendimiento del algoritmo.
- Reduce el consumo de memoria local y total.
- Devuelve árboles de clasificación reducidos.
- Poda atributos irrelevantes para la clasificación.
- Hace la ponderación de atributos más precisa.
- Agrupa valores de atributos discretos en una misma rama.
- Genera reglas con más acierto que las de C4.8.
- Facilita la aplicación de técnicas como bagging y boosting, mejorando sus resultados.⁶

⁵Basado en [Pandya and Pandya, 2015] y la página oficial del autor [Quinlan, 2015]
<http://www.rulequest.com/>

⁶[Freund and Mason, 1999]

C5.0: EJEMPLO EN R

```
crx<-read.table(file="./crx.data",header=FALSE,sep=",")  
head( crx, 6 )
```

Ejemplo de C5.0 en R obtenido de: <http://connor-johnson.com>

C5.0: EJEMPLO EN R

```
crx<-read.table(file="./crx.data",header=FALSE,sep=",")  
head( crx, 6 )  
crx <- crx[ sample( nrow( crx ) ), ]  
X <- crx[,1:15]  
y <- crx[,16]
```

Ejemplo de C5.0 en R obtenido de: <http://connor-johnson.com>

C5.0: EJEMPLO EN R

```
crx<-read.table(file="./crx.data",header=FALSE,sep=",")  
head( crx, 6 )  
crx <- crx[ sample( nrow( crx ) ), ]  
X <- crx[,1:15]  
y <- crx[,16]  
trainX <- X[1:600,]  
trainy <- y[1:600]  
testX <- X[601:690,]  
testy <- y[601:690]
```

Ejemplo de C5.0 en R obtenido de: <http://connor-johnson.com>

C5.0: EJEMPLO EN R

```
crx<-read.table(file="./crx.data",header=FALSE,sep=",")
head( crx, 6 )
crx <- crx[ sample( nrow( crx ) ), ]
X <- crx[,1:15]
y <- crx[,16]

trainX <- X[1:600,]
trainy <- y[1:600]
testX <- X[601:690,]
testy <- y[601:690]
library(C50)
model <- C50::C5.0( trainX, trainy )
summary( model )
```

Ejemplo de C5.0 en R obtenido de: <http://connor-johnson.com>

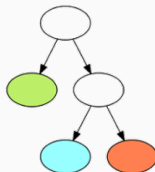
RANDOM FOREST

INTRODUCCIÓN – I

Hasta ahora:

Un conjunto de entrenamiento → Un árbol de clasificación

Sexo	Altura	Peso	Exp.	Act.
H	1,55	45	A+	Fútbol
H	1,67	58	C	Fútbol
M	1,45	45	B	Fútbol
M	1,58	50	A+	Pádel
H	1,20	40	B	Fútbol
M	1,80	60	A	Pádel
...

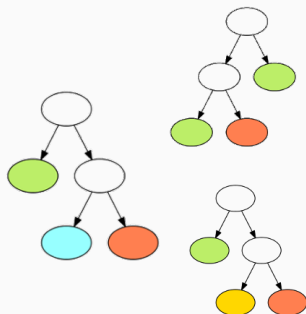


INTRODUCCIÓN – I

Random forest:

Un conjunto de entrenamiento → Varios árboles de clasificación

Sexo	Altura	Peso	Exp.	Act.
H	1,55	45	A+	Fútbol
H	1,67	58	C	Fútbol
M	1,45	45	B	Fútbol
M	1,58	50	A+	Pádel
H	1,20	40	B	Fútbol
M	1,80	60	A	Pádel
...



Random forest⁷

- ¿Cómo se clasifica una instancia?
- ¿Cómo se generan varios árboles con el mismo conjunto de entrenamiento?
- ¿Cómo se genera un árbol en concreto?

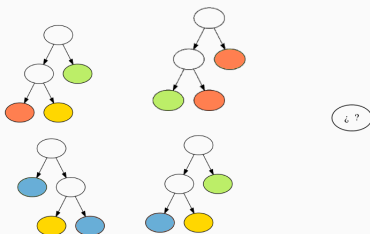
⁷Técnica propuesta por primera vez en [Breiman, 2001].

CLASIFICACIÓN DE INSTANCIAS

Al existir varios árboles, pueden existir varias clases posibles para una instancia.

La clase final será aquella que más veces aparece elegida (**moda**).

```
1 clasifica(x):  
2   for_each a in arboles:  
3     cjto += clasifica(x, a)  
4  
5   return moda(cjto)
```

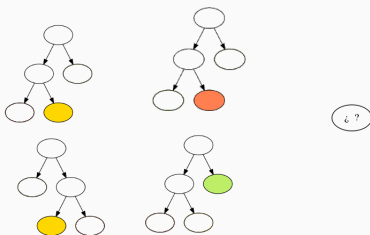


CLASIFICACIÓN DE INSTANCIAS

Al existir varios árboles, pueden existir varias clases posibles para una instancia.

La clase final será aquella que más veces aparece elegida (**moda**).

```
1 clasifica(x):  
2   for_each a in arboles:  
3     cjto += clasifica(x, a)  
4  
5   return moda(cjto)
```

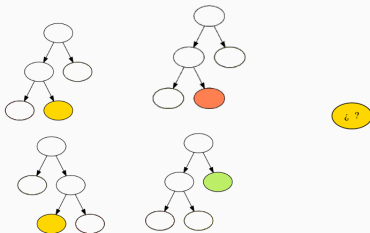


CLASIFICACIÓN DE INSTANCIAS

Al existir varios árboles, pueden existir varias clases posibles para una instancia.

La clase final será aquella que más veces aparece elegida (**moda**).

```
1 clasifica(x):  
2   for_each a in arboles:  
3     cjto += clasifica(x, a)  
4  
5   return moda(cjto)
```



GENERACIÓN DE ÁRBOLES — SELECCIÓN DE INSTANCIAS

Problema: A partir de un mismo conjunto de entrenamiento se desean obtener distintos árboles de clasificación.

Solución: Técnica conocida como **Bagging** o *Bootstrap aggregating*:

- Dado un conjunto de entrenamiento con N instancias, obtener B conjuntos de entrenamiento de tamaño n' ($n' < N$).
- Utilizar *muestreo aleatorio con reemplazamiento*.

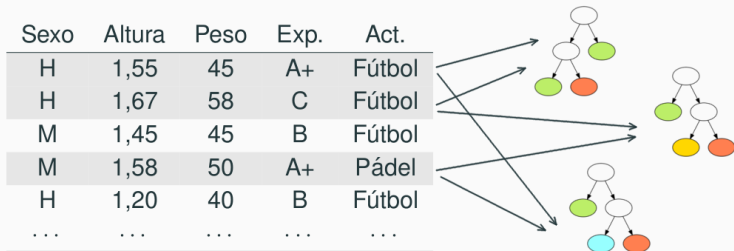
Sexo	Altura	Peso	Exp.	Act.
H	1,55	45	A+	Fútbol
H	1,67	58	C	Fútbol
M	1,45	45	B	Fútbol
M	1,58	50	A+	Pádel
H	1,20	40	B	Fútbol
...

GENERACIÓN DE ÁRBOLES — SELECCIÓN DE INSTANCIAS

Problema: A partir de un mismo conjunto de entrenamiento se desean obtener distintos árboles de clasificación.

Solución: Técnica conocida como **Bagging** o *Bootstrap aggregating*:

- Dado un conjunto de entrenamiento con N instancias, obtener B conjuntos de entrenamiento de tamaño n' ($n' < N$).
- Utilizar *muestreo aleatorio con reemplazamiento*.



– En un nodo split, ¿qué atributo escoger para hacer la división? –

ID3: Escoger entre `todos` los atributos, aquel que proporcione una mayor ganancia de información.

Problemas:

- Un atributo muy decisivo en un árbol seguramente también lo sea en el resto de árboles, por lo que producirá árboles muy correlacionados.
- Comparar entre todos los atributos en todos los árboles necesita muchos recursos.

– En un nodo split, ¿qué atributo escoger para hacer la división? –

Solución: Feature bagging⁸:

- En cada nodo split, observar solamente un subconjunto aleatorio de tamaño m' del conjunto de atributos restantes.
- Sobre el subconjunto, escoger el que mayor ganancia de información proporcione (similar a ID3).
- Si el número de atributos totales es m , el subconjunto de tamaño m' se calcula de *manera aleatoria*, donde:

$$m' = \begin{cases} \lfloor \sqrt{m} \rfloor & \text{si } \sqrt{m} \geq 5 \\ m & \text{c.c.} \end{cases}$$

⁸Descrita principalmente en [Bryll et al., 2003]

BIBLIOGRAFÍA



Auer, P., Holte, R. C., and Maass, W. (1995).

Theory and applications of agnostic pac-learning with small decision trees.

In Proceedings of the Twelfth International Conference on Machine Learning, pages 21–29.



Breiman, L. (2001).

Random forests.

Machine Learning, 45(1):5–32.



Bryll, R., Gutierrez-Osuna, R., and Quek, F. (2003).

Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets.

Pattern Recognition, 36(6):1291 – 1302.



Dougherty, J., Kohavi, R., Sahami, M., et al. (1995).

Supervised and unsupervised discretization of continuous features.

In *Machine learning: proceedings of the twelfth international conference*, volume 12, pages 194–202.



Freund, Y. and Mason, L. (1999).

The alternating decision tree learning algorithm.

In *icml*, volume 99, pages 124–133.



Pandya, R. and Pandya, J. (2015).

C5. 0 algorithm to improved decision tree with feature selection and reduced error pruning.

International Journal of Computer Applications, 117(16).



Quinlan, J. R. (1986).

Induction of decision trees.

Machine learning, 1(1):81–106.



Quinlan, J. R. (1993).

C4.5: programs for machine learning.

Morgan Kaufmann publishers.

pages 17–36, 45–56.



Quinlan, J. R. (1996).

Improved use of continuous attributes in c4.5.

Journal of artificial intelligence research, pages 77–90.



Quinlan, J. R. (2015).

<http://www.rulequest.com/>.

[Web; Último acceso: 28-12-2015].



Salzberg, S. L. (1994).

**C4. 5: Programs for machine learning by J. Ross Quinlan.
Morgan Kaufmann publishers, inc., 1993.**

Machine Learning, 16(3):235–240.

¿ ?
