Teoría sobre Árboles de Decisión

Dr. Oldemar Rodríguez R.

UCR

26 de agosto de 2022

Introducción

- El problema general que se ataca en esta presentación es el de la predicción de una variable, llamada *variable a predecir* o *variable dependiente*, utilizando un conjunto de variables llamadas *variables explicativas* o *predictoras*.
- El objetivo es presentar el método CART (Classification and Regression Trees) que es un método para obtener predicciones vía la extracción de reglas de decisión, a partir de un árbol binario.
- La construcción de un "buen" árbol se hace por divisiones sucesivas de la muestra de datos en submuestras, de acuerdo con el orden decreciente del poder separador de los predictores.
- El problema de cuando detener el proceso de división de la muestra, es el que se trata de resolver con el método CART, ya que otros algoritmos anteriores no habían logrado ofrecer una solución satisfactoria a este problema.

Introducción

- El problema general que se ataca en esta presentación es el de la predicción de una variable, llamada *variable a predecir* o *variable dependiente*, utilizando un conjunto de variables llamadas *variables explicativas* o *predictoras*.
- El objetivo es presentar el método CART (Classification and Regression Trees) que es un método para obtener predicciones vía la extracción de reglas de decisión, a partir de un árbol binario.
- La construcción de un "buen" árbol se hace por divisiones sucesivas de la muestra de datos en submuestras, de acuerdo con el orden decreciente del poder separador de los predictores.
- El problema de cuando detener el proceso de división de la muestra, es el que se trata de resolver con el método CART, ya que otros algoritmos anteriores no habían logrado ofrecer una solución satisfactoria a este problema.

Introducción

- El problema general que se ataca en esta presentación es el de la predicción de una variable, llamada *variable a predecir* o *variable dependiente*, utilizando un conjunto de variables llamadas *variables explicativas* o *predictoras*.
- El objetivo es presentar el método CART (Classification and Regression Trees) que es un método para obtener predicciones vía la extracción de reglas de decisión, a partir de un árbol binario.
- La construcción de un "buen" árbol se hace por divisiones sucesivas de la muestra de datos en submuestras, de acuerdo con el orden decreciente del poder separador de los predictores.
- El problema de cuando detener el proceso de división de la muestra, es el que se trata de resolver con el método CART, ya que otros algoritmos anteriores no habían logrado ofrecer una solución satisfactoria a este problema.

Introducción

- El problema general que se ataca en esta presentación es el de la predicción de una variable, llamada *variable a predecir* o *variable dependiente*, utilizando un conjunto de variables llamadas *variables explicativas* o *predictoras*.
- El objetivo es presentar el método CART (Classification and Regression Trees) que es un método para obtener predicciones vía la extracción de reglas de decisión, a partir de un árbol binario.
- La construcción de un "buen" árbol se hace por divisiones sucesivas de la muestra de datos en submuestras, de acuerdo con el orden decreciente del poder separador de los predictores.
- El problema de cuando detener el proceso de división de la muestra, es el que se trata de resolver con el método CART, ya que otros algoritmos anteriores no habían logrado ofrecer una solución satisfactoria a este problema.

Introducción

Globalmente, el método CART consta de dos partes fundamentales:

1. **Construcción del árbol $A_{\max}$**: Se construye un árbol binario haciendo divisiones sucesivas hasta obtener nodos muy pequeños o nodos formados únicamente por elementos de uno de los grupos a priori (nodos puros).
2. **Etapa de poda de $A_{\max}$**: Como resultado de podar "adecuadamente" el árbol $A_{\max}$ se obtiene un subárbol óptimo en el sentido del costo-complejidad, cuestión que será expuesta más adelante. Este subárbol puede ser un buen instrumento de predicción, o puede necesitar una depuración ulterior, dependiendo del problema que se quiere resolver.

Introducción

Globalmente, el método CART consta de dos partes fundamentales:

1. **Construcción del árbol $A_{\max}$**: Se construye un árbol binario haciendo divisiones sucesivas hasta obtener nodos muy pequeños o nodos formados únicamente por elementos de uno de los grupos a priori (nodos puros).
2. **Etapa de poda de $A_{\max}$**: Como resultado de podar "adecuadamente" el árbol $A_{\max}$ se obtiene un subárbol óptimo en el sentido del costo-complejidad, cuestión que será expuesta más adelante. Este subárbol puede ser un buen instrumento de predicción, o puede necesitar una depuración ulterior, dependiendo del problema que se quiere resolver.

Definiciones y notaciones básicas

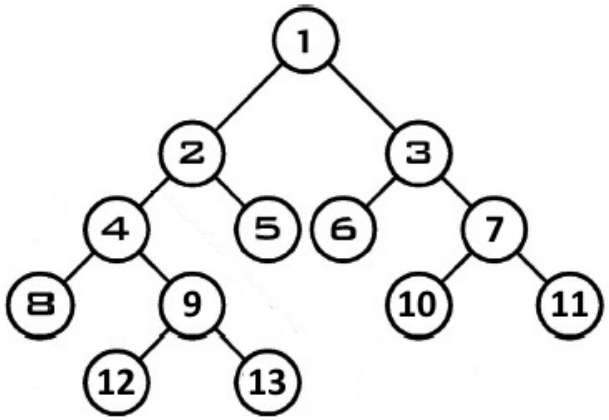
Definición (Árbol Binario)

Sea $\emptyset \neq A \subseteq \mathbb{N}$ *con* $\mid A \mid < \infty$. *Se dice que* (A, Izq, Der) *es un árbol binario*[a] *si las funciones* $Izq, Der : A \longrightarrow A \cup \{0\}$ *tienen las siguientes propiedades:*

1. $\forall\ t, (Izq(t) = 0 = Der(t))$ *ó* $(Izq(t) > 0\ y\ Der(t) > 0)$.
2. $\forall\ t \neq\ min(A), \exists\ s \in A$ *único tal que* $t = Izq(s)$ *ó* $t = Der(s)$.

[a] Si no hay ambigüedad se dirá simplemente que A es un árbol.

Gráficamente:



Definiciones y notaciones básicas

Dado un árbol binario A, se utiliza la siguiente terminología clásica:

- **Nodo**: todo elemento de A se llama nodo ó vértice.
- **Nodo padre**: un nodo t es el padre del nodo u si $\text{Izq}(t) = u$ ó $\text{Der}(t) = u$. Escribimos $t = p(u)$. La propiedad 2 de la definición de árbol significa que todo vértice, excepto $\min(A)$, tiene un único padre.
- **Raíz**: la raíz de A es un nodo que no tiene padre. Por la propiedad 2 de la definición de árbol, $\min(A)$ es el único nodo que no tiene padre, por lo tanto dicho nodo es la raíz de A.
- **Hijo izquierdo, hijo derecho**: un nodo u es el hijo izquierdo (resp. derecho) de t si $\text{Izq}(t) = u$ (resp. $\text{Der}(t) = u$).
- **Antecesor y sucesor**: un nodo t es un antecesor de un nodo u (y u es un sucesor de t) si existe $n \in \mathbb{N}$ tal que $f^n(t) = u$ donde en la composición $f = \text{Izq}$ ó $f = \text{Der}$.
- **Nodo terminal**: un nodo t es terminal (es una hoja) si $\text{Izq}(t) = 0$ y $\text{Der}(t) = 0$. Es decir, t es terminal si no tiene sucesores. El conjunto de nodos terminales de A se denota $\widetilde{A}$.

Definiciones y notaciones básicas

Dado un árbol binario A, se utiliza la siguiente terminología clásica:

- **Nodo**: todo elemento de A se llama nodo ó vértice.
- **Nodo padre**: un nodo t es el padre del nodo u si $\text{Izq}(t) = u$ ó $\text{Der}(t) = u$. Escribimos $t = p(u)$. La propiedad 2 de la definición de árbol significa que todo vértice, excepto $\min(A)$, tiene un único padre.
- **Raíz**: la raíz de A es un nodo que no tiene padre. Por la propiedad 2 de la definición de árbol, $\min(A)$ es el único nodo que no tiene padre, por lo tanto dicho nodo es la raíz de A.
- **Hijo izquierdo, hijo derecho**: un nodo u es el hijo izquierdo (resp. derecho) de t si $\text{Izq}(t) = u$ (resp. $\text{Der}(t) = u$).
- **Antecesor y sucesor**: un nodo t es un antecesor de un nodo u (y u es un sucesor de t) si existe $n \in \mathbb{N}$ tal que $f^n(t) = u$ donde en la composición $f = \text{Izq}$ ó $f = \text{Der}$.
- **Nodo terminal**: un nodo t es terminal (es una hoja) si $\text{Izq}(t) = 0$ y $\text{Der}(t) = 0$. Es decir, t es terminal si no tiene sucesores. El conjunto de nodos terminales de A se denota $\widetilde{A}$.

Definiciones y notaciones básicas

Dado un árbol binario A, se utiliza la siguiente terminología clásica:

- **Nodo**: todo elemento de A se llama nodo ó vértice.
- **Nodo padre**: un nodo t es el padre del nodo u si $\text{Izq}(t) = u$ ó $\text{Der}(t) = u$. Escribimos $t = p(u)$. La propiedad 2 de la definición de árbol significa que todo vértice, excepto $\min(A)$, tiene un único padre.
- **Raíz**: la raíz de A es un nodo que no tiene padre. Por la propiedad 2 de la definición de árbol, $\min(A)$ es el único nodo que no tiene padre, por lo tanto dicho nodo es la raíz de A.
- **Hijo izquierdo, hijo derecho**: un nodo u es el hijo izquierdo (resp. derecho) de t si $\text{Izq}(t) = u$ (resp. $\text{Der}(t) = u$).
- **Antecesor y sucesor**: un nodo t es un antecesor de un nodo u (y u es un sucesor de t) si existe $n \in \mathbb{N}$ tal que $f^n(t) = u$ donde en la composición $f = \text{Izq}$ ó $f = \text{Der}$.
- **Nodo terminal**: un nodo t es terminal (es una hoja) si $\text{Izq}(t) = 0$ y $\text{Der}(t) = 0$. Es decir, t es terminal si no tiene sucesores. El conjunto de nodos terminales de A se denota $\widetilde{A}$.

Definiciones y notaciones básicas

Dado un árbol binario A, se utiliza la siguiente terminología clásica:

- **Nodo**: todo elemento de A se llama nodo ó vértice.
- **Nodo padre**: un nodo t es el padre del nodo u si $\text{Izq}(t) = u$ ó $\text{Der}(t) = u$. Escribimos $t = p(u)$. La propiedad 2 de la definición de árbol significa que todo vértice, excepto $\min(A)$, tiene un único padre.
- **Raíz**: la raíz de A es un nodo que no tiene padre. Por la propiedad 2 de la definición de árbol, $\min(A)$ es el único nodo que no tiene padre, por lo tanto dicho nodo es la raíz de A.
- **Hijo izquierdo, hijo derecho**: un nodo u es el hijo izquierdo (resp. derecho) de t si $\text{Izq}(t) = u$ (resp. $\text{Der}(t) = u$).
- **Antecesor y sucesor**: un nodo t es un antecesor de un nodo u (y u es un sucesor de t) si existe $n \in \mathbb{N}$ tal que $f^n(t) = u$ donde en la composición $f = \text{Izq}$ ó $f = \text{Der}$.
- **Nodo terminal**: un nodo t es terminal (es una hoja) si $\text{Izq}(t) = 0$ y $\text{Der}(t) = 0$. Es decir, t es terminal si no tiene sucesores. El conjunto de nodos terminales de A se denota $\widetilde{A}$.

Definiciones y notaciones básicas

Dado un árbol binario A, se utiliza la siguiente terminología clásica:

- **Nodo**: todo elemento de A se llama nodo ó vértice.
- **Nodo padre**: un nodo t es el padre del nodo u si $\text{Izq}(t) = u$ ó $\text{Der}(t) = u$. Escribimos $t = p(u)$. La propiedad 2 de la definición de árbol significa que todo vértice, excepto $\min(A)$, tiene un único padre.
- **Raíz**: la raíz de A es un nodo que no tiene padre. Por la propiedad 2 de la definición de árbol, $\min(A)$ es el único nodo que no tiene padre, por lo tanto dicho nodo es la raíz de A.
- **Hijo izquierdo, hijo derecho**: un nodo u es el hijo izquierdo (resp. derecho) de t si $\text{Izq}(t) = u$ (resp. $\text{Der}(t) = u$).
- **Antecesor y sucesor**: un nodo t es un antecesor de un nodo u (y u es un sucesor de t) si existe $n \in \mathbb{N}$ tal que $f^n(t) = u$ donde en la composición $f = \text{Izq}$ ó $f = \text{Der}$.
- **Nodo terminal**: un nodo t es terminal (es una hoja) si $\text{Izq}(t) = 0$ y $\text{Der}(t) = 0$. Es decir, t es terminal si no tiene sucesores. El conjunto de nodos terminales de A se denota $\widetilde{A}$.

Definiciones y notaciones básicas

Dado un árbol binario A, se utiliza la siguiente terminología clásica:

- **Nodo**: todo elemento de A se llama nodo ó vértice.
- **Nodo padre**: un nodo t es el padre del nodo u si $\text{Izq}(t) = u$ ó $\text{Der}(t) = u$. Escribimos $t = p(u)$. La propiedad 2 de la definición de árbol significa que todo vértice, excepto $\min(A)$, tiene un único padre.
- **Raíz**: la raíz de A es un nodo que no tiene padre. Por la propiedad 2 de la definición de árbol, $\min(A)$ es el único nodo que no tiene padre, por lo tanto dicho nodo es la raíz de A.
- **Hijo izquierdo, hijo derecho**: un nodo u es el hijo izquierdo (resp. derecho) de t si $\text{Izq}(t) = u$ (resp. $\text{Der}(t) = u$).
- **Antecesor y sucesor**: un nodo t es un antecesor de un nodo u (y u es un sucesor de t) si existe $n \in \mathbb{N}$ tal que $f^n(t) = u$ donde en la composición $f = \text{Izq}$ ó $f = \text{Der}$.
- **Nodo terminal**: un nodo t es terminal (es una hoja) si $\text{Izq}(t) = 0$ y $\text{Der}(t) = 0$. Es decir, t es terminal si no tiene sucesores. El conjunto de nodos terminales de A se denota $\widetilde{A}$.

Definiciones y notaciones básicas

Definición (Subárbol)

Sea A un árbol y $\emptyset \neq B \subseteq A$. Se definen las funciones $Izq_1, Der_1 : B \longrightarrow B \cup \{0\}$ tales que:

$$Izq_1(s) = \begin{cases} Izq(s) & si\ Izq(s) \in B \\ 0 & si\ no \end{cases}$$

$$Der_1(s) = \begin{cases} Der(s) & si\ Der(s) \in B \\ 0 & si\ no \end{cases}$$

Notación

Si $(B, \text{Izq}_1, \text{Der}_1)$ es un árbol binario entonces el triplete $(B, \text{Izq}_1, \text{Der}_1)$ se llama un subárbol de A. Si no hay ambigüedad, decimos simplemente que B es un subárbol de A y escribimos $B \leq A$. Más adelante se demuestra que la unión de subárboles no necesariamente es un subárbol.

Definiciones y notaciones básicas

Definición (Subárbol)

Sea A un árbol y $\emptyset \neq B \subseteq A$. Se definen las funciones $Izq_1, Der_1 : B \longrightarrow B \cup \{0\}$ tales que:

$$Izq_1(s) = \begin{cases} Izq(s) & si\ Izq(s) \in B \\ 0 & si\ no \end{cases}$$

$$Der_1(s) = \begin{cases} Der(s) & si\ Der(s) \in B \\ 0 & si\ no \end{cases}$$

Notación

Si $(B, \text{Izq}_1, \text{Der}_1)$ es un árbol binario entonces el triplete $(B, \text{Izq}_1, \text{Der}_1)$ se llama un subárbol de A. Si no hay ambigüedad, decimos simplemente que B es un subárbol de A y escribimos $B \leq A$. Más adelante se demuestra que la unión de subárboles no necesariamente es un subárbol.

Definiciones y notaciones básicas

Definición (Rama)

Sea $t \in A$, una rama A_t de A es el subárbol de A que consta de t (la raíz de A_t) y todos sus sucesores. En este caso las funciones Izq_1 y Der_1 son las restricciones de Izq y Der al conjunto A_t, respectivamente.

Gráficamente: En la Figura 2 se presenta el árbol cuyo conjunto de nodos es $A = \{1, 2, \ldots, 13\}$.

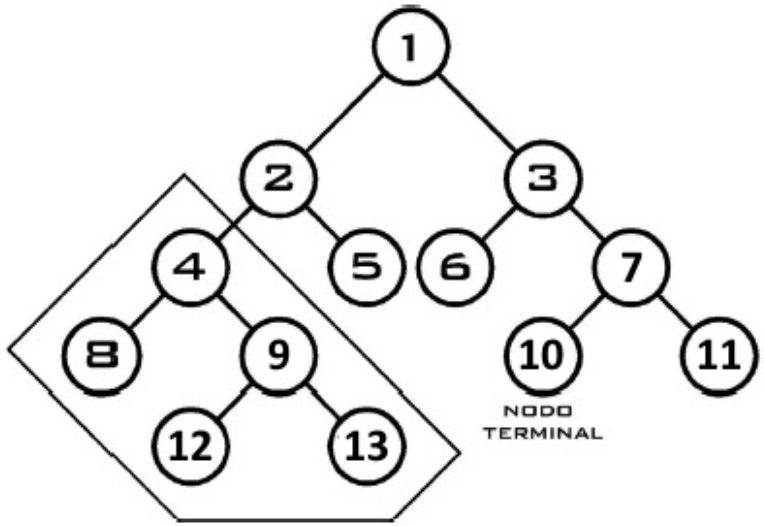


Figura: 2. Ejemplo de un árbol binario.

Funciones numTer, Izq y Der

Definición

- *El número* $numTer(t)$ *indica la cantidad total de los últimos nodos terminales consecutivos contados hasta* $t-1$, *donde* t *es no terminal.*
- *Se definen las funciones Izq y Der para nodos no terminales como sigue:*
  1. $Der(t) = Izq(t) + 1$ *para todo* $t \geq 1$.
  2. $Izq(1) = 2$.
  3. *Y para todo* $t \geq 2$ *no terminal se define:*

$$Izq(t) = Izq(t - 1 - numTer(t)) + 2.$$

Ejemplo:

¿Qué significa esto en el árbol de la Figura 2?

- $\text{Izq}(7) = \text{Izq}(7 - 1 - \text{numTer}(7)) + 2$ donde $\text{numTer}(7) = 2$, luego $\text{Izq}(7) = \text{Izq}(4) + 2 = 8 + 2 = 10$.
- Por otra parte, es fácil ver que A es un árbol binario (Figura 2) y que el nodo 13 es un sucesor del nodo 2 puesto que $(\text{Der} \circ \text{Der} \circ \text{Izq})(2) = 13$.
- El conjunto $B = \{4, 8, 9, 12, 13\}$ es un subárbol de A pero si quitamos el nodo 8 entonces B no es un subárbol de A, puesto que $\text{Izq}_1(4) = 0$ y $\text{Der}_1(4) = 9$, lo que es imposible en un subárbol.
- Por otra parte, $B \cup A_3$ **no** es un subárbol de A ya que la parte 2 de la definición de árbol no es válida.

Ejemplo:

¿Qué significa esto en el árbol de la Figura 2?

- $\text{Izq}(7) = \text{Izq}(7 - 1 - \text{numTer}(7)) + 2$ donde $\text{numTer}(7) = 2$, luego $\text{Izq}(7) = \text{Izq}(4) + 2 = 8 + 2 = 10$.
- Por otra parte, es fácil ver que A es un árbol binario (Figura 2) y que el nodo 13 es un sucesor del nodo 2 puesto que $(\text{Der} \circ \text{Der} \circ \text{Izq})(2) = 13$.
- El conjunto $B = \{4, 8, 9, 12, 13\}$ es un subárbol de A pero si quitamos el nodo 8 entonces B no es un subárbol de A, puesto que $\text{Izq}_1(4) = 0$ y $\text{Der}_1(4) = 9$, lo que es imposible en un subárbol.
- Por otra parte, $B \cup A_3$ **no** es un subárbol de A ya que la parte 2 de la definición de árbol no es válida.

Ejemplo:

¿Qué significa esto en el árbol de la Figura 2?

- $\text{Izq}(7) = \text{Izq}(7 - 1 - \text{numTer}(7)) + 2$ donde $\text{numTer}(7) = 2$, luego $\text{Izq}(7) = \text{Izq}(4) + 2 = 8 + 2 = 10$.
- Por otra parte, es fácil ver que A es un árbol binario (Figura 2) y que el nodo 13 es un sucesor del nodo 2 puesto que $(\text{Der} \circ \text{Der} \circ \text{Izq})(2) = 13$.
- El conjunto $B = \{4, 8, 9, 12, 13\}$ es un subárbol de A pero si quitamos el nodo 8 entonces B no es un subárbol de A, puesto que $\text{Izq}_1(4) = 0$ y $\text{Der}_1(4) = 9$, lo que es imposible en un subárbol.
- Por otra parte, $B \cup A_3$ **no** es un subárbol de A ya que la parte 2 de la definición de árbol no es válida.

Ejemplo:

¿Qué significa esto en el árbol de la Figura 2?

- $\text{Izq}(7) = \text{Izq}(7 - 1 - \text{numTer}(7)) + 2$ donde $\text{numTer}(7) = 2$, luego $\text{Izq}(7) = \text{Izq}(4) + 2 = 8 + 2 = 10$.
- Por otra parte, es fácil ver que A es un árbol binario (Figura 2) y que el nodo 13 es un sucesor del nodo 2 puesto que $(\text{Der} \circ \text{Der} \circ \text{Izq})(2) = 13$.
- El conjunto $B = \{4, 8, 9, 12, 13\}$ es un subárbol de A pero si quitamos el nodo 8 entonces B no es un subárbol de A, puesto que $\text{Izq}_1(4) = 0$ y $\text{Der}_1(4) = 9$, lo que es imposible en un subárbol.
- Por otra parte, $B \cup A_3$ **no** es un subárbol de A ya que la parte 2 de la definición de árbol no es válida.

Construcción del árbol $A_{\max}$

Notación:

- Sean $y^1, \ldots, y^p$ el conjunto de variables predictoras y z la variable categórica a predecir, llamada también variable de clase o de grupo.
- Sea $\mathcal{I} = \{1, 2, \ldots, n\}$ el conjunto de individuos, $Y = [y^1 | \cdots | y^p | z]$ la matriz de datos de tamaño $n \times (p+1)$, donde la última columna z es la variable de grupo con r modalidades.
- La variable z determina una partición de $\mathcal{I}$ en r grupos, llamados grupos a priori por ser dados exógenamente al modelo.
- Otros términos usados en este contexto en lugar de predicción, son "discriminación" o "reconocimiento" de estos grupos a priori.

Construcción del árbol $A_{\max}$

Notación:

- Sean $y^1, \ldots, y^p$ el conjunto de variables predictoras y z la variable categórica a predecir, llamada también variable de clase o de grupo.
- Sea $\mathcal{I} = \{1, 2, \ldots, n\}$ el conjunto de individuos, $Y = [y^1 | \cdots | y^p | z]$ la matriz de datos de tamaño $n \times (p+1)$, donde la última columna z es la variable de grupo con r modalidades.
- La variable z determina una partición de $\mathcal{I}$ en r grupos, llamados grupos a priori por ser dados exógenamente al modelo.
- Otros términos usados en este contexto en lugar de predicción, son "discriminación" o "reconocimiento" de estos grupos a priori.

Construcción del árbol $A_{\max}$

Notación:

- Sean $y^1, \ldots, y^p$ el conjunto de variables predictoras y z la variable categórica a predecir, llamada también variable de clase o de grupo.
- Sea $\mathcal{I} = \{1, 2, \ldots, n\}$ el conjunto de individuos, $Y = [y^1 | \cdots | y^p | z]$ la matriz de datos de tamaño $n \times (p+1)$, donde la última columna z es la variable de grupo con r modalidades.
- La variable z determina una partición de $\mathcal{I}$ en r grupos, llamados grupos a priori por ser dados exógenamente al modelo.
- Otros términos usados en este contexto en lugar de predicción, son "discriminación" o "reconocimiento" de estos grupos a priori.

Construcción del árbol $A_{\max}$

Notación:

- Sean $y^1, \ldots, y^p$ el conjunto de variables predictoras y z la variable categórica a predecir, llamada también variable de clase o de grupo.
- Sea $\mathcal{I} = \{1, 2, \ldots, n\}$ el conjunto de individuos, $Y = [y^1 | \cdots | y^p | z]$ la matriz de datos de tamaño $n \times (p+1)$, donde la última columna z es la variable de grupo con r modalidades.
- La variable z determina una partición de $\mathcal{I}$ en r grupos, llamados grupos a priori por ser dados exógenamente al modelo.
- Otros términos usados en este contexto en lugar de predicción, son "discriminación" o "reconocimiento" de estos grupos a priori.

Ejemplo de Tabla de Datos:

A continuación se da un ejemplo de matriz de datos $Y = [y^1|y^2|y^3|y^4|\,z]$ de tamaño 15×5, donde las modalidades de y^1 son $\{1,2,3\}$, las de y^2 son $\{1,2,3,4,5,6\}$, las de y^3 son $\{1,2\}$, las de y^4 son $\{1,2,3,4\}$ y las de z son $\{1,2\}$:

	Predictores				
Individuos	y^1	y^2	y^3	y^4	z
1	1	6	2	3	1
2	1	2	2	4	1
3	1	6	2	2	1
4	1	4	2	3	1
5	2	1	1	1	1
6	2	3	2	3	2
7	2	5	2	3	2
8	2	6	1	4	2
9	2	5	2	3	2
10	2	3	1	1	2
11	1	1	1	1	2
12	2	5	2	2	2
13	1	6	2	2	2
14	2	4	2	2	2
15	3	5	2	3	2

Pregunta binaria, división binaria

Definición

Sea v un nodo de un árbol binario, $\emptyset \neq v \subseteq \mathcal{I}$; $y^{j,v}$ es la restricción del predictor y^j al nodo v. Se define el concepto "pregunta binaria" de acuerdo con el tipo de variable y^j.

- *y^j* ***es un predictor categórico****: si y^j tiene m modalidades $\{1, \ldots, m\}$, entonces para cada $q \neq \emptyset$, $q \subset \{1, \ldots, m\}$, la pregunta binaria respectiva es ¿$y^{j,v} \in q$? En este caso hay $2^{m-1} - 1$ conjuntos q que determinan igual número de preguntas binarias (no se toma en cuenta el conjunto vacio). La cantidad de conjuntos se puede determinar con ayuda de las fórmulas:*

$$2^{m-1} - 1 = \begin{cases} \displaystyle\sum_{s=1}^{k} \frac{m!}{(m-s)!\ s!} & si\ m = 2k+1; k \geq 1 \\ \dfrac{(m-1)!}{(m-k)!\ (k-1)!} + \displaystyle\sum_{s=1}^{k-1} \frac{m!}{(m-s)!\ s!} & si\ m = 2k; k \geq 1. \end{cases}$$

Pregunta binaria, división binaria

Ejemplo:

Sea $m = 7$ y $k = 3$. Entonces:

- $\frac{7!}{(7-1)!\ 1!} = 7$ corresponde al número de conjuntos unitarios $\{x\}$ con $x \in \{1, \ldots, 7\}$.
- $\frac{7!}{(7-2)!\ 2!} = 21$ corresponde al número de conjuntos con dos elementos $\{x, y\}$ con $x, y \in \{1, \ldots, 7\}$.
- $\frac{7!}{(7-3)!\ 3!} = 35$ corresponde al número de conjuntos con tres elementos $\{x, y, z\}$ con $x, y, z \in \{1, \ldots, 7\}$.
- De donde se cumple la relación $7 + 21 + 35 = 63 = 2^7 - 1$.

Pregunta binaria, división binaria

Definición (Continuación)

- y^j ***es un predictor categórico ordinal***: *si* y^j *tiene* m *modalidades ordenadas* $1 < \cdots < m$, *entonces hay exactamente* $m-1$ *preguntas binarias de la forma:*

$$¿y^{j,v} \in q? \text{ donde } q = \{1, \ldots, x\} \text{ con } x \in \{1, \ldots, m-1\}.$$

También se escribe $¿y^{j,v} \leq x?$ *En cualquier caso, cada pregunta binaria* $¿y^{j,v} \in q?$ *determina una partición del nodo* v, *llamada división binaria.*

- *Si* (v_i, v_d) *denota dicha partición, entonces:*

$$v_i = \{s \in v \mid y^{j,v}(s) \in q\} \;\; y \;\; v_d = \{s \in v \mid y^{j,v}(s) \in v - q\}.$$

Es claro que $v = v_i \cup v_d$ *y* $v_i \cap v_d = \emptyset$.

Ejemplo: Pregunta binaria, división binaria

En la tabla de abajo considere $q = \{1, 2\}$ y $v = \{1, 2, 8, 11, 15\}$. La división binaria correspondiente a la pregunta binaria ¿$y^{2,v} \in q$? es (v_i, v_d), donde $v_i = \{2, 11\}$ y $v_d = \{1, 8, 15\}$, puesto que $y^{2,2} = 2$ y $y^{2,11} = 1$ mientras que $y^{2,1} = 6$, $y^{2,8} = 6$ y $y^{2,15} = 5$.

	Predictores				
Individuos	y^1	y^2	y^3	y^4	z
1	1	6	2	3	1
2	1	2	2	4	1
3	1	6	2	2	1
4	1	4	2	3	1
5	2	1	1	1	1
6	2	3	2	3	2
7	2	5	2	3	2
8	2	6	1	4	2
9	2	5	2	3	2
10	2	3	1	1	2
11	1	1	1	1	2
12	2	5	2	2	2
13	1	6	2	2	2
14	2	4	2	2	2
15	3	5	2	3	2

Criterio de impureza

En el proceso de construcción del árbol $A_{\max}$, las variables predictoras generan un conjunto de preguntas binarias en el nodo actual que se quiere dividir. La idea es escoger entre todas ellas, la mejor, en el sentido de un criterio de impureza, como lo definen Breiman et.al.

Definición

- *Una función* $\phi : [0,1]^r \longrightarrow [0,\infty[$ *se llama* **Función de Impureza** *si tiene las siguientes propiedades:*
  1. *Para cualquier* $i \in \{1,\ldots,r\}$, $\phi(e_i) = min\{\phi(x_1,\ldots,x_r) \mid \sum_{i=1}^{r} x_i = 1\}$, *donde* e_i *es el vector que tiene un* 1 *en la celda* i *y* 0 *en las restantes.*
  2. $\phi(\frac{1}{r},\ldots,\frac{1}{r}) = max\{\phi(x_1,\ldots,x_r) \mid \sum_{i=1}^{r} x_i = 1\}$.
  3. ϕ *es simétrica, esto es, para cualquier permutación* σ *de* r *letras:*

$$\phi(x_1,\ldots,x_r) = \phi(x_{\sigma(1)},\ldots,x_{\sigma(r)}).$$

Criterio de impureza

En el proceso de construcción del árbol $A_{\max}$, las variables predictoras generan un conjunto de preguntas binarias en el nodo actual que se quiere dividir. La idea es escoger entre todas ellas, la mejor, en el sentido de un criterio de impureza, como lo definen Breiman et.al.

Definición

- *Una función* $\phi : [0,1]^r \longrightarrow [0,\infty[$ *se llama* **Función de Impureza** *si tiene las siguientes propiedades:*
  1. *Para cualquier* $i \in \{1,\ldots,r\}$, $\phi(e_i) = min\{\phi(x_1,\ldots,x_r) \mid \sum_{i=1}^{r} x_i = 1\}$, *donde* e_i *es el vector que tiene un* 1 *en la celda* i *y* 0 *en las restantes.*
  2. $\phi(\frac{1}{r},\ldots,\frac{1}{r}) = max\{\phi(x_1,\ldots,x_r) \mid \sum_{i=1}^{r} x_i = 1\}$.
  3. ϕ *es simétrica, esto es, para cualquier permutación* σ *de* r *letras:*

$$\phi(x_1,\ldots,x_r) = \phi(x_{\sigma(1)},\ldots,x_{\sigma(r)}).$$

Impureza de un nodo

Definición

- *Sea* $p(s|v) = \frac{|E_s \cap v|}{|v|}$ *la probabilidad del grupo a priori* E_s *en el nodo* v*;* $s = 1, \ldots, r$*. La impureza de* v *es:*

$$Imp(v) = \phi(p(1|v), \ldots, p(r|v)).$$

donde ϕ *es una función de impureza.*

- *El nodo* v *es puro si* $Imp(v) = 0$*.*

Partiendo de esta definición, Breiman et. al. plantearon el problema de seleccionar la mejor división de un nodo v en nodo izquierdo v_i y en nodo derecho v_d, mediante el criterio del Descenso de la Impureza (Información Ganada), que se introduce más adelante.

Impureza de un nodo

Definición

- *Sea* $p(s|v) = \frac{|E_s \cap v|}{|v|}$ *la probabilidad del grupo a priori* E_s *en el nodo* v*;* $s = 1, \ldots, r$*. La impureza de* v *es:*

$$Imp(v) = \phi(p(1|v), \ldots, p(r|v)).$$

  donde ϕ *es una función de impureza.*
- *El nodo* v *es puro si* $Imp(v) = 0$*.*

Partiendo de esta definición, Breiman et. al. plantearon el problema de seleccionar la mejor división de un nodo v en nodo izquierdo v_i y en nodo derecho v_d, mediante el criterio del Descenso de la Impureza (Información Ganada), que se introduce más adelante.

Descenso de la Impureza (Información Ganada)

Definición

El Descenso de la Impureza (Información Ganada) $\Delta Imp(v)$, obtenido a consecuencia de la división del nodo v en v_i y v_d se define como:

$$\Delta Imp(v) = Imp(v) - \left[p(v_i)Imp(v_i) + p(v_d)Imp(v_d)\right].$$

donde $p(v_i) = \frac{|v_i|}{|v|}$ *y* $p(v_d) = \frac{|v_d|}{|v|}$.

El siguiente resultado garantiza que el Descenso de la Impureza es no negativo, siempre que la función de impureza sea cóncava.

Descenso de la Impureza (Información Ganada)

Definición

El Descenso de la Impureza (Información Ganada) $\Delta Imp(v)$, obtenido a consecuencia de la división del nodo v en v_i y v_d se define como:

$$\Delta Imp(v) = Imp(v) - \left[p(v_i)Imp(v_i) + p(v_d)Imp(v_d)\right].$$

donde $p(v_i) = \frac{|v_i|}{|v|}$ *y* $p(v_d) = \frac{|v_d|}{|v|}$.

El siguiente resultado garantiza que el Descenso de la Impureza es no negativo, siempre que la función de impureza sea cóncava.

No negatividad de la impureza

Teorema

Si la función de impureza ϕ es estrictamente cóncava[a] *entonces* $\Delta Imp(v) \geq 0$ *y* $\Delta Imp(v) = 0$ *si y solo si* $p(s|v) = p(s|v_i) = p(s|v_d)$ *para* $s = 1, \ldots, r$.

[a] Sea D un conjunto convexo de $\mathbb{R}^n$ y $\alpha, \beta \in [0,1]$, con $\alpha + \beta = 1$. Una función $f : D \longrightarrow \mathbb{R}$ es estrictamente cóncava si $\forall\ x, y \in D$, $\alpha f(x) + \beta f(y) \leq f(\alpha x + \beta y)$ con igualdad si y solo si $x = y$.

Prueba:

Dado que $p(v_i) + p(v_d) = 1$, $p(v_i)p(s|v_i) + p(v_d)p(s|v_d) = p(s|v)$ y ϕ es estrictamente cóncava se tiene que:

$$0 \leq p(v_i)\text{Imp}(v_i) + p(v_d)\text{Imp}(v_d) \leq \text{Imp}(v).$$

Completar detalles en la Tarea. La prueba de la igualdad $\Delta\text{Imp}(v) = 0$ si y solo si $p(s|v) = p(s|v_i) = p(s|v_d)$ para $s = 1, \ldots, r$ queda también como parte de la Tarea.

No negatividad de la impureza

Teorema

Si la función de impureza ϕ es estrictamente cóncava[a] *entonces* $\Delta Imp(v) \geq 0$ *y* $\Delta Imp(v) = 0$ *si y solo si* $p(s|v) = p(s|v_i) = p(s|v_d)$ *para* $s = 1, \ldots, r$.

[a] Sea D un conjunto convexo de $\mathbb{R}^n$ y $\alpha, \beta \in [0,1]$, con $\alpha + \beta = 1$. Una función $f : D \longrightarrow \mathbb{R}$ es estrictamente cóncava si $\forall\ x, y \in D$, $\alpha f(x) + \beta f(y) \leq f(\alpha x + \beta y)$ con igualdad si y solo si $x = y$.

Prueba:

Dado que $p(v_i) + p(v_d) = 1$, $p(v_i)p(s|v_i) + p(v_d)p(s|v_d) = p(s|v)$ y ϕ es estrictamente cóncava se tiene que:

$$0 \leq p(v_i)\text{Imp}(v_i) + p(v_d)\text{Imp}(v_d) \leq \text{Imp}(v).$$

Completar detalles en la Tarea. La prueba de la igualdad $\Delta\text{Imp}(v) = 0$ si y solo si $p(s|v) = p(s|v_i) = p(s|v_d)$ para $s = 1, \ldots, r$ queda también como parte de la Tarea.

Criterio de impureza de Gini

Teorema

Sea la función $g : [0,1]^r \longrightarrow [0,\infty[$ *definida por* $g(x_1,\ldots,x_r) = \sum_{i\neq j} x_i x_j$, *con* $\sum_{i=1}^{r} x_i = 1$. *Entonces la función* g *tiene las siguientes propiedades:*

1. $g(x_1,\ldots,x_r) = 1 - \sum_{i=1}^{r} x_i^2$.
2. g *es estrictamente cóncava.*
3. g *es una función de impureza.*

Criterio de impureza de Gini

Prueba:

1. Es obvio que $\sum_{i \neq j}^{r} x_i x_j + \sum_{i=1}^{r} x_i^2 = \left(\sum_{i=1}^{r} x_i \right)^2 = 1$, es decir:

   $$g(x_1, \ldots, x_r) = 1 - \sum_{i=1}^{r} x_i^2.$$

2. La función x^2 es estrictamente convexa, luego $\sum_{i=1}^{r} x_i^2$ es también estrictamente convexa, entonces como $g(x_1, \ldots, x_r) = 1 - \sum_{i=1}^{r} x_i^2$ se tiene que g es estrictamente cóncava.
3. Hay que probar las 3 propiedades de una función de impureza.
   1. Tarea.
   2. Tarea (optativo).
   3. Tarea.

Criterio de impureza de Gini

Prueba:

1. Es obvio que $\sum_{i \neq j}^{r} x_i x_j + \sum_{i=1}^{r} x_i^2 = \left(\sum_{i=1}^{r} x_i \right)^2 = 1$, es decir:

   $g(x_1, \ldots, x_r) = 1 - \sum_{i=1}^{r} x_i^2$.

2. La función x^2 es estrictamente convexa, luego $\sum_{i=1}^{r} x_i^2$ es también estrictamente convexa, entonces como $g(x_1, \ldots, x_r) = 1 - \sum_{i=1}^{r} x_i^2$ se tiene que g es estrictamente cóncava.

3. Hay que probar las 3 propiedades de una función de impureza.
   1. Tarea.
   2. Tarea (optativo).
   3. Tarea.

Criterio de impureza de Gini

Prueba:

1. Es obvio que $\sum_{i \neq j}^{r} x_i x_j + \sum_{i=1}^{r} x_i^2 = \left(\sum_{i=1}^{r} x_i \right)^2 = 1$, es decir:

   $g(x_1, \ldots, x_r) = 1 - \sum_{i=1}^{r} x_i^2$.

2. La función x^2 es estrictamente convexa, luego $\sum_{i=1}^{r} x_i^2$ es también estrictamente convexa, entonces como $g(x_1, \ldots, x_r) = 1 - \sum_{i=1}^{r} x_i^2$ se tiene que g es estrictamente cóncava.
3. Hay que probar las 3 propiedades de una función de impureza.
   1. Tarea.
   2. Tarea (optativo).
   3. Tarea.

Criterio de impureza de Gini

El criterio de impureza de Gini utilizado por Breiman et al. consiste en usar la función de impureza g para calcular el Descenso de la Impureza. En este caso se pueden caracterizar los nodos puros y establecer una fórmula simplificada para el cálculo del Descenso de la Impureza, como se indica en el siguiente Teorema.

Teorema

Sea v un nodo de A_{max}; v_i y v_d sus hijos izquierdo y derecho respectivamente, $n_d = |v_d|$, $n_i = |v_i|$, $n_{sd} = |E_s \cap v_d|$, $n_{si} = |E_s \cap v_i|$, $p_i = \frac{|v_i|}{|v|}$ y $p_d = \frac{|v_d|}{|v|}$. Entonces:

1. $\Delta Imp(v) = \frac{1}{|v|^2 n_i n_d} \sum_{s=1}^{r} (n_d n_{si} - n_i n_{sd})^2$.
2. *El nodo v es puro* $\iff \exists\, h \in \{1, \ldots, r\}$ *tal que* $v \subseteq E_h$.

Criterio de impureza de Gini

El criterio de impureza de Gini utilizado por Breiman et al. consiste en usar la función de impureza g para calcular el Descenso de la Impureza. En este caso se pueden caracterizar los nodos puros y establecer una fórmula simplificada para el cálculo del Descenso de la Impureza, como se indica en el siguiente Teorema.

Teorema

Sea v un nodo de A_{max}; v_i y v_d sus hijos izquierdo y derecho respectivamente, $n_d = |v_d|$, $n_i = |v_i|$, $n_{sd} = |E_s \cap v_d|$, $n_{si} = |E_s \cap v_i|$, $p_i = \frac{|v_i|}{|v|}$ y $p_d = \frac{|v_d|}{|v|}$. Entonces:

1. $\Delta Imp(v) = \frac{1}{|v|^2 n_i n_d} \sum_{s=1}^{r} (n_d n_{si} - n_i n_{sd})^2.$
2. *El nodo v es puro* $\iff \exists\, h \in \{1, \ldots, r\}$ *tal que* $v \subseteq E_h$.

Criterio de impureza de Gini

Prueba de 1. (Tarea detallar todos los pasos y probar parte 2)

$$
\begin{aligned}
\Delta\text{Imp}(v) &= \text{Imp}(v) - [p_i\text{Imp}(v_i) + p_d\text{Imp}(v_d)] \\
&= \left(1 - \sum_{s=1}^{r} \frac{|E_s \cap v|^2}{|v|^2}\right) - p_i\left(1 - \sum_{s=1}^{r} \frac{|E_s \cap v_i|^2}{|v_i|^2}\right) \\
&\quad - p_d\left(1 - \sum_{s=1}^{r} \frac{|E_s \cap v_d|^2}{|v_d|^2}\right) \\
&= \frac{|v_i|}{|v|}\sum_{s=1}^{r} \frac{|E_s \cap v_i|^2}{|v_i|^2} + \frac{|v_d|}{|v|}\sum_{s=1}^{r} \frac{|E_s \cap v_d|^2}{|v_d|^2} - \sum_{s=1}^{r} \frac{|E_s \cap v|^2}{|v|^2} \\
&= \frac{1}{|v|}\sum_{s=1}^{r}\left(\frac{|E_s \cap v_i|^2}{|v_i|} + \frac{|E_s \cap v_d|^2}{|v_d|} - \frac{|E_s \cap v|^2}{|v|}\right) \\
&= \frac{1}{|v|}\sum_{s=1}^{r}\left(\frac{n_{si}^2}{n_i} + \frac{n_{sd}^2}{n_d} - \frac{n_{si}^2 + n_{sd}^2 + 2n_{si}n_{sd}}{n_i + n_d}\right) \\
&= \frac{1}{|v|^2 n_i n_d}\sum_{s=1}^{r}(n_{si}n_d - n_{sd}n_i)^2 .
\end{aligned}
$$

División binaria óptima

Definición

Una división $(\widetilde{v}_i, \widetilde{v}_d)$ del nodo v es óptima en el sentido de Gini, si se cumple:

$$(\widetilde{v}_i, \widetilde{v}_d) = \underset{(v_i, v_d)}{\text{máx}} \frac{1}{n_i n_d} \sum_{s=1}^{r} (n_{si} n_d - n_{sd} n_i)^2$$

donde (v_i, v_d) es una división binaria de v. Es decir, es una división que maximiza el Descenso de la Impureza, i.e. maximiza la Información Ganada.

Algoritmo para determinar una división binaria óptima

Algoritmo:

Para un nodo v hacer 1) y 2).

1. *Para $j = 1, \ldots, p$ determinar,*
   1. *Todas las divisiones binarias correspondiente al predictor y^j.*
   2. *La división binaria óptima d_j correspondiente a y^j. Es decir la división binaria de máximo Descenso de la Impureza.*
2. *Hallar la mejor división binaria entre $d_1, \ldots, d_p$.*

El cálculo, en el item 1. a) del algoritmo anterior, de todas las divisiones binarias de un nodo para el caso de predictores categóricos, se simplifica cuando el número de clases a priori es 2. El siguiente Teorema indica como lograr dicha simplificación.

Algoritmo para determinar una división binaria óptima

Algoritmo:

Para un nodo v hacer 1) y 2).

1. *Para $j = 1, \ldots, p$ determinar,*
   1. *Todas las divisiones binarias correspondiente al predictor y^j.*
   2. *La división binaria óptima d_j correspondiente a y^j. Es decir la división binaria de máximo Descenso de la Impureza.*
2. *Hallar la mejor división binaria entre $d_1, \ldots, d_p$.*

El cálculo, en el item 1. a) del algoritmo anterior, de todas las divisiones binarias de un nodo para el caso de predictores categóricos, se simplifica cuando el número de clases a priori es 2. El siguiente Teorema indica como lograr dicha simplificación.

Cuando el número de clases a priori es 2

Definición

Sea $B = \{b_1, \ldots, b_m\}$ *las modalidades del predictor categórico nominal* Y*. Se define la probabilidad de que un individuo del grupo a priori* E_j *pertenezca al nodo* v *y posea la modalidad* b_s*, como:*

$$p(j|y = b_s) = \frac{\pi_j N_{j,s}(v)}{\pi_1 N_{1,s}(v) + \pi_2 N_{2,s}(v)}$$

donde $\pi_j = \frac{|E_j|}{n}$, $j = 1, 2$*;* I_s *es el conjunto de individuos que poseen la modalidad* s *de la variable* Y *y* $N_{j,s}(v) = |v \cap E_j \cap I_s|$.

Cuando el número de clases a priori es 2

Teorema

Si se ordenan las modalidades $B = \{b_1, \ldots, b_m\}$ *en* $B = \{b_{k_1}, \ldots, b_{k_m}\}$ *según el criterio,*

$$p(1|x = b_{k_1}) \leq \cdots \leq p(1|x = b_{k_m}).$$

Entonces uno de los $m - 1$ *conjuntos de modalidades* $\{b_{k_1}\}$, $\{b_{k_1}, b_{k_2}\}$, $\{b_{k_1}, \ldots, b_{k_{m-1}}\}$, *determina una división binaria óptima sobre el predictor categórico nominal* Y *en el nodo* v *en el sentido de Gini*[a].

[a] La demostración de este resultado se halla en Breiman et al. (al final del capítulo 9).

Selección de un subárbol óptimo

Cada nodo y cada subárbol de $A_{\max}$ tienen un costo de mala clasificación (o riesgo) asociado, que refleja su capacidad predictiva que debe cumplir lo siguiente:

Definición (Costo de mala clasificación)

Sea $c(i|j)$ el costo de clasificar un objeto en la clase a priori "i" dado que pertenece a la clase a priori "j". Este costo debe satisfacer: $c(i|j) \geq 0$ para todo $i \neq j$ y $c(i|i) = 0$ para todo $i = 1, \ldots, r$.

Un buen subárbol de $A_{\max}$ debe producir reglas de predicción fácilmente interpretables y eficientes. Es decir, se quiere un subárbol de complejidad mínima y costo también mínimo. Como no se pueden lograr ambos objetivos simultáneamente, entonces se utiliza como criterio de calidad, un compromiso entre los dos objetivos, llamado costo-complejidad.

Selección de un subárbol óptimo

Cada nodo y cada subárbol de $A_{\max}$ tienen un costo de mala clasificación (o riesgo) asociado, que refleja su capacidad predictiva que debe cumplir lo siguiente:

Definición (Costo de mala clasificación)

Sea $c(i|j)$ el costo de clasificar un objeto en la clase a priori "i" dado que pertenece a la clase a priori "j". Este costo debe satisfacer: $c(i|j) \geq 0$ para todo $i \neq j$ y $c(i|i) = 0$ para todo $i = 1, \ldots, r$.

Un buen subárbol de $A_{\max}$ debe producir reglas de predicción fácilmente interpretables y eficientes. Es decir, se quiere un subárbol de complejidad mínima y costo también mínimo. Como no se pueden lograr ambos objetivos simultáneamente, entonces se utiliza como criterio de calidad, un compromiso entre los dos objetivos, llamado costo-complejidad.

Selección de un subárbol óptimo

Cada nodo y cada subárbol de $A_{\max}$ tienen un costo de mala clasificación (o riesgo) asociado, que refleja su capacidad predictiva que debe cumplir lo siguiente:

Definición (Costo de mala clasificación)

Sea $c(i|j)$ el costo de clasificar un objeto en la clase a priori "i" dado que pertenece a la clase a priori "j". Este costo debe satisfacer: $c(i|j) \geq 0$ para todo $i \neq j$ y $c(i|i) = 0$ para todo $i = 1, \ldots, r$.

Un buen subárbol de $A_{\max}$ debe producir reglas de predicción fácilmente interpretables y eficientes. Es decir, se quiere un subárbol de complejidad mínima y costo también mínimo. Como no se pueden lograr ambos objetivos simultáneamente, entonces se utiliza como criterio de calidad, un compromiso entre los dos objetivos, llamado costo-complejidad.

Costo-Complejidad

Definición (Costo esperado estimado)

Si un objeto que es seleccionado al azar, cae en el nodo v_t del árbol A_{max} y es clasificado en la clase a priori "i", el costo esperado (estimado) de mala clasificación, dado el nodo v_t, se define como:

$$c(v_t) = c(t) = \frac{1}{|v_t|} \min_{i=1,\ldots,r} \sum_{j=1}^{r} c(i|j)|E_j \cap v_t|.$$

Si se asumen costos unitarios[a]*, entonces:*

$$c(t) = 1 - \frac{1}{|v_t|} \max_{j=1,\ldots,r} |E_j \cap v_t|.$$

[a] Es decir, $c(i|j) = 1$ para todo $i \neq j$.

Ejercicio

1. Pruebe que $p(v_t)c(v_t) \geq p(v_i)c(v_i) + p(v_d)c(v_d)$.
2. Falso o Verdadero que: La igualdad puede ocurrir aún cuando los nodos hijos v_i y v_d contengan una mezcla de objetos de distintas clases a priori.

Costo-Complejidad

Definición (Costo-Complejidad)

Dado $\alpha \in \mathbb{R}$.

1. *El Costo-Complejidad del nodo u_t de A_{max} de define como:*

$$c_\alpha(u_t) = p(u_t)c(u_t) + \alpha.$$

2. *El Costo-Complejidad de una rama A_v de A_{max} se define como:*

$$c_\alpha(A_v) = \sum_{u_t \in \widetilde{A}_v} c_\alpha(u_t),$$

es decir,

$$c_\alpha(A_v) = \alpha|\widetilde{A}_v| + \sum_{u_t \in \widetilde{A}_v} p(u_t)c(u_t),$$

donde el conjunto de nodos terminales de A se denota $\widetilde{A}$.

Construcción de una cadena de subárboles α-óptimos

La estrategia para determinar un subárbol óptimo es construir una cadena de subárboles de $A_{\max}$ óptimos (en el sentido de Costo-Complejidad) y seleccionar entre ellos el que minimiza el porcentaje de objetos mal clasificados.

Definición

1. $C(u_t) = p(u_t)c(u_t)$ *se define como el costo del nodo* u_t.
2. $C(A_v) = \sum_{u_t \in \widetilde{A}_v} p(u_t)c(u_t)$ *se define como el costo de la rama* A_v.
3. *Un nodo* v *es preferido a la rama* A_v *si:*

$$c_\alpha(v) \leq c_\alpha(A_v).$$

Construcción de una cadena de subárboles α-óptimos

La estrategia para determinar un subárbol óptimo es construir una cadena de subárboles de $A_{\max}$ óptimos (en el sentido de Costo-Complejidad) y seleccionar entre ellos el que minimiza el porcentaje de objetos mal clasificados.

Definición

1. $C(u_t) = p(u_t)c(u_t)$ *se define como el costo del nodo* u_t.
2. $C(A_v) = \sum_{u_t \in \widetilde{A}_v} p(u_t)c(u_t)$ *se define como el costo de la rama* A_v.
3. *Un nodo* v *es preferido a la rama* A_v *si:*

$$c_\alpha(v) \leq c_\alpha(A_v).$$

Construcción de una cadena de subárboles α-óptimos

Teorema

Un nodo v es preferido a la rama A_v si:

$$\alpha \geq \frac{C(v) - C(A_v)}{|\widetilde{A}_v| - 1}.$$

Prueba: *Tarea.*

- Por lo tanto para v fijo, el valor mínimo de α tal que se prefiere podar $A_{\max}$ en el nodo v, es $\alpha = \frac{C(v) - C(A_v)}{|\widetilde{A}_v| - 1}$.
- A partir de esta observación es natural definir los nodos terminales del primer subárbol A_1 como sigue:

Construcción de una cadena de subárboles α-óptimos

Teorema

Un nodo v es preferido a la rama A_v si:

$$\alpha \geq \frac{C(v) - C(A_v)}{|\widetilde{A}_v| - 1}.$$

Prueba: *Tarea.*

- Por lo tanto para v fijo, el valor mínimo de α tal que se prefiere podar $A_{\max}$ en el nodo v, es $\alpha = \frac{C(v) - C(A_v)}{|\widetilde{A}_v| - 1}$.
- A partir de esta observación es natural definir los nodos terminales del primer subárbol A_1 como sigue:

Construcción de una cadena de subárboles α-óptimos

Teorema

Un nodo v es preferido a la rama A_v si:

$$\alpha \geq \frac{C(v) - C(A_v)}{|\widetilde{A}_v| - 1}.$$

Prueba: *Tarea.*

- Por lo tanto para v fijo, el valor mínimo de α tal que se prefiere podar $A_{\max}$ en el nodo v, es $\alpha = \frac{C(v) - C(A_v)}{|\widetilde{A}_v| - 1}$.
- A partir de esta observación es natural definir los nodos terminales del primer subárbol A_1 como sigue:

Construcción de una cadena de subárboles α-óptimos

Definición (Primer subárbol de la cadena)

- *Sea* $A_0 = A_{max}$ *y* $A_{0,v}$ *la rama de* A_0 *con raíz* v. *Se define la función* $g(v, A_0) = \frac{C(v) - C(A_{0,v})}{|\widetilde{A}_{0,v}| - 1}$ *para todo* $v \in A_0 - \widetilde{A}_0$.
- *Un nodo* v *se dice terminal del subárbol en construcción* A_1 ($v \in \widetilde{A}_1$) *si* $g(v, A_0) = \alpha_1$ *donde:*

$$\alpha_1 = \ min \ \{g(u, A_0) \text{ para } u \in A_0 - \widetilde{A}_0\}.$$

Los nodos terminales del subárbol siguiente A_2 se determinan igual, donde A_1 ocupa el lugar de A_0. El proceso se repite hasta obtener en $A_k = \{v_1\}$ donde v_1 es la raíz de A_0. El proceso de construcción está garantizado por el Teorema de Existencia Unicidad y el Teorema de sobre la Cadena de Subárboles que se presentan más adelante.

Construcción de una cadena de subárboles α-óptimos

Definición (Primer subárbol de la cadena)

- *Sea* $A_0 = A_{max}$ *y* $A_{0,v}$ *la rama de* A_0 *con raíz* v. *Se define la función* $g(v, A_0) = \frac{C(v) - C(A_{0,v})}{|\widetilde{A}_{0,v}| - 1}$ *para todo* $v \in A_0 - \widetilde{A}_0$.
- *Un nodo* v *se dice terminal del subárbol en construcción* A_1 ($v \in \widetilde{A}_1$) *si* $g(v, A_0) = \alpha_1$ *donde:*

$$\alpha_1 = \ min \ \{g(u, A_0) \ para \ u \in A_0 - \widetilde{A}_0\}.$$

Los nodos terminales del subárbol siguiente A_2 se determinan igual, donde A_1 ocupa el lugar de A_0. El proceso se repite hasta obtener en $A_k = \{v_1\}$ donde v_1 es la raíz de A_0. El proceso de construcción está garantizado por el Teorema de Existencia Unicidad y el Teorema de sobre la Cadena de Subárboles que se presentan más adelante.

Construcción de una cadena de subárboles α-óptimos

Definición (Subárbol α-óptimo)

Sea $\alpha > 0$, un subárbol $A(\alpha)$ de un árbol binario A, ambos con la misma raíz v_1, es α-óptimo si:

1. *$A(\alpha)$ tiene α-costo-complejidad mínimo, es decir:*

$$c_\alpha(A(\alpha)) = min\{c_\alpha(X) \text{ para } X < A \text{ y } X \text{ tiene raíz } v_1\},$$

   donde $X < A$ significa que X es subárbol de A pero no es igual a A.

2. *No existe un subárbol de $A(\alpha)$ que satisface 1) excepto $A(\alpha)$ mismo, en este sentido se dice que $A(\alpha)$ es minimal.*

Construcción de una cadena de subárboles α-óptimos

Teorema (Existencia y Unicidad)

Sea $\alpha > 0$ y A un árbol binario, entonces:

1. *Existe $A(\alpha)$ y es único.*
2. *$A(\alpha) = \{t \in A$ tal que $\forall\ s \in ant(t, A),\ g(s, A) > \alpha\}$, donde $ant(t, A)$ es el conjunto de antecesores de t en A.*
3. *$g(t, A(\alpha)) > g(t, A)$ si $A_t(\alpha) < A_t$.*

Prueba: *Se Omite.*

Construcción de una cadena de subárboles α-óptimos

Teorema (Cadena de subárboles)

Existe una cadena de subárboles $A_0 > A_1 > \cdots > A_k$ *y* $\alpha_1, \alpha_2, \ldots, \alpha_k$ *tales que:*

1. $\alpha_{i+1} = \min_{u \in A_i - \widetilde{A}_i} g(u, A_i)$ *con* $i = 0, \ldots, k-1$.
2. *Se define* $A_{i+1} := \{u \in A_i \text{ tal que } \forall\, x \in ant(u, A_0),\ g(x, A_i) > \alpha_{i+1}\}$.
3. $0 \leq \alpha_1 < \alpha_2 < \cdots < \alpha_k < +\infty$ *y* $A_k = \{v_1\}$.
4. $$A_0(\alpha) = \begin{cases} A_0 & si \quad \alpha \leq \alpha_1 \\ A_i & si \quad \alpha_i \leq \alpha < \alpha_{i+1};\ i = 1, \ldots, k-1 \\ A_k & si \quad \alpha \geq \alpha_k \end{cases}$$
5. $A_i = \{t \in A_0 \text{ tal que } \forall s \in ant(t, A_0) \text{ se tiene que } g_{k-1}(s) > \alpha_i\}$, *donde la función* g_{k-1} *se define recursivamente:* $g_0(t) = g(t, A_0)$ *para todo* $t \in A_0 - \widetilde{A}_0$ *y para* $i = 1, \ldots, k-1$ *se define como:*

$$g_i(t) = \begin{cases} g(t, A_i) & si \quad t \in A_i - \widetilde{A}_i \\ g_{i-1}(t) & si \quad t \in suc(\widetilde{A}_i) - \widetilde{A}_0 \end{cases}$$

Esbozo del Algoritmo CART

Salida del algoritmo: $p_1, \ldots, p_{k-1}$.

For $[\ i = 1, \ldots, k-1$

$\quad p_i = 0$

$\quad$ *For* $[j = 1, \ldots, n$

$\quad\quad h = 1$ *(inicia en la raíz)*

$\quad\quad$ *While* $[g_{k-1}(h) > \alpha_i$

$\quad\quad\quad$ *If [j pertenece al hijo izquierdo de h entonces:* $h = Izq(h)$.

$\quad\quad\quad$ *Si no:* $h = Der(h)]$

$\quad\quad$]

$\quad\quad$ *If [j pertenece al grupo a priori r y* $r \neq h$ *entonces* $p_i = p_i + 1]$

$\quad$]

$\quad p_i = \frac{p_i}{n}$

$\quad \ldots\ldots\ldots$

]

Gracias ...

Gracias ....