Aprendizaje Estadístico Supervisado

Natalia da Silva

2024

Ya vimos

- Árboles de decisión
- Métodos de agregación/ensembles (Bagging y Random Forest)

Veremos

- Algunas extensiones de árboles
- Extensiones de Random Forest
- Otro método de agregación, Boosting

Recap Árboles de decisión

- Fifty Years of Classification and Regression Trees (Loh 2014)
- Los modelos basados en árboles consisten en una secuencia de particiones anidadas que dividen el espacio de los predictores donde en cada partición se usa un modelo para predecir la respuesta.
- En cada paso del árbol se establecen decisiones de tipo si pasa A entonces B.
- Los árboles de decisión se pueden agrupar por el número de variables predictoras utilizadas en cada partición de nodos.
- Los árboles que usan sólo una variable en cada partición de los nodos producen particiones paralelas a los ejes, mientras que cuando usan muchas variables para la partición producen particiones oblicuas.

Recap Árboles de decisión

- En teoría, las regiones que define un árbol podrían ser de cualquier forma.
- Nosotros seleccionamos CART (para regresión y clasificación) donde se divide el espacio de los predictores en rectángulos en altas dimensiones.

Recap Árboles de decisión

- Un árbol puede ser visto como un conjunto de reglas de dicisión que definen particiones del espacio de las predictoras. El valor esperado de la variable de respuesta Y puede ser definido como:

$$E(Y/X = x) = \sum_{s=1}^{S} c_s I_{R_s}(x)$$

- Donde R_s representa una partición en el espacio de las predictoras tal que $R_s \in R$ con $R = \cup_{i=1}^{S} R_i$ y $R_s \cap R_i = \emptyset$.

Si $x \in R_s$ entonces el valor predicho para Y es c_s.

$$I_{R_s}(x) = \begin{cases} 1 & \text{si } x \in R_s \\ 0 & \text{si } x \notin R_s \end{cases}$$

c_s es calculado diferente si el problema es de regresión o de clasificación

Recap Árboles de decisión

Para problemas de clasificación $Y_i \in \{1, 2 \dots K\}$ denota la clase para cada observación. El valor esperado para Y cuando $X_i \in R_s$ es la clase más frecuente in la partición R_s, i.e.

$$c_s = \arg\max_k \left\{ \frac{\#(Y = k)}{\#R_s} \right\} \quad X_i \in R_s$$

Para problemas de regresión el valor predicho es:

$$c_s = \frac{1}{\#R_s} \sum_{i/X_i \in R_s} Y_i$$

Recap Árboles de decisión

Finalmente para un conjunto de datos dato $\{Y_i, X_{1i,} X_{2i} \ldots X_{pi}\}_{i=1}^{n}$ el valor predicho para la respuesta Y puede ser definido como $\hat{f}(x) = \sum_{s=1}^{S} \hat{c_s} I_{\hat{R_s}}(x)$

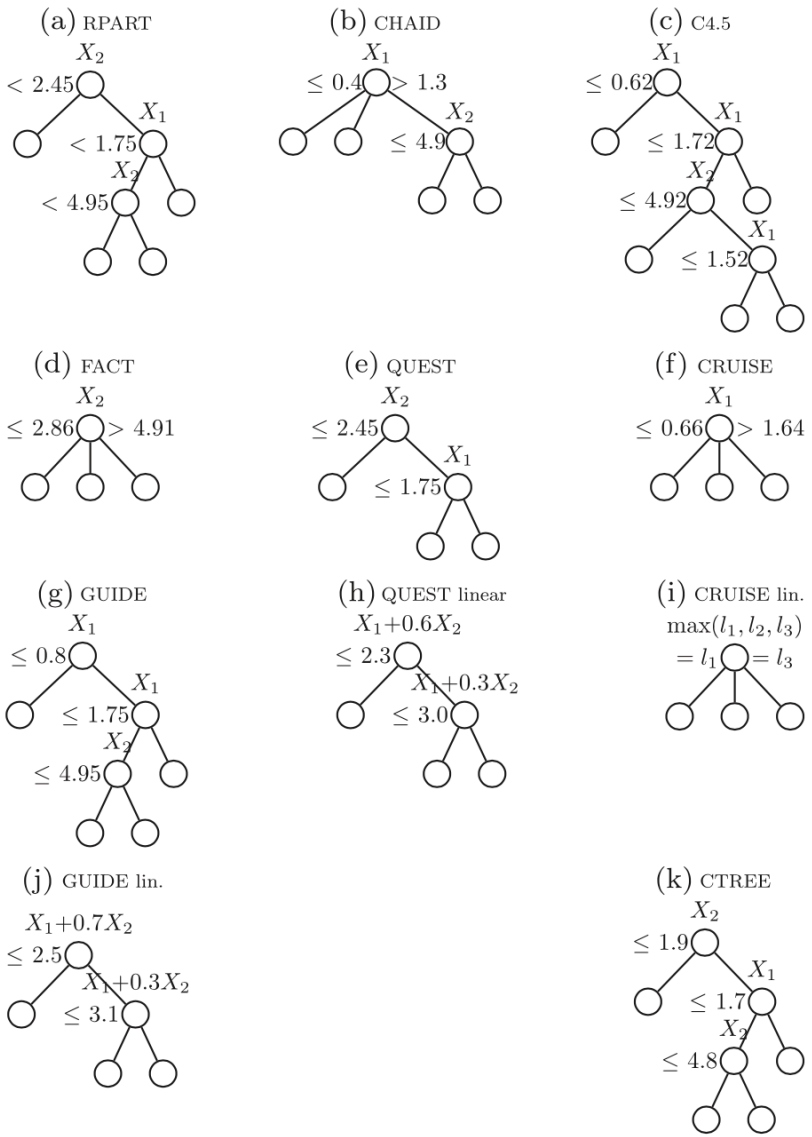
Diferente para regresión y clasificación:

- Regresión: Promedio de la respuesta en la hoja
- Clasificacón: Clase más frecuente de la respuesta en la hoja

Variantes de CART

- CHAID, C4.5, FACT, QUEST, CRUISE, GUIDE, CTREE, PPtree, etc.
- Principales diferencias entre los distintos métodos son la forma en que se parte el nodo.
- Algunos usan métodos de kernel, vecino más cercano, particiones lineales en un subconjunto de variables seleccionadas.
- Pueden ser árboles binarios o con particiones múltiples.
- Los árboles que usan una sola variable en la partición del nodo generan particiones paralelas a los ejes, cundo se usan más variables las particiones tienden a ser oblicuas.

Variantes


(a) RPART
X_2
< 2.45
X_1
< 1.75
X_2
< 4.95
(b) CHAID
X_1
≤ 0.4
> 1.3
X_2
≤ 4.9
(c) C4.5
X_1
≤ 0.62
X_1
≤ 1.72
X_2
≤ 4.92
X_1
≤ 1.52
(d) FACT
X_2
≤ 2.86
> 4.91
(e) QUEST
X_2
≤ 2.45
X_1
≤ 1.75
(f) CRUISE
X_1
≤ 0.66
> 1.64
(g) GUIDE
X_1
≤ 0.8
X_1
≤ 1.75
X_2
≤ 4.95
(h) QUEST linear
$X_1+0.6X_2$
≤ 2.3
$X_1+0.3X_2$
≤ 3.0
(i) CRUISE lin.
$\max(l_1, l_2, l_3)$
$= l_1$
$= l_3$
(j) GUIDE lin.
$X_1+0.7X_2$
≤ 2.5
$X_1+0.3X_2$
≤ 3.1
(k) CTREE
X_2
≤ 1.9
X_1
≤ 1.7
X_2
≤ 4.8

Variantes

- Chaid (CHi-squared Automatic Interaction Detector aproximación) similar a regresión stepwise para el criterio de selección. Para buscar una variable X para dividir un nodo, este último se divide inicialmente en dos o más nodos hijos, cuyo número depende del tipo de variable. Disponible sólo en software comercial.
- C4.5 para clasificación, si X tiene m valores distintos en un nodo las particiones son en m nodos hijos, un nodo por hijo. Si X es ordenada los nodos se parten en la forma habitual en dos nodos ($\mathrm{X} < \mathrm{c}$) usa una medida de impureza basada en entropía, ratio de ganancia.

Variantes

- C4.5 es que inducen sesgo en la selección de variables. Si X_1 y X_2 con n_1 y n_2 distintos valores es más probable seleccionar la que tiene más valores distintos.
- Hay algunos algoritmos que intentan superar este problema, CRUISE, QUEST, CTREE …..

Variantes

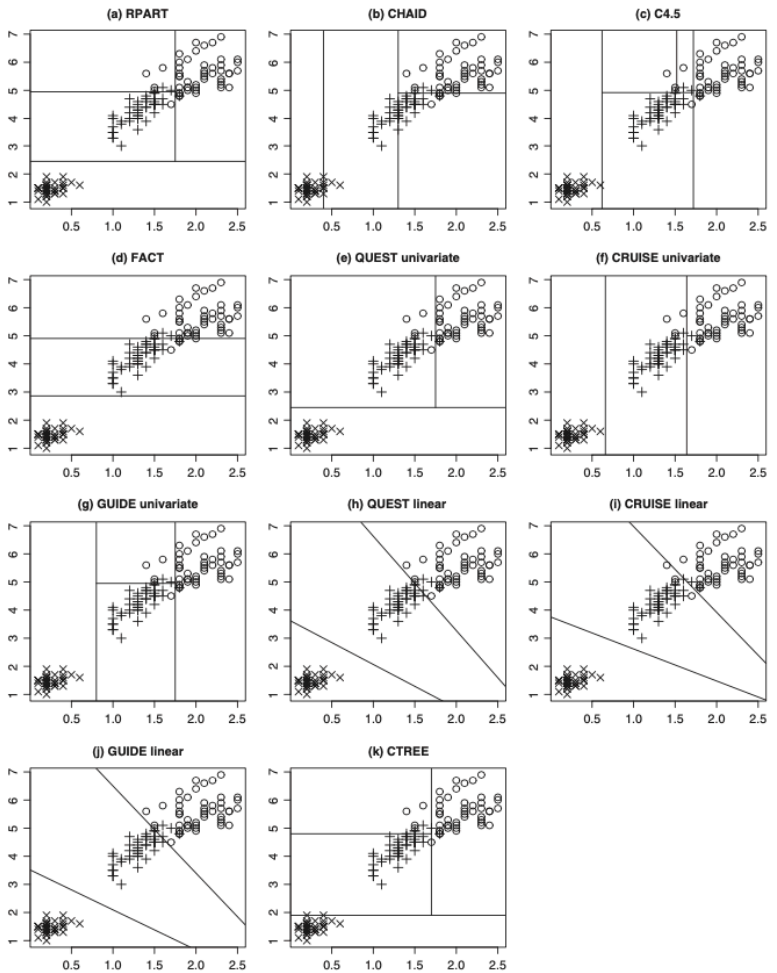


Figure 3. *Plots of petal length versus petal width with classification tree partitions; Setosa, Versicolour, and Virginica are marked by triangles, circles, and crosses, respectively.*

https://pages.stat.wisc.edu/~loh/research.html

Projection pursuit Trees

Les voy a contar sobre PPtree que combina concepto de árboles con optimización de índices de proyección Lee et al. (2013)

Projection Pursuit

- PP es una técnica estadística para exploración para encontrar proyecciones de datos que revelen estructuras **interesantes** en algún sentido.
- Los algoritmos de PP buscan proyecciones de bajas dimensiones optimizando un índice de proyección que mide si la proyección es útil/interesante en algún sentido.
- Pocos PP índices incorporan información de las clases para el cálculo.
- Lee et. al. proponen un índice derivado del análisis disciminante lineal útil para exploración en clasificación supervisada.

PP_LDA

$$I_{LDA}(A) = \begin{cases} 1 - \frac{|A^T W A|}{|A^T (W+B) A|} & \text{for} |A^T (W+B) A| \neq 0 \\ 0 & \text{for} |A^T (W+B) A| = 0 \end{cases}$$

donde $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_q]$ es una matriz $\mathbf{p} \times \mathbf{q}$ y define una proyección ortonormal de $\mathbf{p}$-dimensiones a un subespacio $\mathbf{q}$-dimensional,
$\mathbf{B} = \sum_{g=1}^{G} \mathbf{n}_g (\bar{\mathbf{x}}_g - \bar{\mathbf{x}})(\bar{\mathbf{x}}_g - \bar{\mathbf{x}})^T$ es La suma de cuadrados entre grupos $\mathbf{p} \times \mathbf{p}$.
$\mathbf{W} = \sum_{g=1}^{G} \sum_{i \in H_g} (\mathbf{x}_i - \bar{\mathbf{x}}_g)(\mathbf{x}_i - \bar{\mathbf{x}}_g)^T$ es la suma de cuadrados al interor de los grupos $\mathbf{p} \times \mathbf{p}$ donde $\mathbf{H}_g = \{i | y_i = g, i = 1, \ldots, n\}$, $\bar{\mathbf{x}}_g$ es el vector de medias de los grupos y $\bar{\mathbf{x}}$ vector de medias global. Si el índice LDA tiene valores altos hay una gran diferencia entre clases y no de lo contrario.

PP_PDA

Cuando las variables est'an altamente correlacionadas $|\mathbf{A}^{\mathbf{T}}(\mathbf{W} + \mathbf{B})\mathbf{A}|$ en $\mathbb{I}_{\mathbf{LDA}}$ es cercano a cero y no funciona bien.

$$I_{PDA}(A, \lambda) = 1 - \frac{|A^T W_{PDA} A|}{|A^T (W_{PDA} + B) A|}$$

misma notación que en $\mathbb{I}_{\mathbf{LDA}}$, agregando que $\lambda \in [0, 1)$ es un parámetro de contracción y usamos una suma de cuadrados al interior de grupos diferente, $\mathbf{W}_{\mathbf{PDA}}(\lambda) = \mathbf{diag}(\mathbf{W}) + (1 - \lambda)\mathbf{offdiag}(\mathbf{W})$.

Projection pursuit

- PP quiere encontrar todas las proyecciones interesantes.
- No solamente un óptimo global ya que los optimos locales pueden revelear estucturas interesantes.

PCA como caso particular de PP

- PP selecciono proyecciones de bajas dimensiones en mis datos multivariados optimizando algún indice de proyección ``interesante''.
- PCA se puede ver como un caso particular de PP donde el índice de proyección es la varianza de los datos que se maximiza sobre todas las proyecciones.

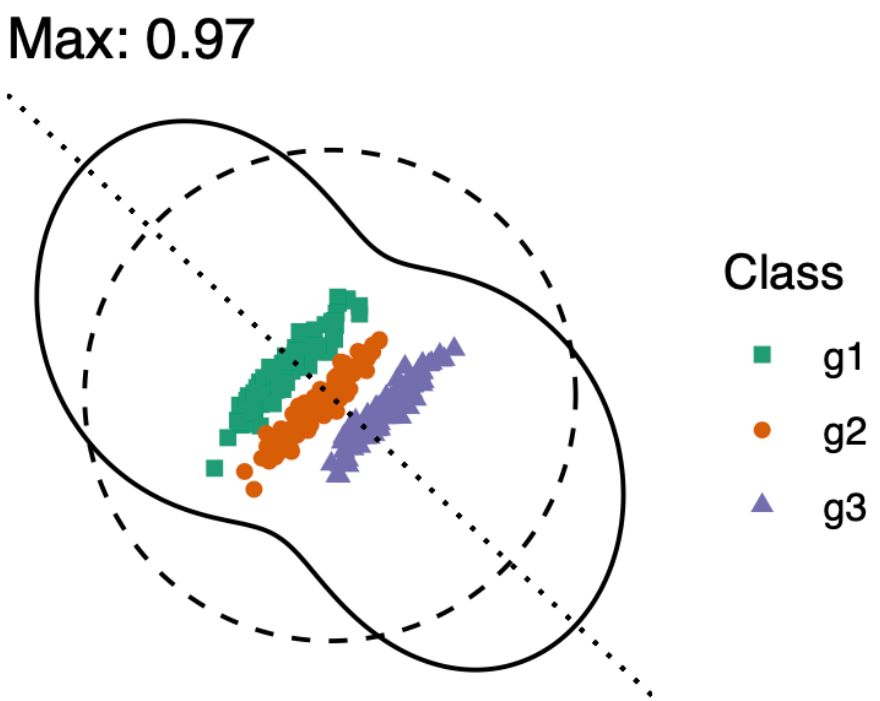
Optimización e Φ_{LDA} y Φ_{PDA}

Para los dos índices de proyección seleccionados se puede encontrar el óptimo teórico.

- La proyección q-dimensional que maximiza Φ_{LDA}, son los primeros q vectores propios de $(\mathbf{W} + \mathbf{B})^{-1}\,\mathbf{B}$
- La proyección q-dimensional que maximiza Φ_{LDA}, on los primeros q vectores propios de $(\mathbf{W}_{\mathrm{PDA}} + \mathbf{B})^{-1}\,\mathbf{B}$

Huber plot, datos simulados

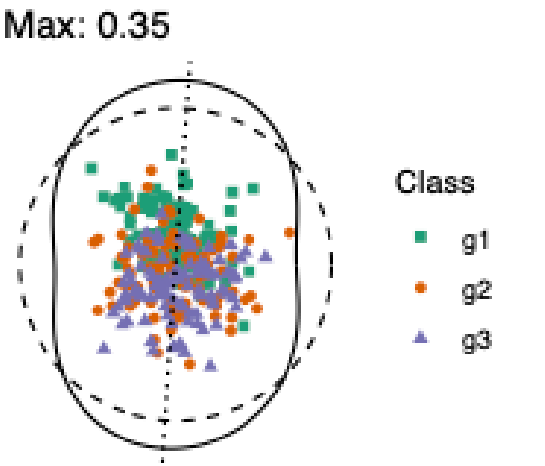
Huber plot (huber1990data?)

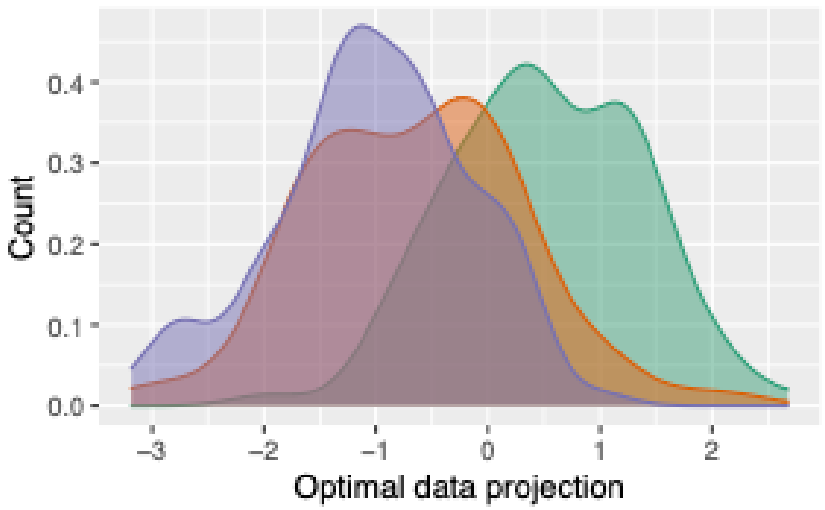


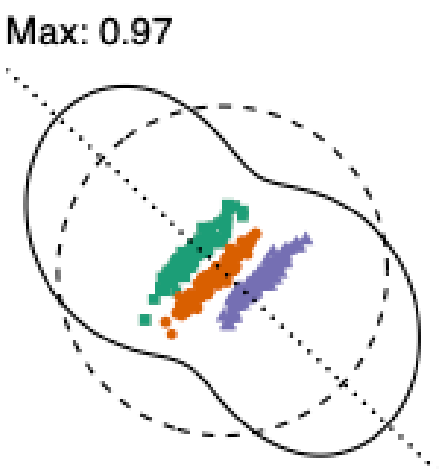
- Muestra el índice PP en todas las posibles direcciones en 2D
- Los índices se calculan usando las proyecciones para $(\cos\theta, \sin\theta)$
- $\theta = 1^\circ, \ldots, 180^\circ$
- Para cada proyección, cálculo el índice en los datos proyectados (línea sólida)
- Círculo punteado es de referencia, mediana de todos los valores del índice. Línea punteada corresponde al máximo índice.

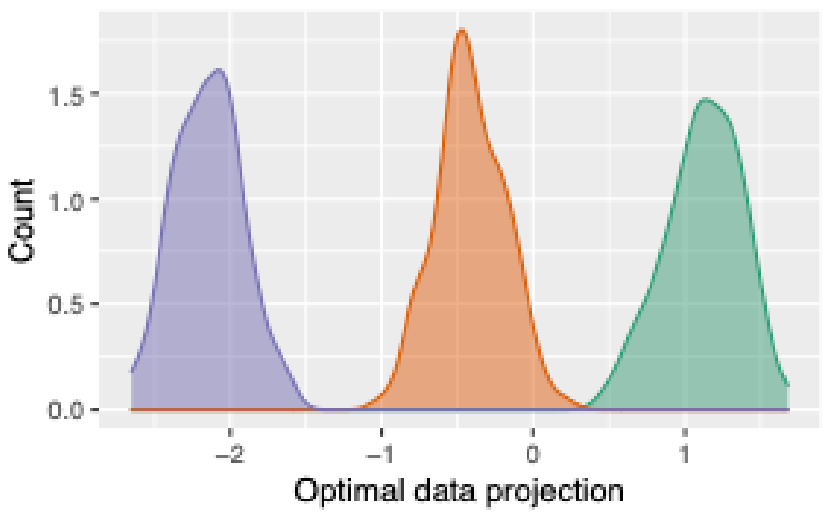
LDA, proyección 1-D

Huber Plot





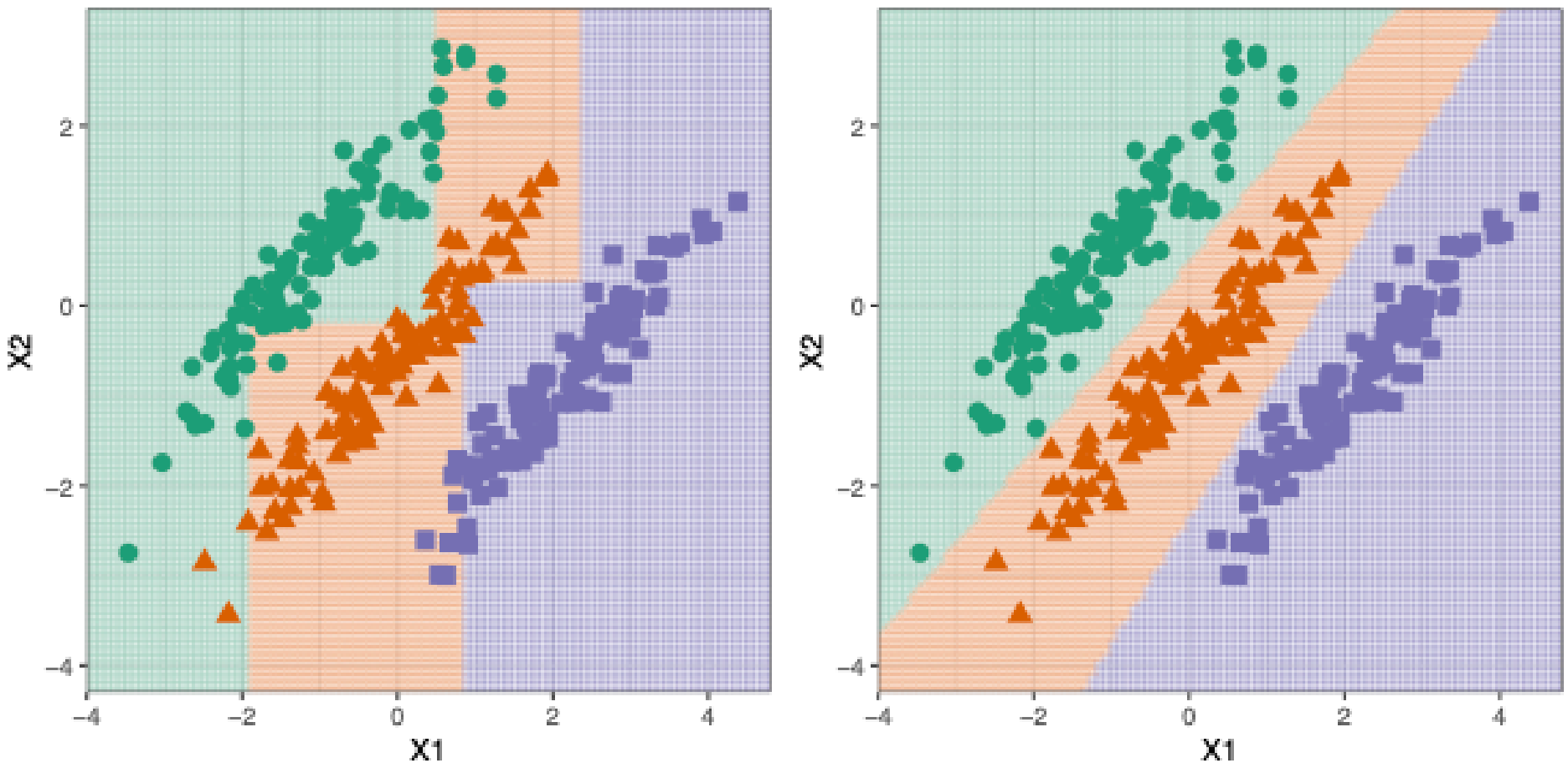




PPtree

- **PPtree** son árboles de clasificación basados en proyecciones.
- Combina métodos de árboles con reducción de dimensionalidad basada en proyecciones.
- Las particiones en **PPtree** se basan en combinaciones lineales de variables por lo que toma en cuenta la correlación entre las variables para separar las clases.

PPtree



PPtree

- Trata los datos siempre como un problema de dos clases.
- Cuando las clases son más de dos el algoritmo usa dos pasos de proyección optimizando un índice de proyección en cada partición del nodo.
- Los coeficientes de proyección en cada nodo representan la importancia de la variables, útiles para explorar como las clases son separadas en cada árbol.

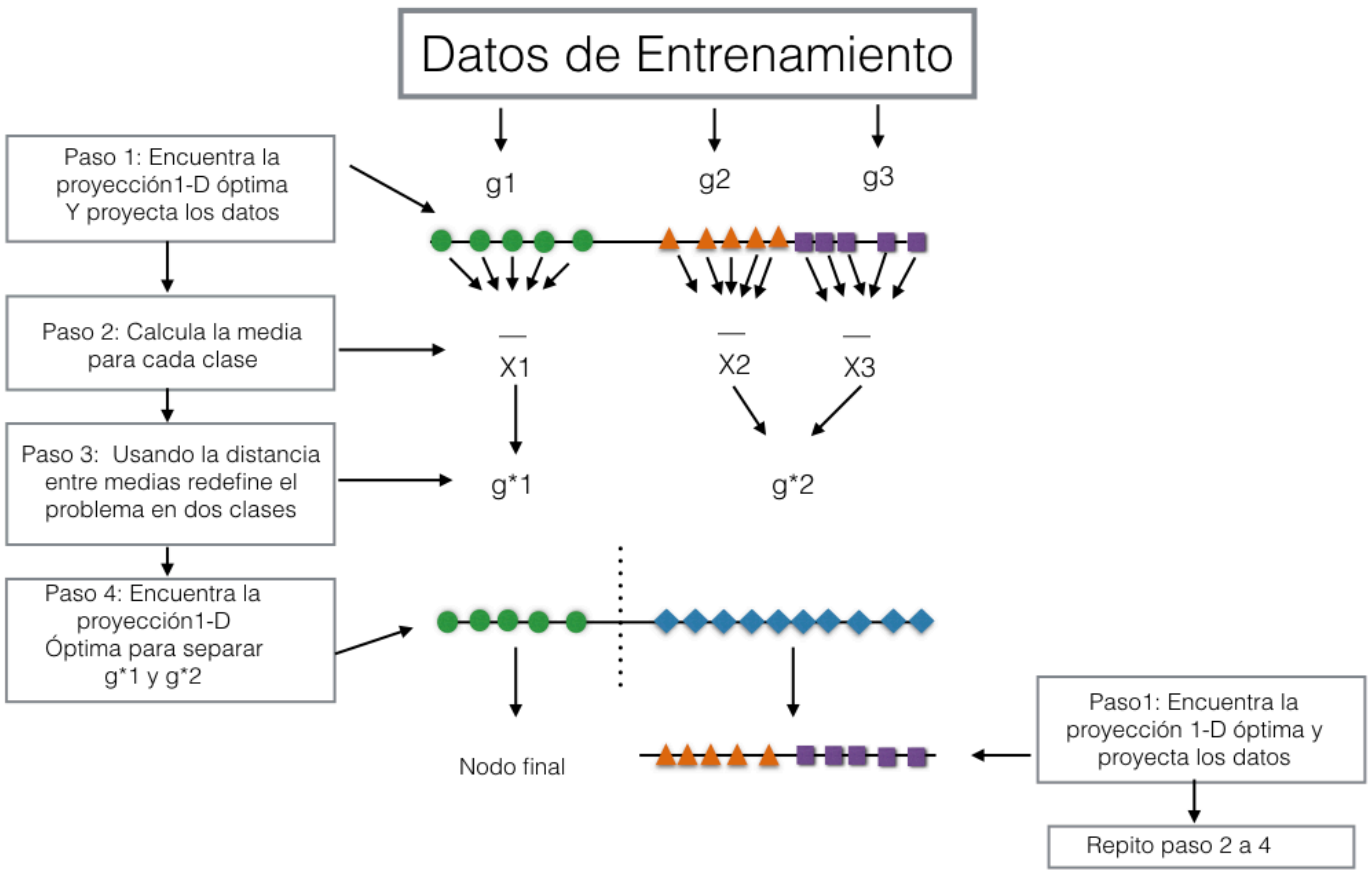
Algoritmo de PPtree

1. Optimiza un indice de proyección para encontrar una proyección $1-D$ óptima, $\mathbf{a_1}^*$, para separarar todas las clases obteniendo los datos proyectados $z_i = \mathbf{a_1}^{*T}\mathbf{x_i}$ para todo i.
2. En los datos proyectados, redefine el problema en uno de dos clases usando la distancia de la media entre clases asigna una nueva etiqueta, g_1^* o g_2^* para cada observación, generando una nueva variable y_i^* para la clase. Notar que esto requiere juntar dos o más grupos en uno, tal que los nuevos grupos g_1^* y g_2^* posiblemente tnengan más de una clase original.
3. Encuentra una proyección $1-D$ óptima $\mathbf{a_1}^{**}$, usando $\{(\mathbf{x_i}, y_i^*)\}_{i=1}^n$ para separar un problema de dos clases g_1^* y g_2^*. La mejor separación de g_1^* y g_2^* es determinada en este paso proporcionando la regla de dicisiónn para el nodo.
4. si $\mathbf{a_1}^{**T}\bar{\mathbf{x}}_{g_1^*} < c$ entonces asigna g_1^* al nodo izquierdo y g_2^* al nodo derecho,

donde $\bar{\mathbf{x}}_{g_1^*}$ es el vector de medias g_1^*. El valor de corte en los datos proyectados se calcula como $c = 0.5 * m_1 + 0.5 * m_2$ (defecto), donde m_1 y m_2 son las medias de grupos de los datos proyectados. Hay 8 reglas para calcular c.

5. Para cada grupo todos los pasos previos son calculados y repetidos hasta que g_1^* y g_2^* tienen solo una clase de la clase original. La profundidad del árbole es como mucho la cantidad de clases.

Ilustración del algoritmo



La profundidad del PPtree es como mucho el número de clases.

PPtree, 8 reglas de corte

$c = \frac{1}{2}\bar{\mathbf{x}}_1 + \frac{1}{2}\bar{\mathbf{x}}_2$
$c = \frac{n_1}{n_1+n_2}\bar{\mathbf{x}}_1 + \frac{n_1}{n_1+n_2}\bar{\mathbf{x}}_2$
$c = \frac{s_1}{s_1+s_2}\bar{\mathbf{x}}_1 + \frac{s_1}{s_1+s_2}\bar{\mathbf{x}}_2$
$c = \frac{s_2/\sqrt{n_2}}{s_1/\sqrt{n_1}+s_2\sqrt{n_2}}\bar{\mathbf{x}}_1 + \frac{s_1/\sqrt{n_1}}{s_1/\sqrt{n_1}+s_2\sqrt{n_2}}\bar{\mathbf{x}}_2$
$c = \frac{1}{2}\mathbf{x}_1^{med} + \frac{1}{2}\mathbf{x}_2^{med}$
$c = \frac{n_1}{n_1+n_2}\mathbf{x}_1^{med} + \frac{n_1}{n_1+n_2}\mathbf{x}_2^{med}$
$c = \frac{IQR_2}{IQR_1+IQR_2}\mathbf{x}_1^{med} + \frac{IQR_1}{IQR_1+IQR_2}\mathbf{x}_2^{med}$
$c = \frac{IQR_2/\sqrt{n_2}}{IQR_1/\sqrt{n_2}+IQR_2/\sqrt{n_2}}\mathbf{x}_1^{med} + \frac{IQR_1/\sqrt{n_1}}{IQR_1/\sqrt{n_1}+IQR_2/\sqrt{n_2}}\mathbf{x}_2^{med}$

Características

- PPtree produce árboles sencillos
- Si existe una banda lineal en los datos, PPtree produce un árbols sin error de clasificación.
- En cada nodo el PPtree separa dos clase susando combinaciones lineales, el número de clases es el mismo que el número de nodos finales por lo que la profundidad del árboles como mucho G-1.
- No necesita ser podado.
- En cada nodo, es posible hacer histogramas de los datos proyectados que puede servir para estudiar la separación. Entonces la razón del la mala clasificación puede ser analizada en cada nodo.
- Los coeficientes de proyección pueden ser usados para medir la contribución de las variables (los datos deben estar estandarizados)

Implementación

```r
library(PPtreeViz)
library(help=PPtreeViz)
data(iris)
Tree.result <- PPTreeclass(Species~.,data = iris,"LDA")
Tree.result
```

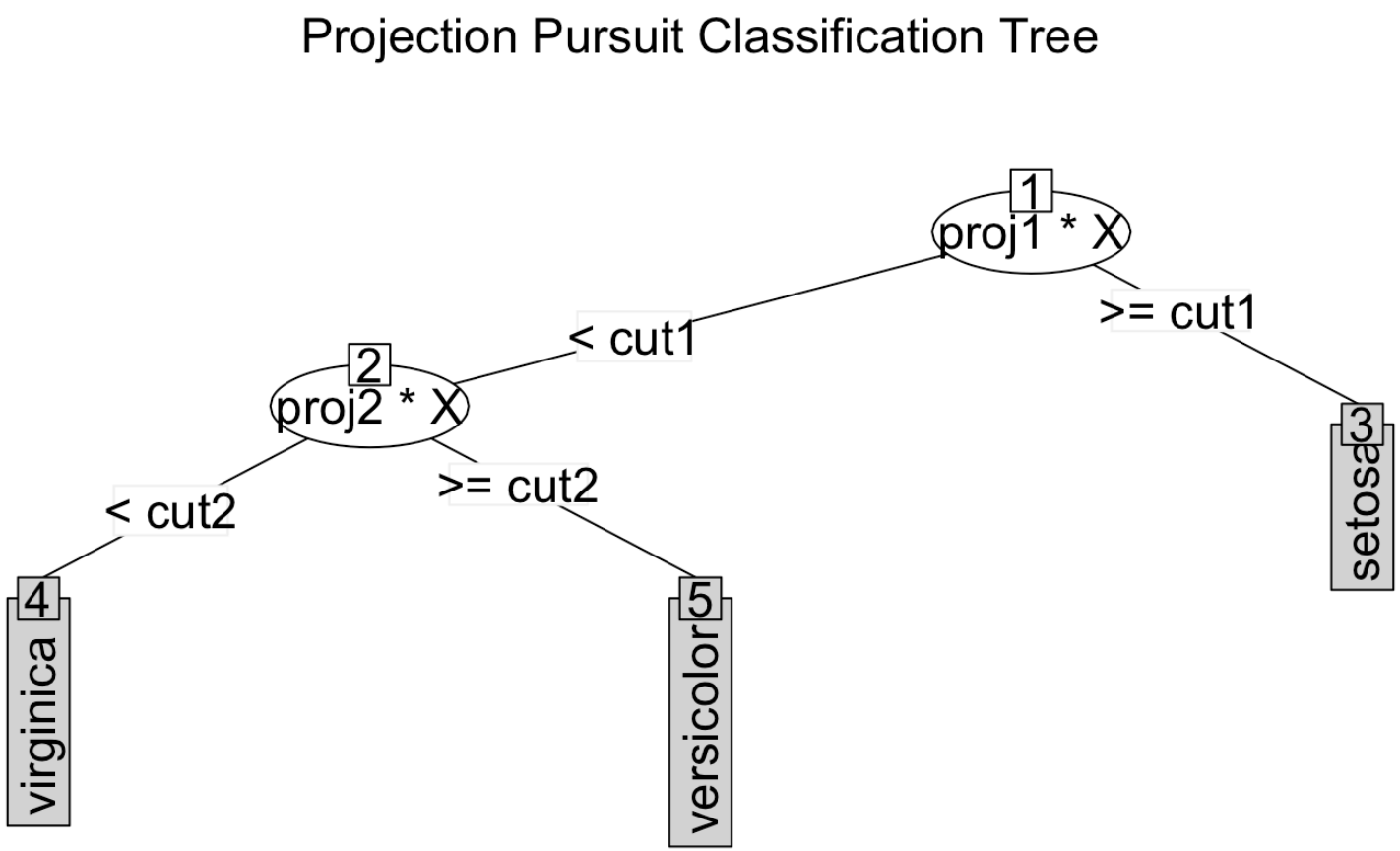
```
=============================================================
Projection Pursuit Classification Tree result
=============================================================

1) root
   2)  proj1*X < cut1
      4)* proj2*X < cut2  ->  "virginica"
      5)* proj2*X >= cut2  ->  "versicolor"
   3)* proj1*X >= cut1  ->  "setosa"

Error rates of various cutoff values
-------------------------------------------------------------
```

Implementación

```
1  plot(Tree.result)
```



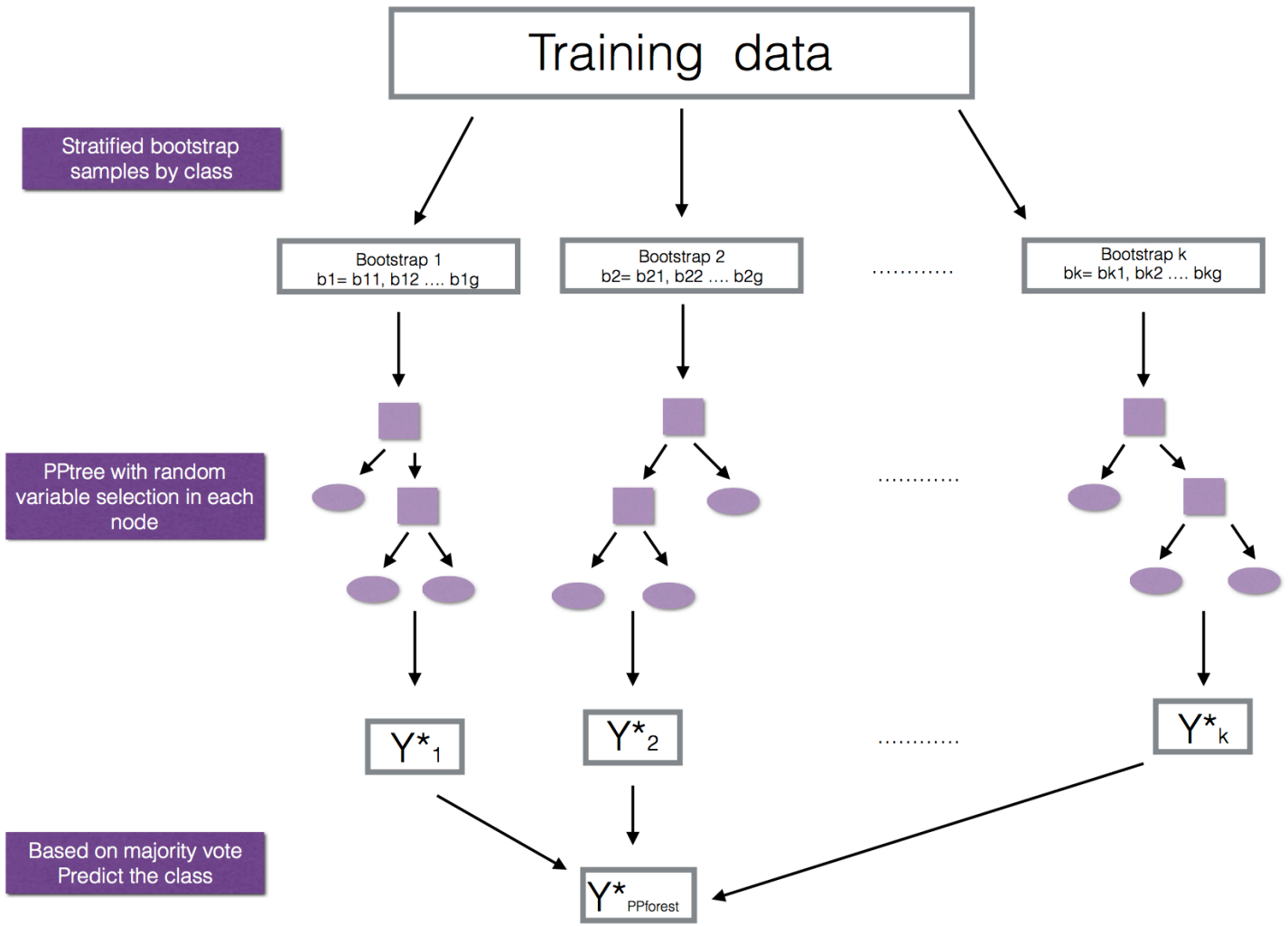
Extensión PPforest

daSilva, Cook, and Lee (2021)

Para problemas de **clasificación**, usa árboles de proyección `PPtree` en vez de CART.

- Usa bootstrap estratificado para atacar desbalance
- Usa combinaciones lineales para separar las clases
- Aprovecha la correlación entre las variables para separar grupos

PPforest diagrama



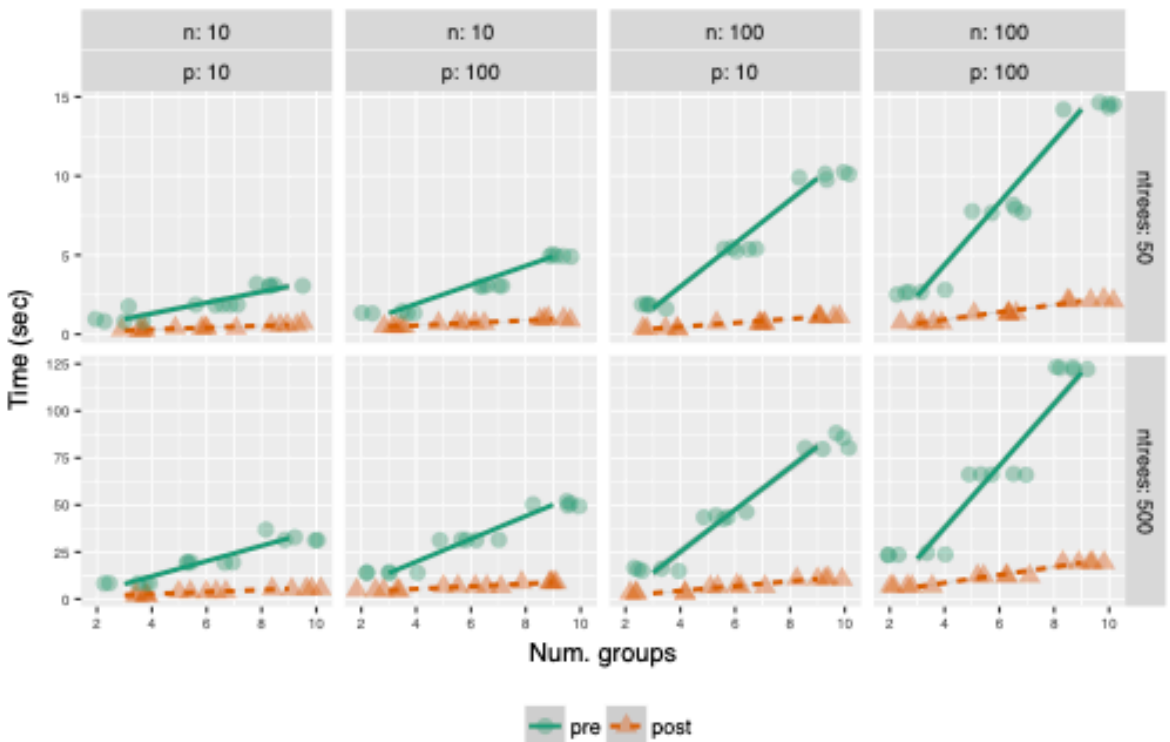
PPforest: Implementación

- Implementado en un paquete de R `PPforest`
- Versión inicial de `PPforest` enteramente en R.

Dos estrategias para optimizar el código fueron empleadas:

- Funciones principales en Rcpp (integración de R con C++)
- Paralelización

PPforest: Tiempo de procesamiento



PPforest implementación

```r
library(PPforest)
pprf.crab<- PPforest(data = crab, class = 'Type',
 std = FALSE, size.tr = 1, m = 200, size.p = .5,
 PPmethod = 'LDA' , parallel = TRUE, cores = 2, rule=1)
pprf.crab
```

```
Call:
 PPforest(data = crab, class = "Type", std = FALSE, size.tr = 1,
m = 200, PPmethod = "LDA", size.p = 0.5, parallel = TRUE,      cores
= 2, rule = 1)
               Type of random forest: Classification
                     Number of trees: 200
No. of variables tried at each split: 2

        OOB estimate of  error rate: 6%
Confusion matrix:
             BlueFemale BlueMale OrangeFemale OrangeMale class.error
```

APP

Quantile RF

Meinshausen (2006)

Para problemas de regresión.

- En RF aproximamos $\mathrm{E}(\mathrm{Y}/\mathrm{X} = \mathrm{x})$ como un promedio ponderado de las observaciones de Y.
- El promedio ponderado podría aproximar bien no solo la esperanza condicional sino toda la distribución condicional, en esto se enfoca QRF

$$\mathrm{F}(\mathrm{y}/\mathrm{X} = \mathrm{x}) = \mathrm{P}(\mathrm{Y} \leq \mathrm{y}/\mathrm{X} = \mathrm{x}) = \mathrm{E}(\mathrm{I}_{\{\mathrm{Y} \leq \mathrm{y}\}} /\mathrm{X} = \mathrm{x})$$

De forma similar a RF se puede aproximar $\mathrm{E}(\mathrm{I}_{\{\mathrm{Y} \leq \mathrm{y}\}} /\mathrm{X} = \mathrm{x})$ como un promedio ponederado de las observaciones $\mathrm{I}_{\{\mathrm{Y} \leq \mathrm{y}\}}$

RF vistos como promedios ponderados

En el usamos el promedio de la pedicción de B árboles,

$$\begin{aligned}
\hat{f}_{rf}(x) &= \frac{1}{B}\sum_b T(x, \Theta_b) \\
&= \frac{1}{B}\sum_b \sum_{i=1}^{n} w_i(x, \Theta_b) Y_i \\
&= \sum_{i=1}^{n} \left(\frac{1}{B}\sum_b w_i(x, \Theta_b)\right) Y_i \\
&= \sum_{i=1}^{n} w_i(x) Y_i
\end{aligned}$$

donde $w_i(x) = \frac{1}{B}\sum_{t=1}^{B} w_i(x, \theta_t)$.

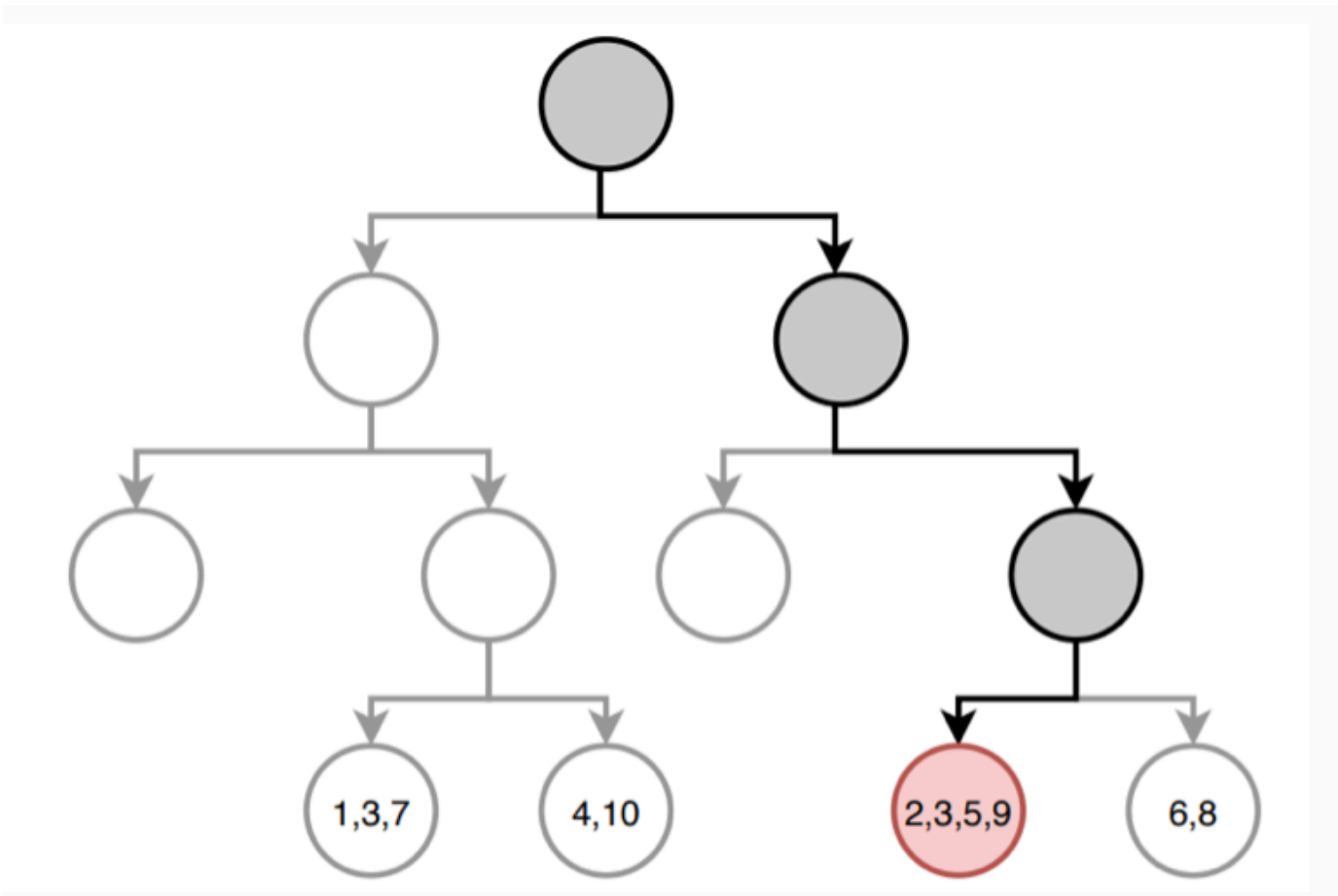
Notar que $w_i(x, \theta)$ suman 1 en cada árbol y $w_i(x)$ suma 1 en el bosque.

Ejemplo con 10 observaciones

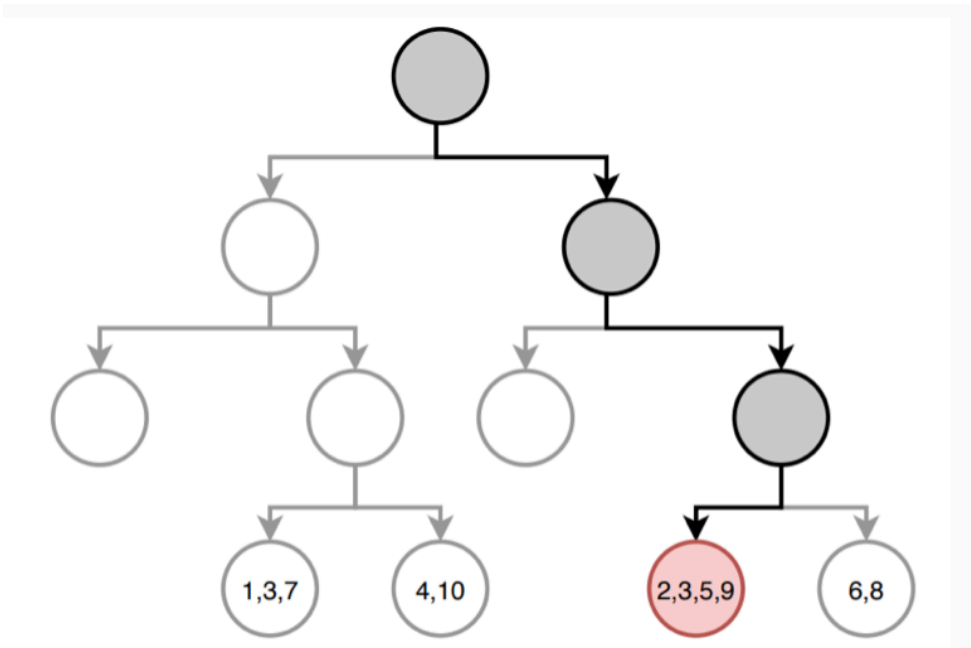
id	1	2	3	4	5	6	7	8	9	10
Y	10	18	24	8	2	9	16	10	20	14
X	…	…	…	…	…	…	…	…	…	…

Un solo árbol

Predicción con un árbol de regresión para $\mathbf{x}$: el camino hasta llegar al nodo final está resaltado. En cada nodo terminal se detalla el índice de las observaciones de entrenamiento que forman parte de la hoja.



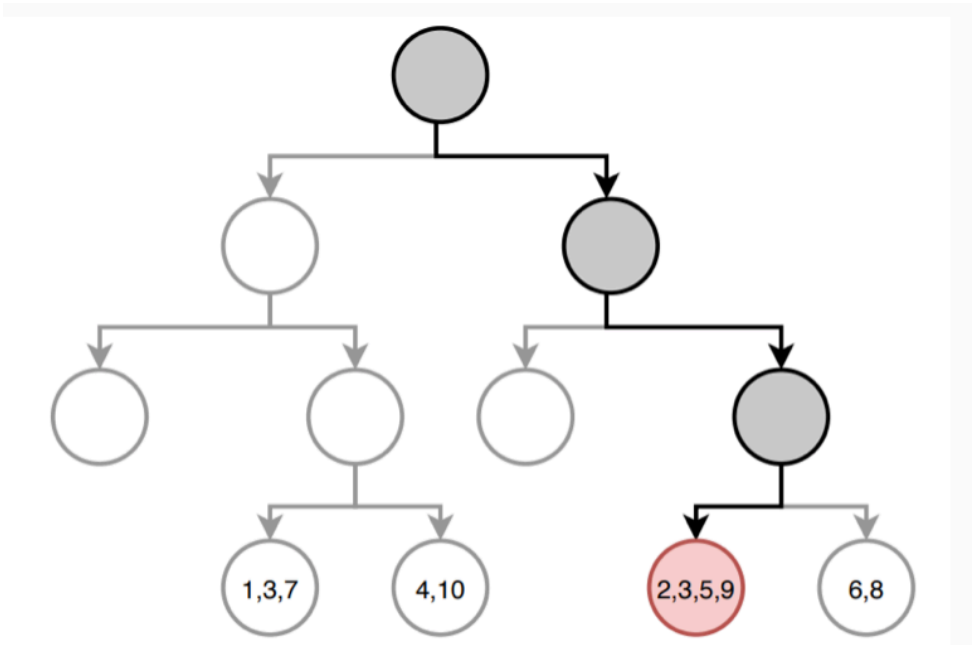
Predicción un árbol de regresión



- id: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- Y: 10,18,24,8,2,9,16,10,20,14
- El valor predicho por el árbol es la media de la variable respuesta Y de las observaciones con id: 2, 3, 5, 9.
- $\hat{f}_{tr}(x) = \frac{18+24+2+20}{4} = 16$

Predicción un árbol de regresión

Usando la notación de ponderadores



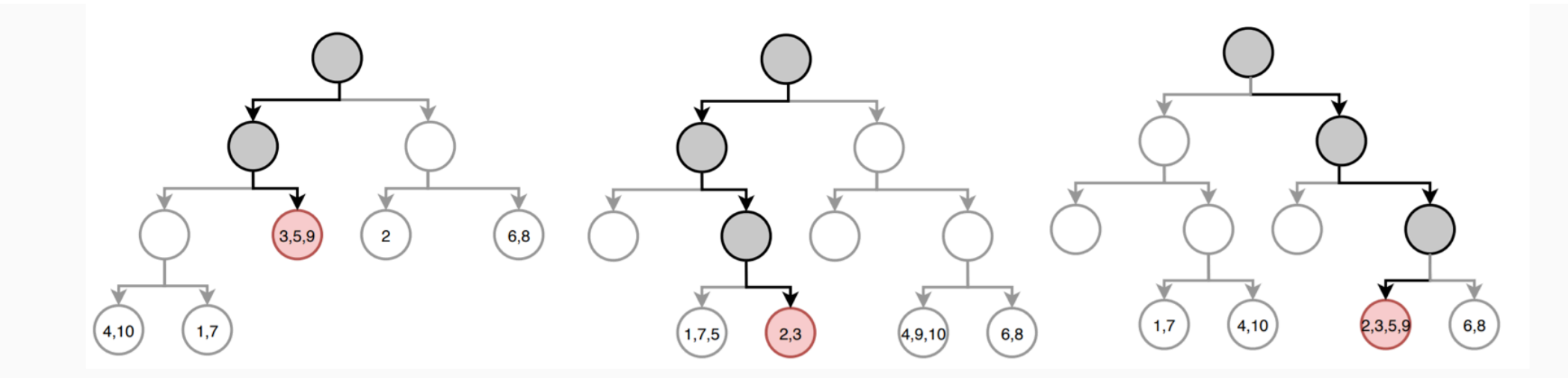
- id: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- Y: 10,18,24,8,2,9,16,10,20,14
- El valor predicho por el árbol es la media de la variable respuesta Y de las observaciones con id: 2, 3, 5, 9.
- $\hat{f_{tr}}(x) = \sum_{i=1}^{10} w_i(x, \theta) Y_i$
- $w = (0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, 0, 0, 0, \frac{1}{4}, 0)$
- $\hat{f_{tr}}(x) = w * Y^T$

Predicción un árbol de regresión

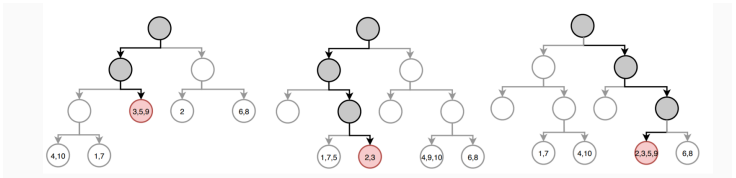
$$\hat{f}_{tr}(x) = w * Y^T = \begin{pmatrix} 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & 0 \end{pmatrix} \begin{pmatrix} 10 \\ 18 \\ 24 \\ 8 \\ 2 \\ 9 \\ 16 \\ 10 \\ 20 \\ 14 \end{pmatrix}$$

Predicción del bosque

Predicción con random forest: en cada árbol, el camino hasta llegar al nodo final está resaltado. En cada nodo terminal se detalla el índice de las observaciones de entrenamiento que forman parte de él.



Bosque con tres árboles



- id: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Y: 10,18,24,8,2,9,16,10,20,14
- $w_{arbol1} = (0, 0, \frac{1}{3}, 0, \frac{1}{3}, 0, 0, 0, \frac{1}{3}, 0)$
- $w_{arbol2} = (0, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0, 0, 0, 0)$
- $w_{arbol3} = (0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, 0, 0, 0, \frac{1}{4}, 0)$

- $\bar{w} = \frac{1}{3}(w_{arbol1} + w_{arbol2} + w_{arbol3})$

$= (0, \frac{1}{4}, \frac{13}{36}, 0, \frac{7}{36}, 0, 0, 0, \frac{7}{36}, 0)$

Bosque con tres árboles

$$\hat{f}_{rf}(x) = \bar{w} * Y^T = \begin{pmatrix} 0 & \frac{1}{4} & \frac{13}{36} & 0 & \frac{7}{36} & 0 & 0 & 0 & \frac{7}{36} & 0 \end{pmatrix} \begin{pmatrix} 10 \\ 18 \\ 24 \\ 8 \\ 2 \\ 9 \\ 16 \\ 10 \\ 20 \\ 14 \end{pmatrix}$$

Quantile RF

QRF propone estimar la distribución acumulada como:

$$\hat{F}(y|X = x) = \frac{1}{B}\sum_{i=1}^{B}\sum_{i=1}^{n} w_i(x, \theta)I(Y_i \leq y) = \sum_{i=1}^{n} w_i(x)I(Y_i \leq y)$$

Estimación empírica de la función de distribución

Algoritmo:

- Mismo algoritmo que RF pero en cada hoja de cada árbol se queda con todas las observaciones no sólo con el promedio
- Para cada $X = x$ se recorren todos los árboles y se calcula los pesos $w_i(x, \theta_t)$ y su promedio $w_i(x)$
- Estimo la funcion de distribución como se definió anteriormente para todo $y \in R$

Y se estima el cuantil condicional Q_α como $\hat{Q_\alpha} = \inf\{y : \hat{F}(y/X = x) \geq \alpha\}$

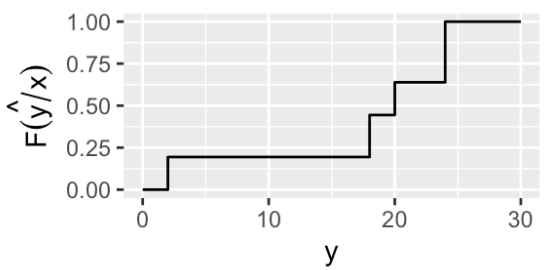
Quantile RF

Bosque con tres árboles

$Y_{ordenado}$: 2, 8, 9, 10, 10, 14, 16, 18, 20, 24

$\bar{w}_{ordenado} = (\frac{7}{36}, 0, 0, 0, 0, 0, 0, \frac{1}{4}, \frac{7}{36}, \frac{13}{36})$

$$\hat{F}(y|X = x) = \sum_{i=1}^{n} w_i(x) I(Y_i \leq y)$$



Una vez obtenida la estimación de la distribución, se puede calcular cualquiera de sus cuantiles.

Ensembles de Árboles

Why do tree-based models still outperform deep learning on typical tabular data? (**grinsztajn2022tree?**)

Ya mencionamos dos variantes principales:

Bagging (vimos)

- Construir árboles independientes
- Árbol individual grande: poco sesgo, mucha varianza
- Promediar predicciones de cada árbol

Boosting (veremos)

- Construir árboles de forma secuencial
- Árbol individual chico: mucho sesgo, poca varianza
- Agregar las predicciones de cada árbol

Boosting

Boosting de árboles

$$g(x|T, M) = \sum_k \mu_k I(x \in R_k)$$

- T es la estructura del árbol: formada por $\{X_j \leq s\}$
- M es el valor de las hojas del árbol, $M = (\mu_1, \ldots, \mu_b)$

Boosting de árboles

Construir un conjunto de B árboles:

$$\{g_1(x|\cdot), g_2(x|\cdot), \ldots, g_B(x|\cdot)\}$$

- Los árboles son entrenados *secuencialmente*.
- Cada modelo sucesivo aprende de los errores del árbol anterior.
- **AdaBoost**: Adaptive Boosting (Freund & Schapire, 1997)
- Greedy function approximation: a **gradient boosting machine** (Friedman, 2001)

Gradient boosting

A partir de un conjunto de datos (X_i, Y_i), una de las formas básicas de aplicar `gbm` es:

1. Inciar $\hat{g_o}() = \bar{Y}$, calcular $r_i^o = Y_i - \bar{Y}$
2. Para $b = 1, 2, \ldots, B$, se repite:
   - Obtener $\hat{g_b}()$ en base a (X_i, r_i) con $d + 1$ nodos terminales
   - Actualizar modelo: $\hat{f}() = \hat{f}() + \lambda \hat{g_b}()$
   - Actualizar residuos: $r_i = r_i - \lambda \hat{g_b}()$
3. El modelo resultante es:

$$\hat{f}(x) = \hat{g_o}(x) + \lambda \sum_b \hat{g_b}(x)$$

Ejemplo de juguete: datos

los datos ...

```r
d <- data.frame(
  altura = c(1.6, 1.6, 1.5, 1.8, 1.5, 1.4),
  color  = c('A', 'V', 'A', 'N', 'V', 'A'),
  sexo   = c('M', 'F', 'F', 'M', 'M', 'F'),
  peso   = c(88, 76, 56, 77, 73, 57)
)

d
```

```
  altura color sexo peso
1    1.6     A    M   88
2    1.6     V    F   76
3    1.5     A    F   56
4    1.8     N    M   77
5    1.5     V    M   73
6    1.4     A    F   57
```

Ejemplo de juguete: datos

En un primer paso, estimamos todos los valores con el promedio

```
1 mean(d$peso)
```

```
[1] 71.16667
```

```
1 d$peso - mean(d$peso)
```

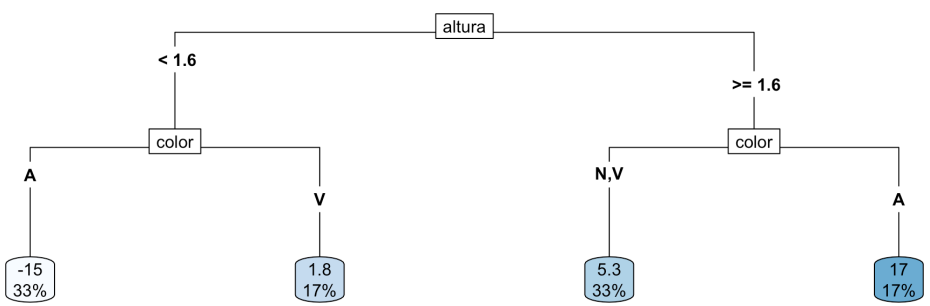
```
[1]  16.833333   4.833333 -15.166667   5.833333   1.833333 -14.166667
```

Ejemplo de juguete: Paso 1

```r
library(rpart)
library(rpart.plot)
d$rr<- d$peso - mean(d$peso)

tt1 <- rpart( rr ~ altura+color+sexo, data=d,
              control = rpart.control(minsplit = 1, minbuck
```

```r
rpart.plot(tt1, type = 5)
```



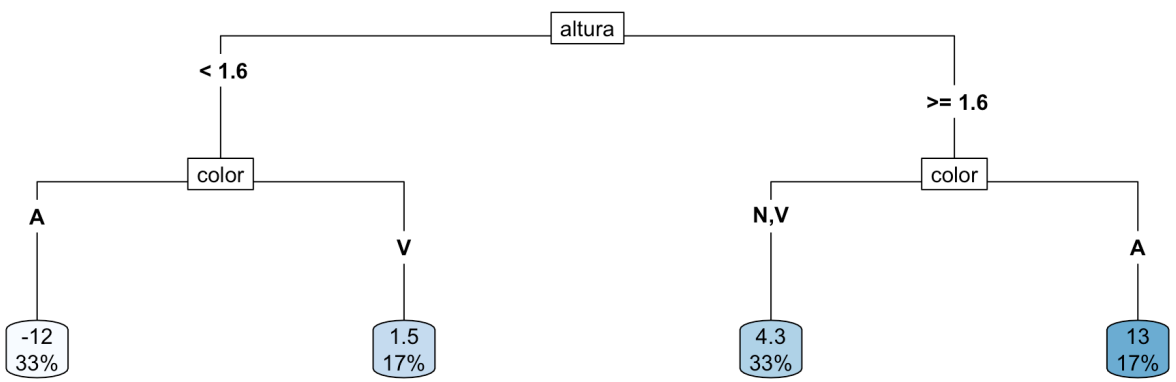
¿Cómo cambia la predicción del primer dato?

Ejemplo de juguete: Paso 2

```
d$rr <- d$rr - 0.2*predict(tt1)

tt2 <- rpart( rr ~ altura+color+sexo, data=d,
              control = rpart.control(minsplit = 1, minbuck
```

```
rpart.plot(tt2, type = 5)
```



¿Cómo cambia la predicción del primer dato?

Ejemplo de juguete: Predicción

¿Como se predice un valor nuevo con nuestro Boosting de 2 árboles?

```
1  nd <- data.frame(altura=1.7, color='A', sexo='F')
```

El predictor resultante con 2 árboles y $\lambda = 0.2$ es:

$$\hat{f}(x) = \hat{g_0}(x) + 0.2\hat{g_1}(x) + 0.2\hat{g_2}(x)$$

```
1  mean(d$peso) + 0.2*predict(tt1, nd) + 0.2*predict(tt2, nd)
```

```
       1
77.22667
```

Referencias

daSilva, Natalia, Dianne Cook, and Eun-Kyung Lee. 2021. “A Projection Pursuit Forest Algorithm for Supervised Classification.” *Journal of Computational and Graphical Statistics* 30 (4): 1168–80.

Lee, Yoon Dong, Dianne Cook, Ji-won Park, and Eun-Kyung Lee. 2013. “PPtree: Projection Pursuit Classification Tree.” *Electronic Journal of Statistics* 7: 1369–86.

Loh, Wei-Yin. 2014. “Fifty Years of Classification and Regression Trees.” *International Statistical Review* 82 (3): 329–48.

Meinshausen, Nicolai. 2006. “Quantile Regression Forests.” *Journal of Machine Learning Research* 7 (Jun): 983–99.