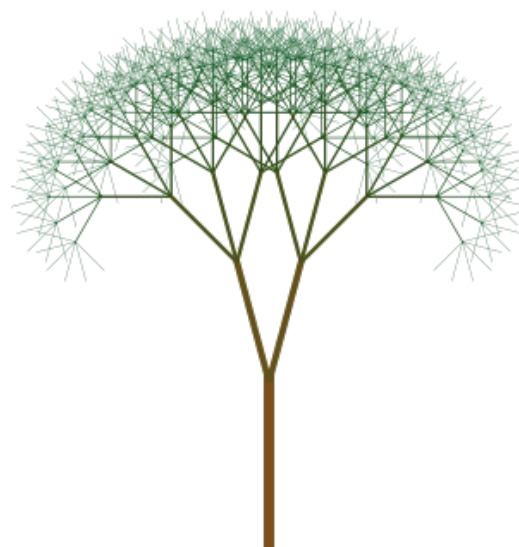


Árboles y Otras Hierbas

Aprendizaje Automático INF398 II-2021

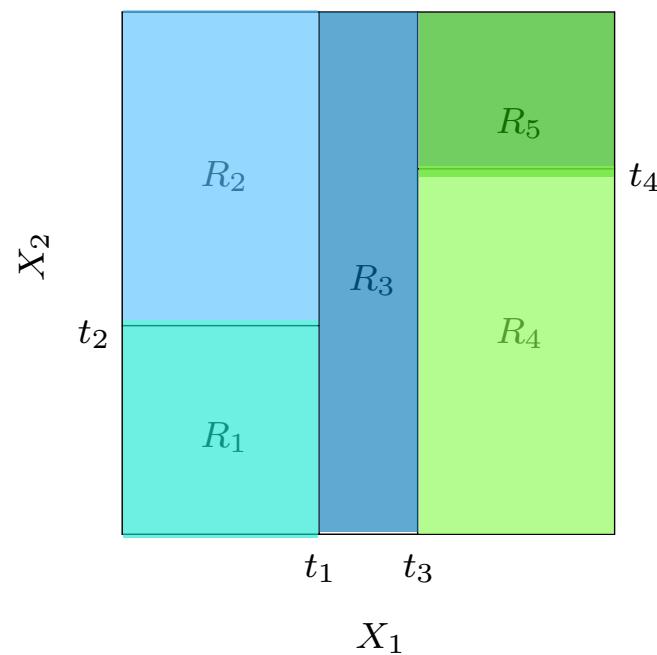


Agenda

- Ideas Generales
- Arquitectura y Aprendizaje

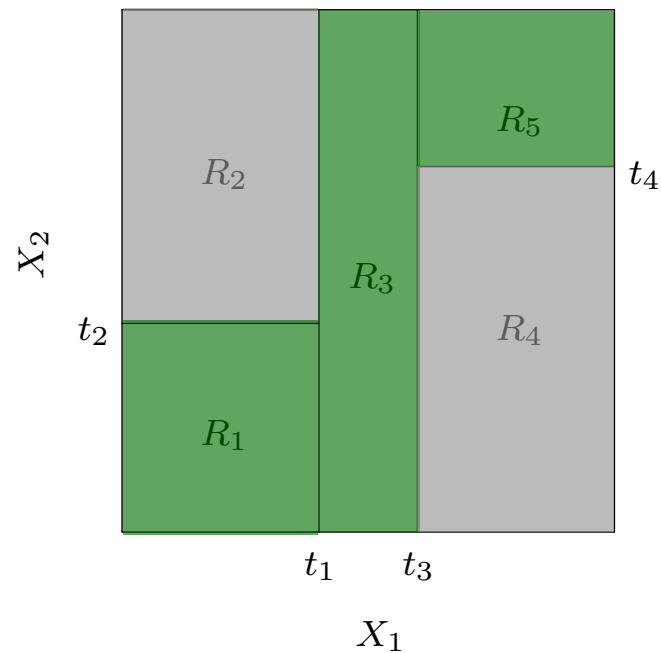
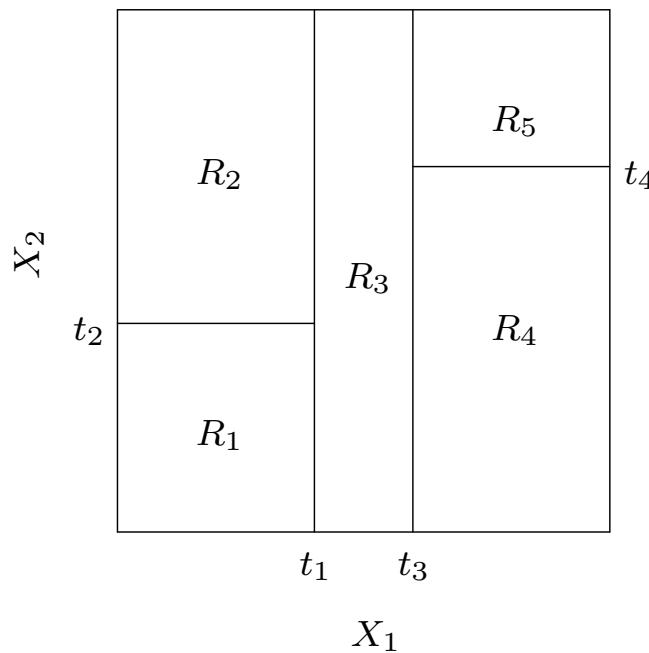
Generalidades

- La estrategia básica detrás de estos métodos es generar una partición del espacio de características en regiones y definir una regla de predicción (potencialmente) diferente para cada una de ellas.



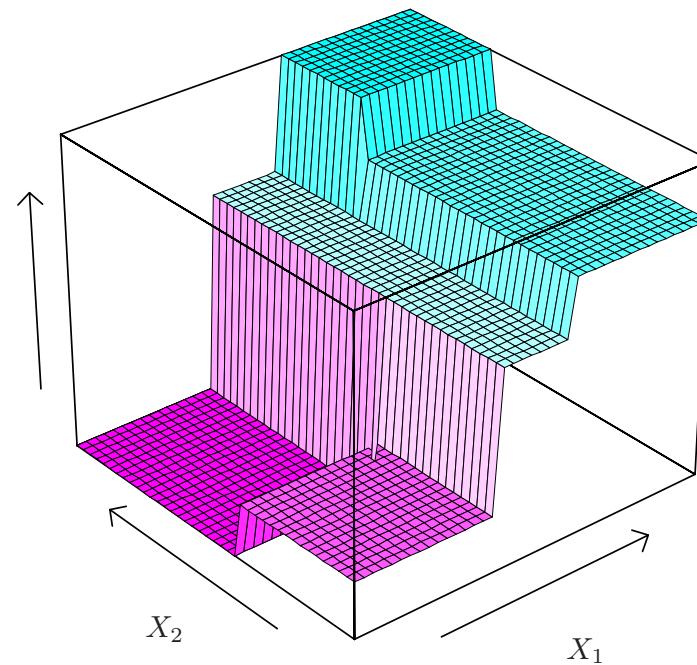
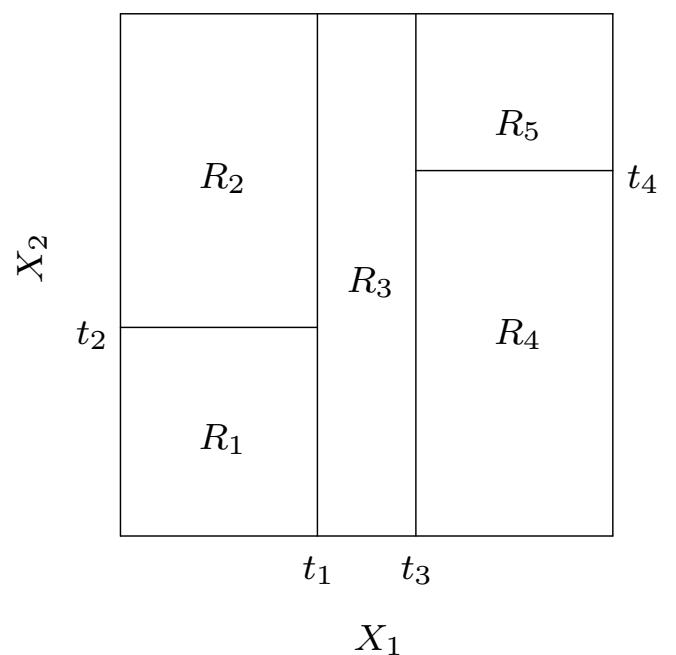
Generalidades

- Por ejemplo, en un problema de clasificación (clase1: rojo, clase2: verde), el siguiente etiquetado de las regiones genera fronteras claramente no lineales en el espacio de características.



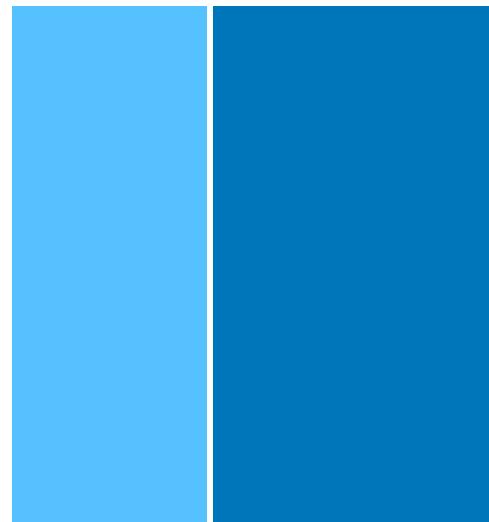
Generalidades

- En un problema de regresión, incluso el etiquetado de las regiones con un valor constante diferente genera una función no lineal de x :



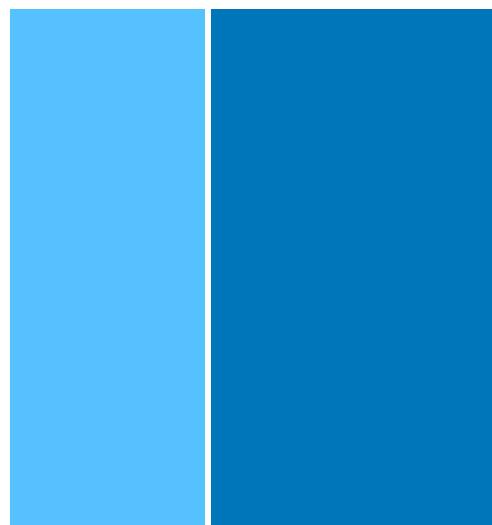
Generalidades

- La forma de particionar el espacio es **recursiva**: luego de hacer una partición, cada región resultante se somete de nuevo al esquema de particionamiento.



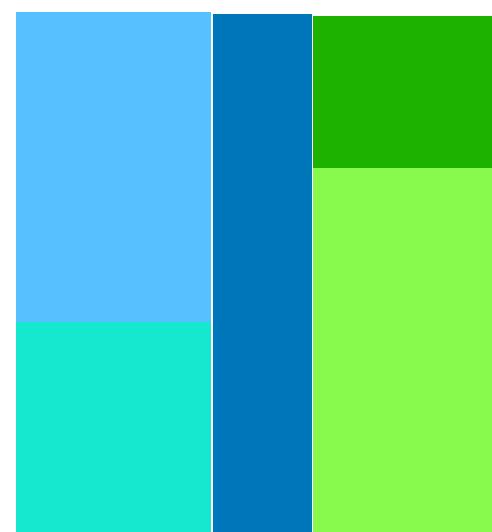
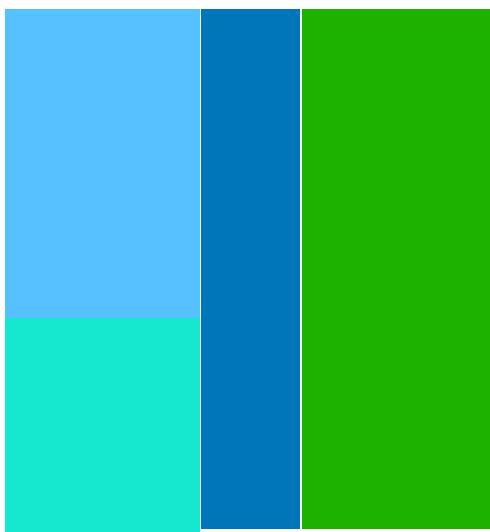
Generalidades

- La forma de particionar el espacio es **recursiva**: luego de hacer una partición, cada región resultante se somete de nuevo al esquema de particionamiento.



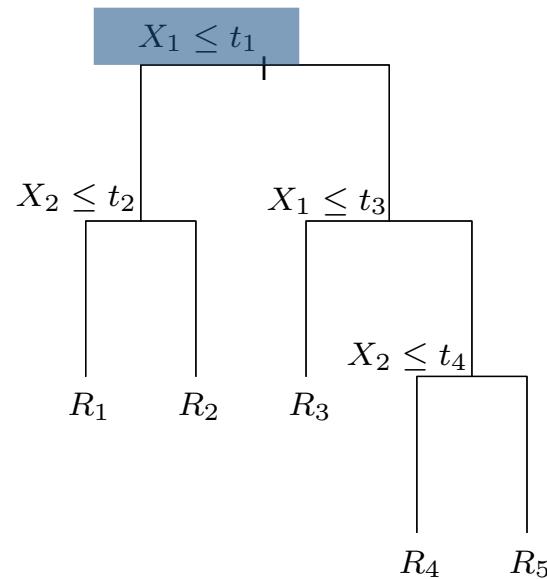
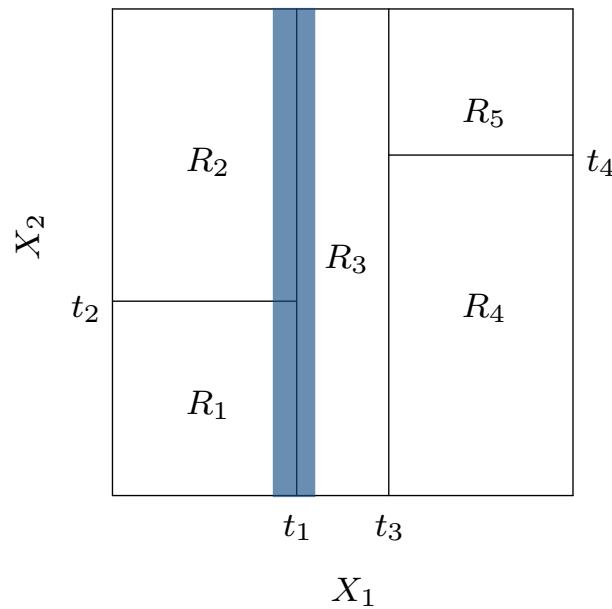
Generalidades

- La forma de particionar el espacio es **recursiva**: luego de hacer una partición, cada región resultante se somete de nuevo al esquema de particionamiento.



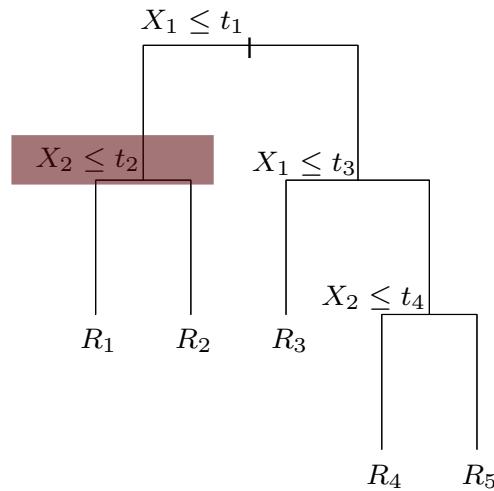
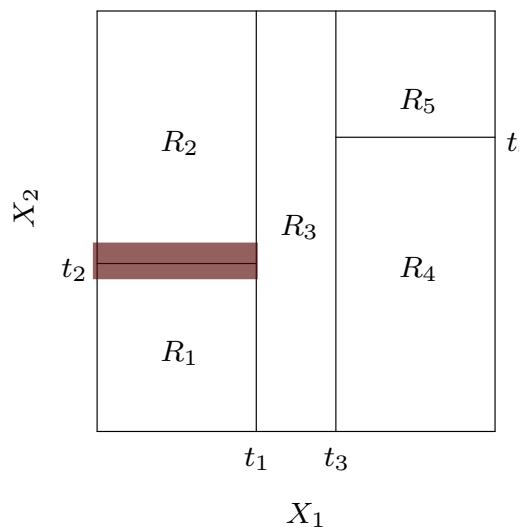
Generalidades

- La ventaja de ese *approach* es que las condiciones utilizadas para particionar se pueden organizar en un **árbol**, donde la raíz representa la regla utilizada para la primera partición.



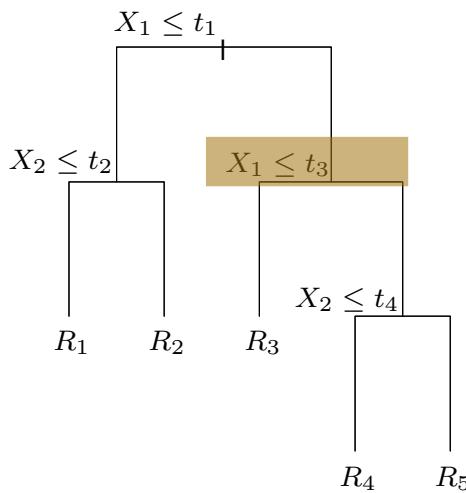
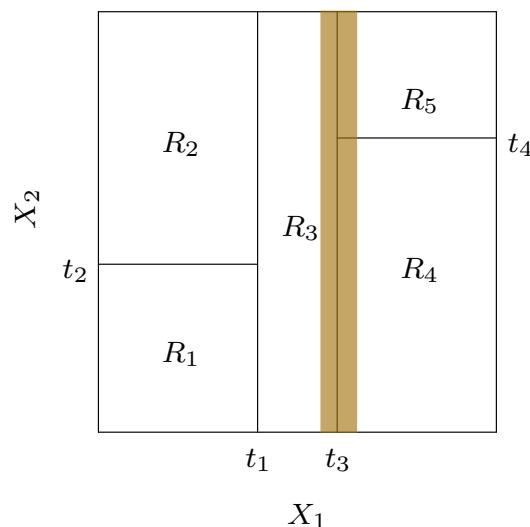
Generalidades

- Y donde los nodos internos representan las reglas utilizadas para las particiones ulteriores.



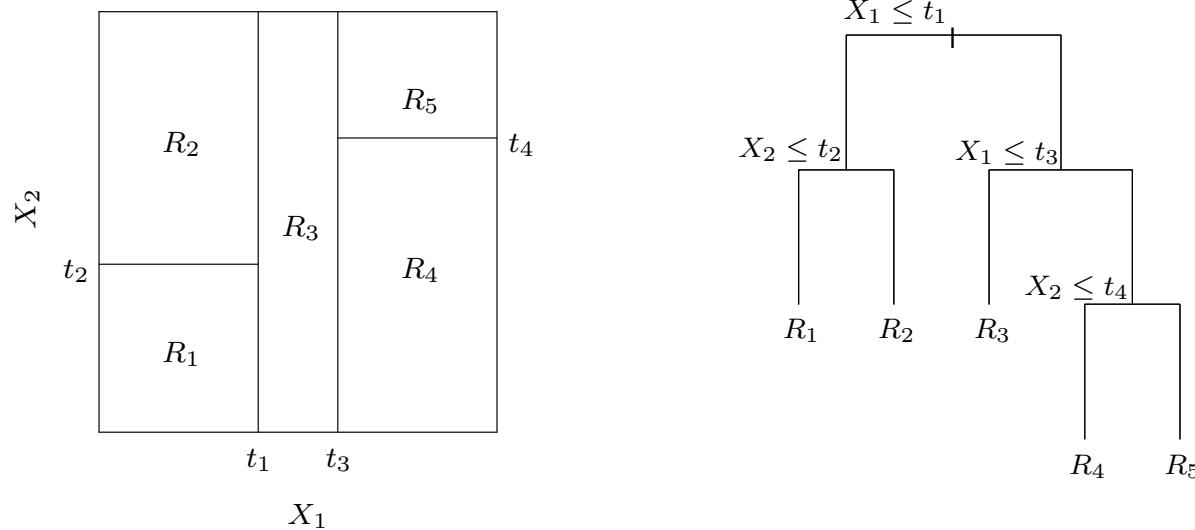
Generalidades

- Y donde los nodos internos representan las reglas utilizadas para las particiones ulteriores.



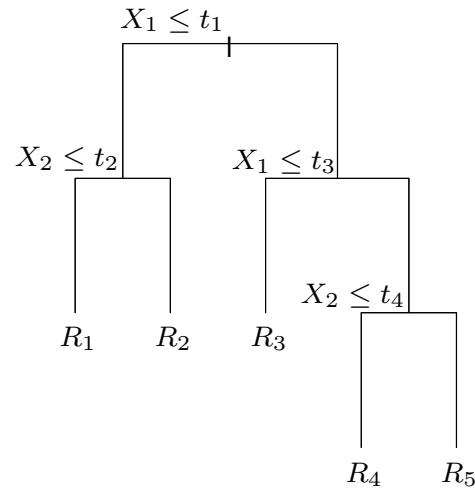
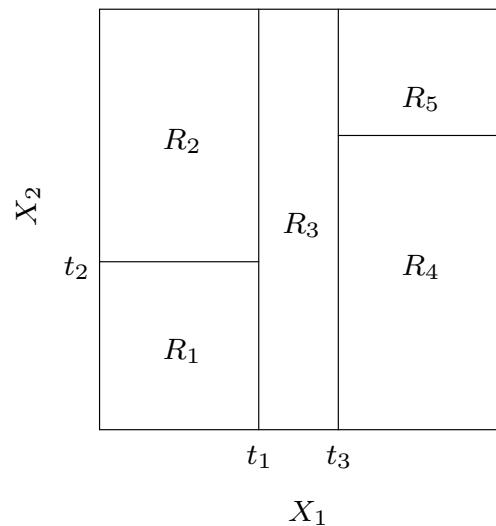
Generalidades

- Como una partición se hace siempre sobre un resultado de la partición anterior, las reglas ó condiciones de un nivel se aplican siempre de modo **condicional al resultado de una regla ó condición del nivel anterior.**



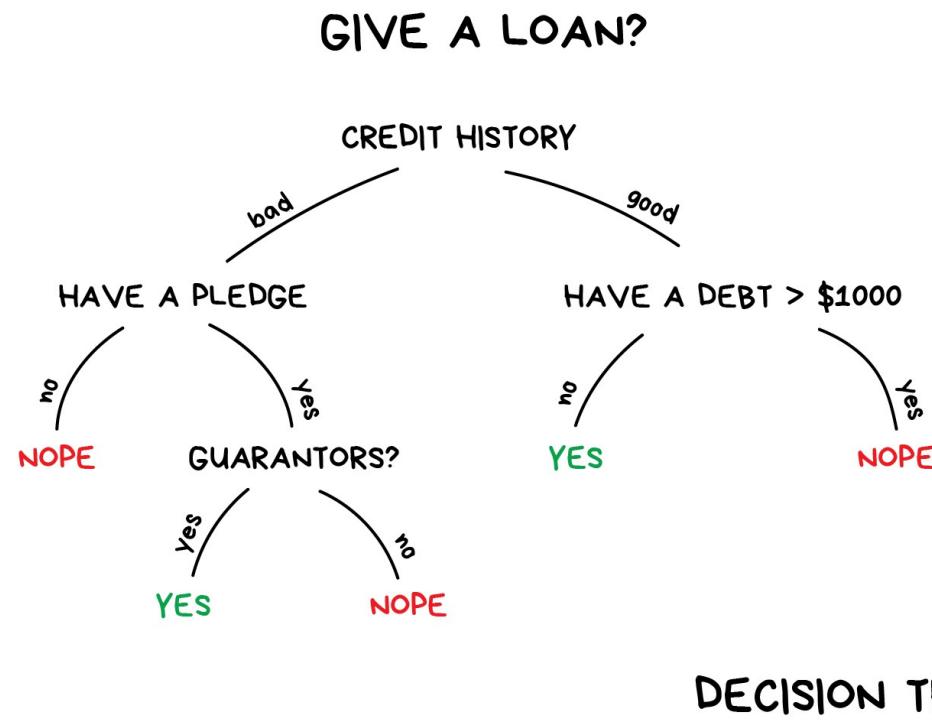
Interpretabilidad

- La principal ventaja de esto es la **interpretabilidad**: el árbol resume de modo muy intuitivo la secuencia de condiciones que es necesario verificar para llegar una región y por lo tanto a una predicción o decisión sobre un dato.



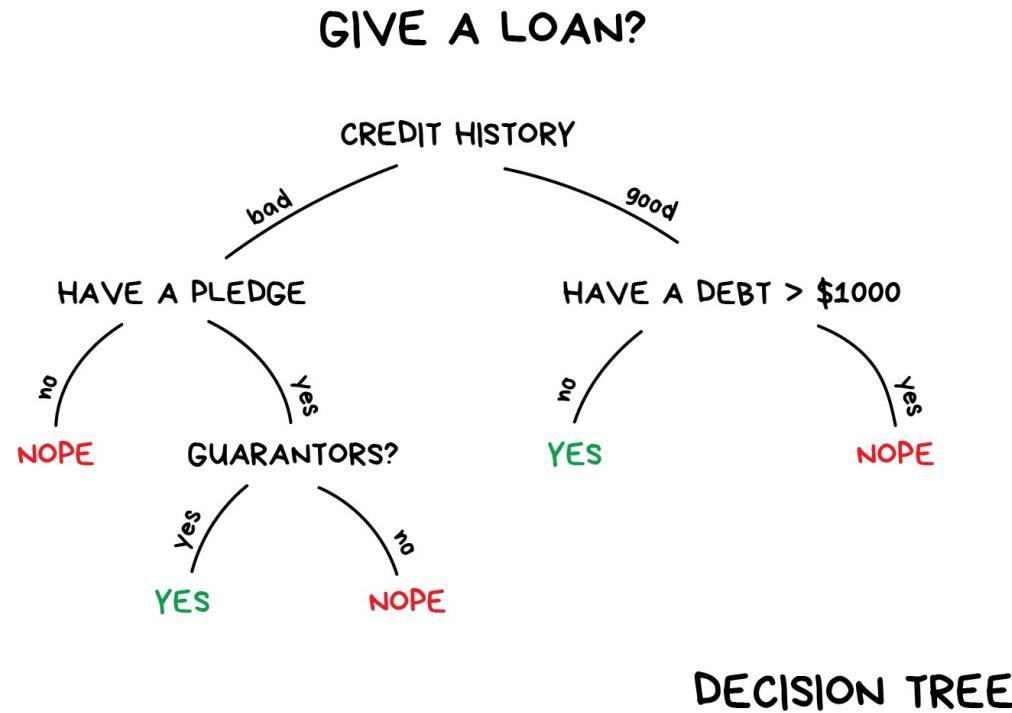
Particiones Canónicas

- La forma canónica de hacer una partición será **elegir** una variable y revisar su valor. Por ejemplo: con variables categóricas, diferentes valores podrían generar diferentes ramas.



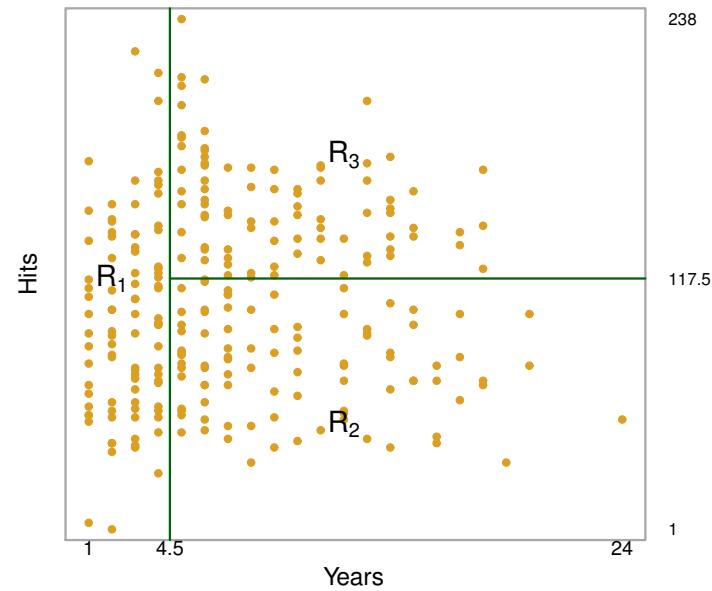
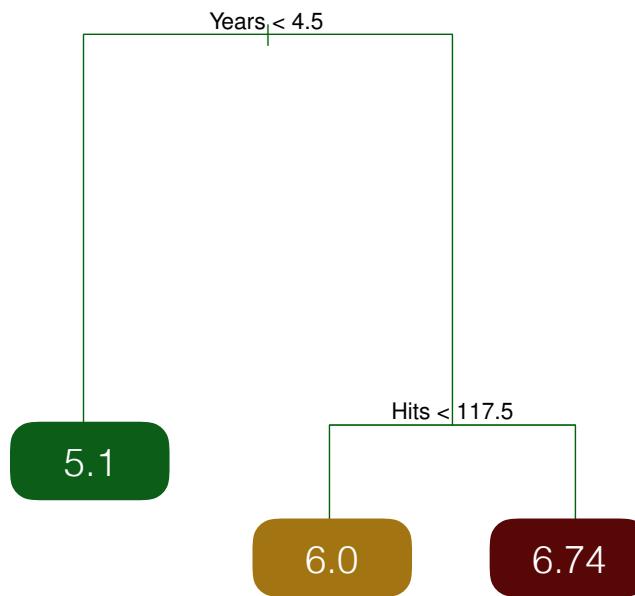
Particiones Canónicas

- Con variables continuas, lo usual es que diferentes rangos de valores generen diferentes ramas.



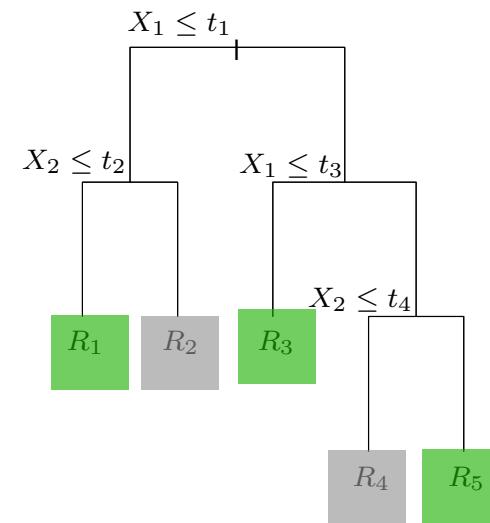
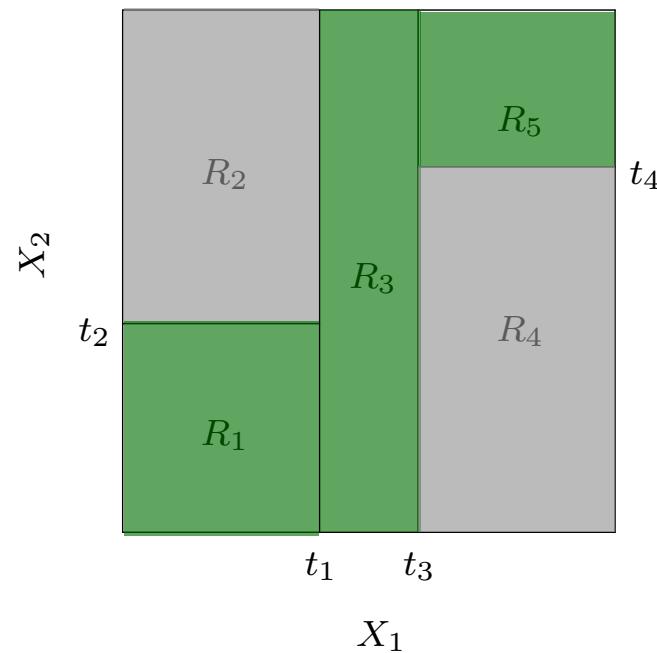
Regiones

- Las regiones del espacio de características que corresponden al árbol se obtienen al concatenar, desde una hoja hasta la raíz las condiciones que generaron las ramificaciones.



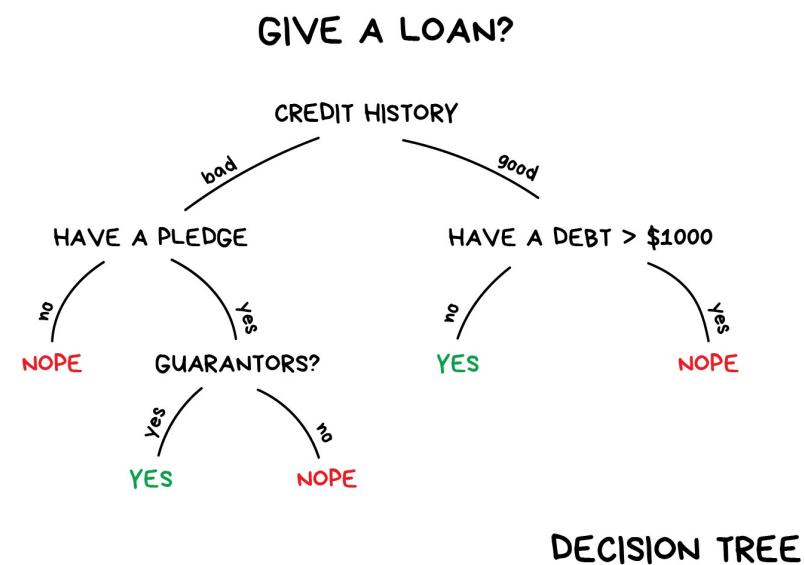
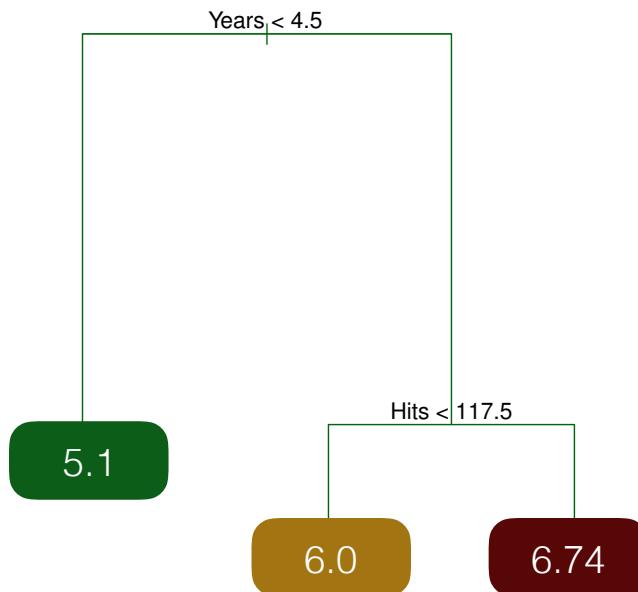
Predicciones

- Como cada región gatilla una predicción, es natural representar estas predicciones en las hojas del árbol.



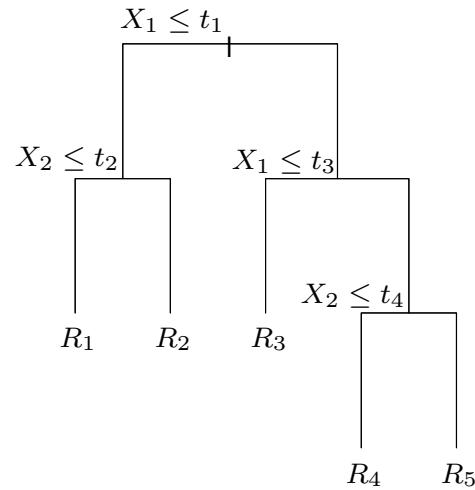
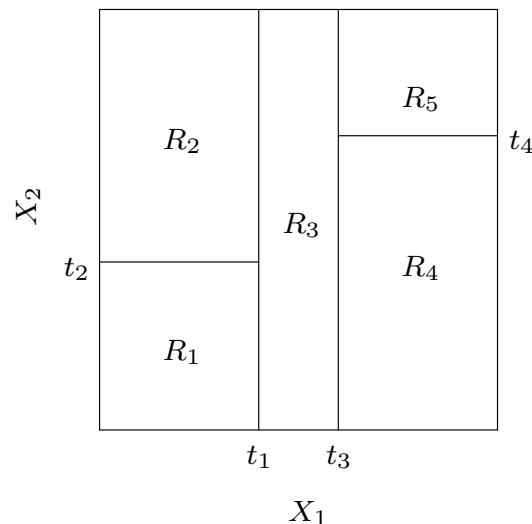
Predicciones

- Es común que la predicción que se aplica a cada región constante: clase, probabilidad o valor continuo (regresión).



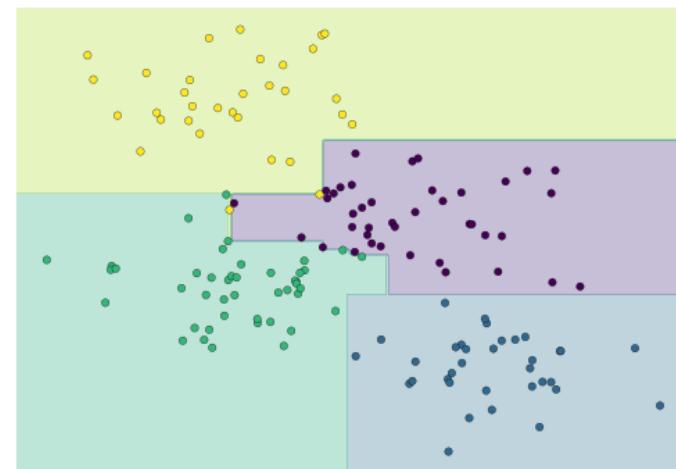
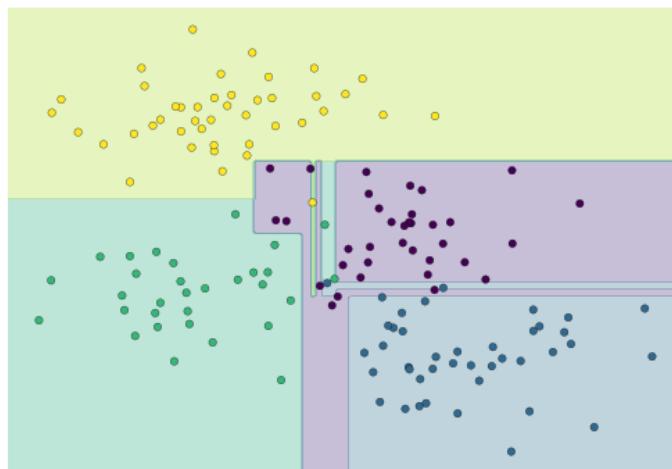
Interpretabilidad

- Todas estas elecciones contribuyen a la interpretabilidad del predictor. No sólo eso sencillo entender cómo se toma una decisión sino también porqué. Además, el árbol entrega de modo implícito un mecanismo para determinar cuáles son las variables más importantes.



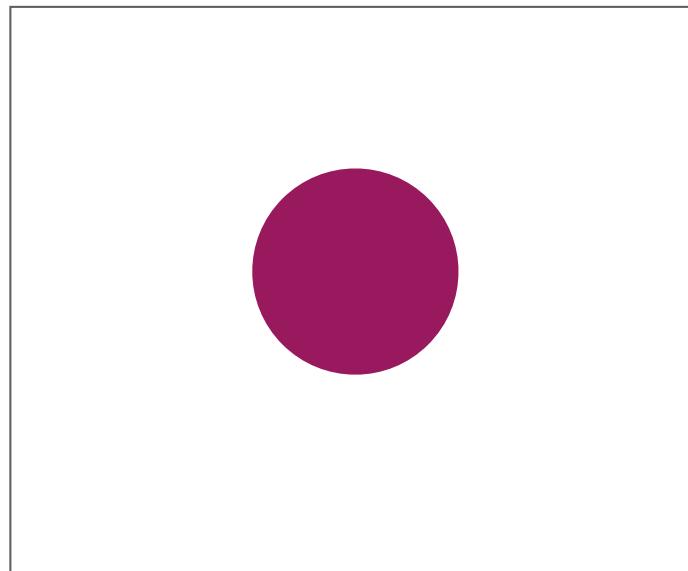
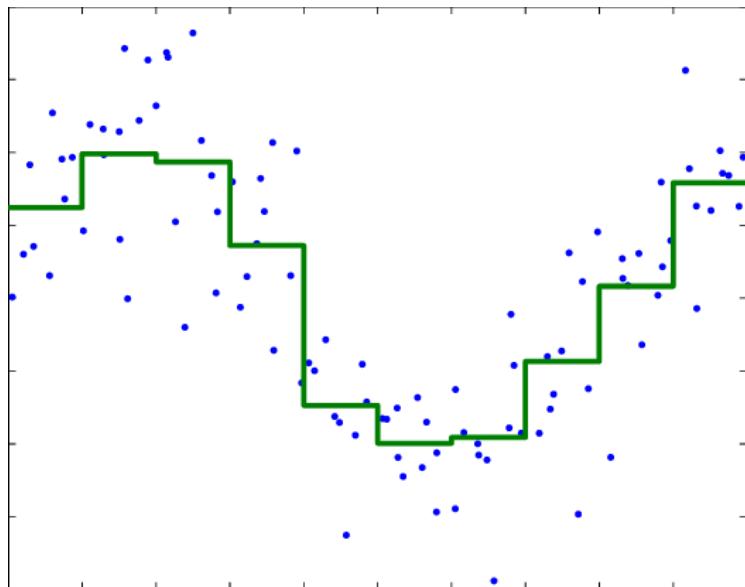
Expresividad

- Toda esta interpretabilidad no resta expresividad: es bastante claro que para un conjunto de entrenamiento cualquiera, podremos siempre encontrar un árbol consistente con el mismo (de hecho podemos encontrar infinitos).



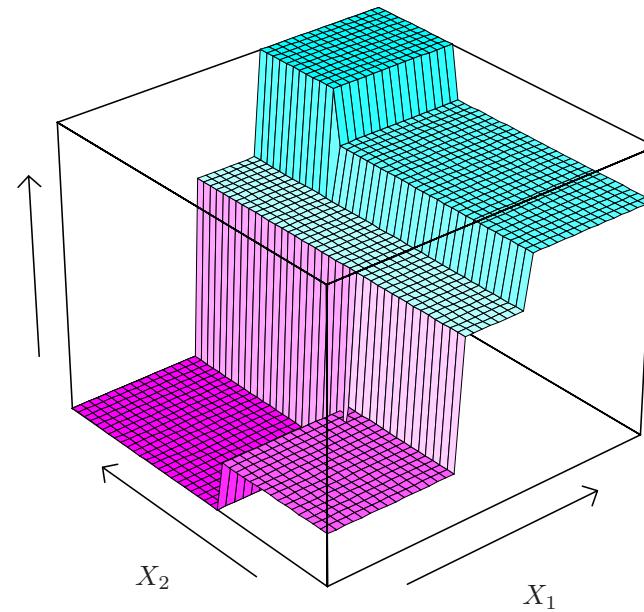
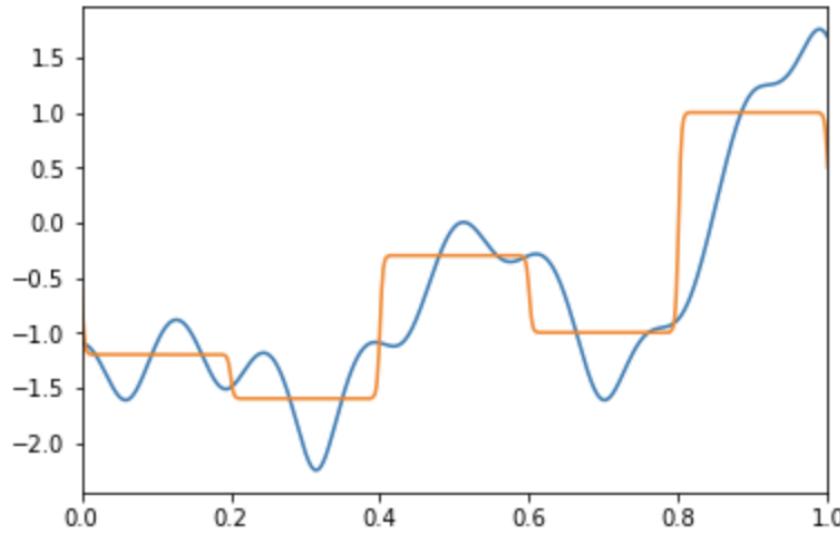
Expresividad

- Si no limitamos el tamaño del árbol, podemos de hecho aproximar cualquier función discreta arbitrariamente bien.



Expresividad

- Y también cualquier función real/continua.

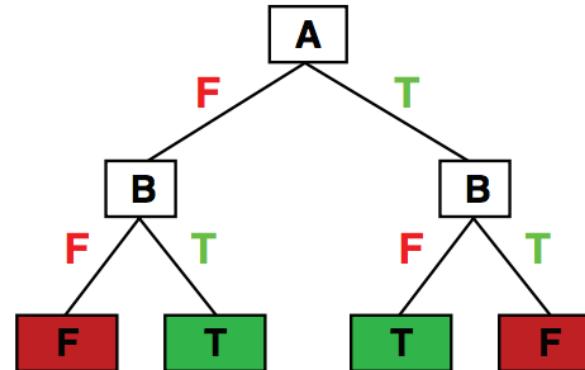


(formamente decimos que los árboles de decisión son
aproximadores universales)

Expresividad

- Para ciertas funciones (e.g. booleanas) la aproximación es exacta.

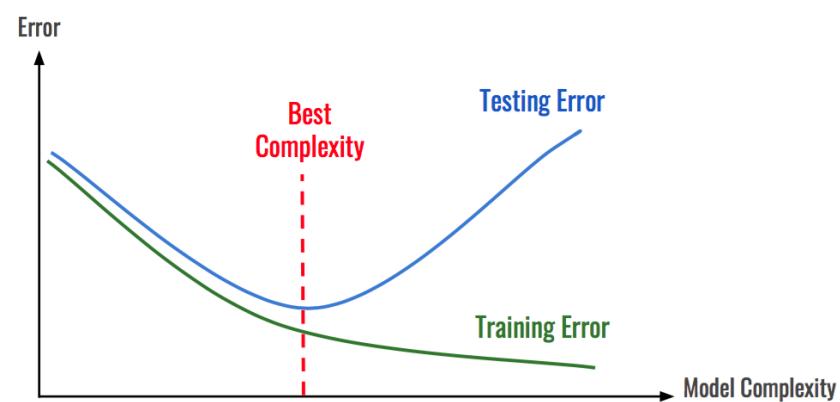
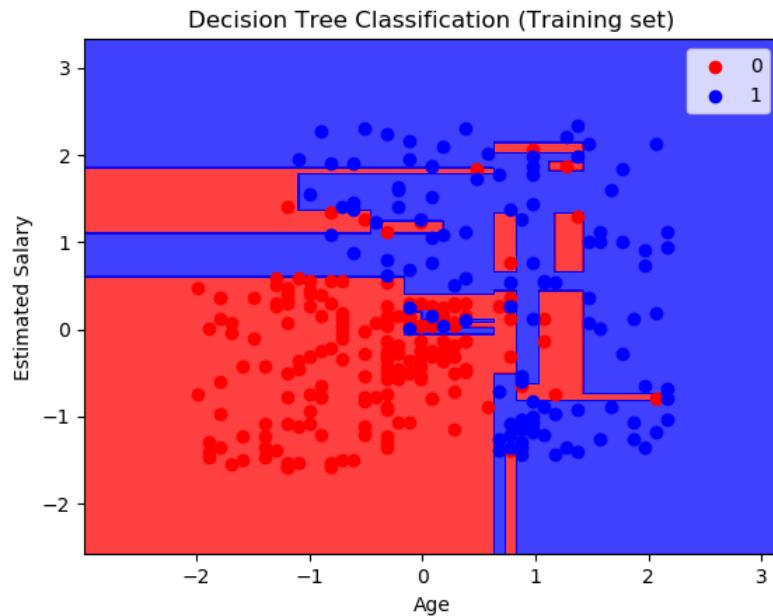
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



(Basta elegir una hoja por entrada de la tabla de verdad: las ramas forman a fórmula lógica)

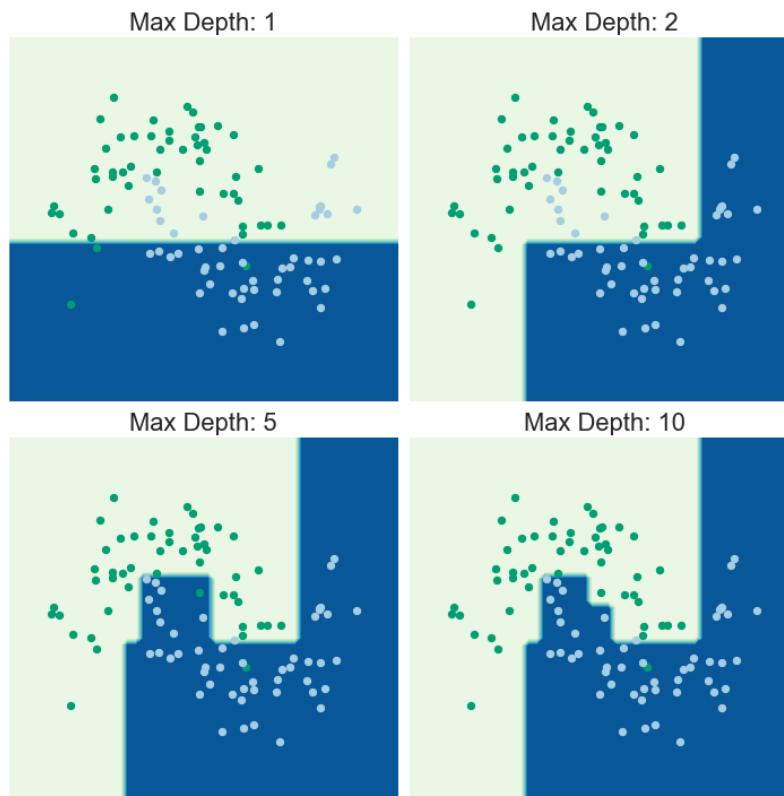
Aprendizaje

- La expresividad de los árboles hace que, si queremos, podamos garantizar **error de aproximación 0**. Sin embargo, como hemos aprendido, esto no garantiza aprendizaje, y aumenta el riesgo de overfitting.



Overfitting

- El mecanismo más eficaz para controlar overfitting en árboles será controlar la **profundidad** del árbol.



NP Completitud

- Lamentablemente, encontrar el árbol más compacto (menor número de test) para una tarea es un problema NP completo (incluso simplificarlo lo es).

The screenshot shows a digital journal cover for 'Information Processing Letters'. The title 'Information Processing Letters' is at the top, followed by 'Volume 5, Issue 1, May 1976, Pages 15-17'. To the left is the Elsevier logo, which includes a tree and the word 'ELSEVIER'. The main text on the page is 'Constructing optimal binary decision trees is NP-complete' with a small star icon. Below the article title, it says 'Laurent Hyafil, Ronald L. Rivest'. There are links for 'Show more ▾', 'Get rights and content', and a DOI link 'https://doi.org/10.1016/0020-0190(76)90095-8'. At the bottom, there are buttons for 'Previous article in issue' and 'Next article in issue'.

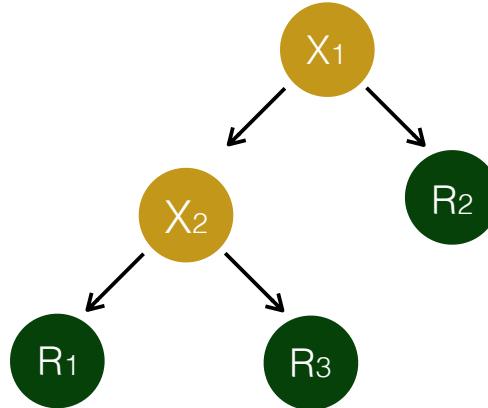
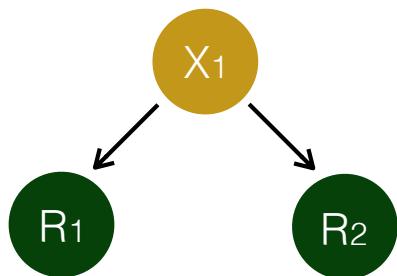
Laurent, Hyafil, and Ronald L. Rivest. "Constructing optimal binary decision trees is NP-complete." *Information processing letters* 5.1 (1976): 15-17.

Construcción del Árbol

- Algoritmos populares:
 - **ID3**: clasificación con atributos discretos
 - **C4.5**: clasificación con atributos discretos o continuos.
 - **CART**: clasificación o regresión con atributos discretos o continuos y control de overfitting vía poda.
 - **MARS**: regresión con funciones basales un poco más complejas.

Top-Down + Greedy

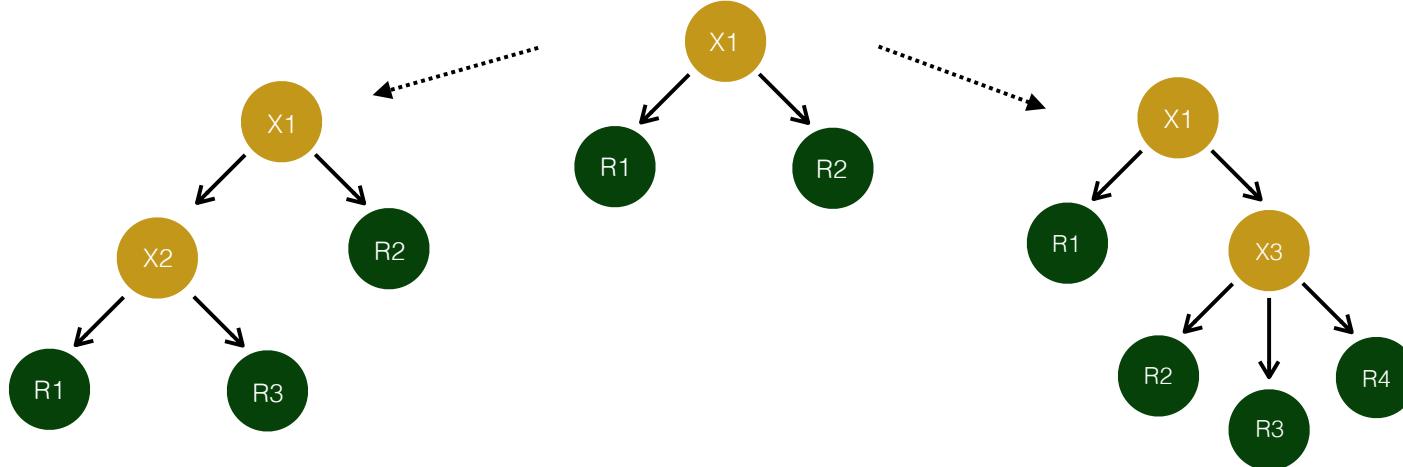
1. Elegir una variable y regla de partición para la raíz.
2. Elegir una de las hojas del árbol para transformarse en nodo interno, marcar como raíz y repetir 1.



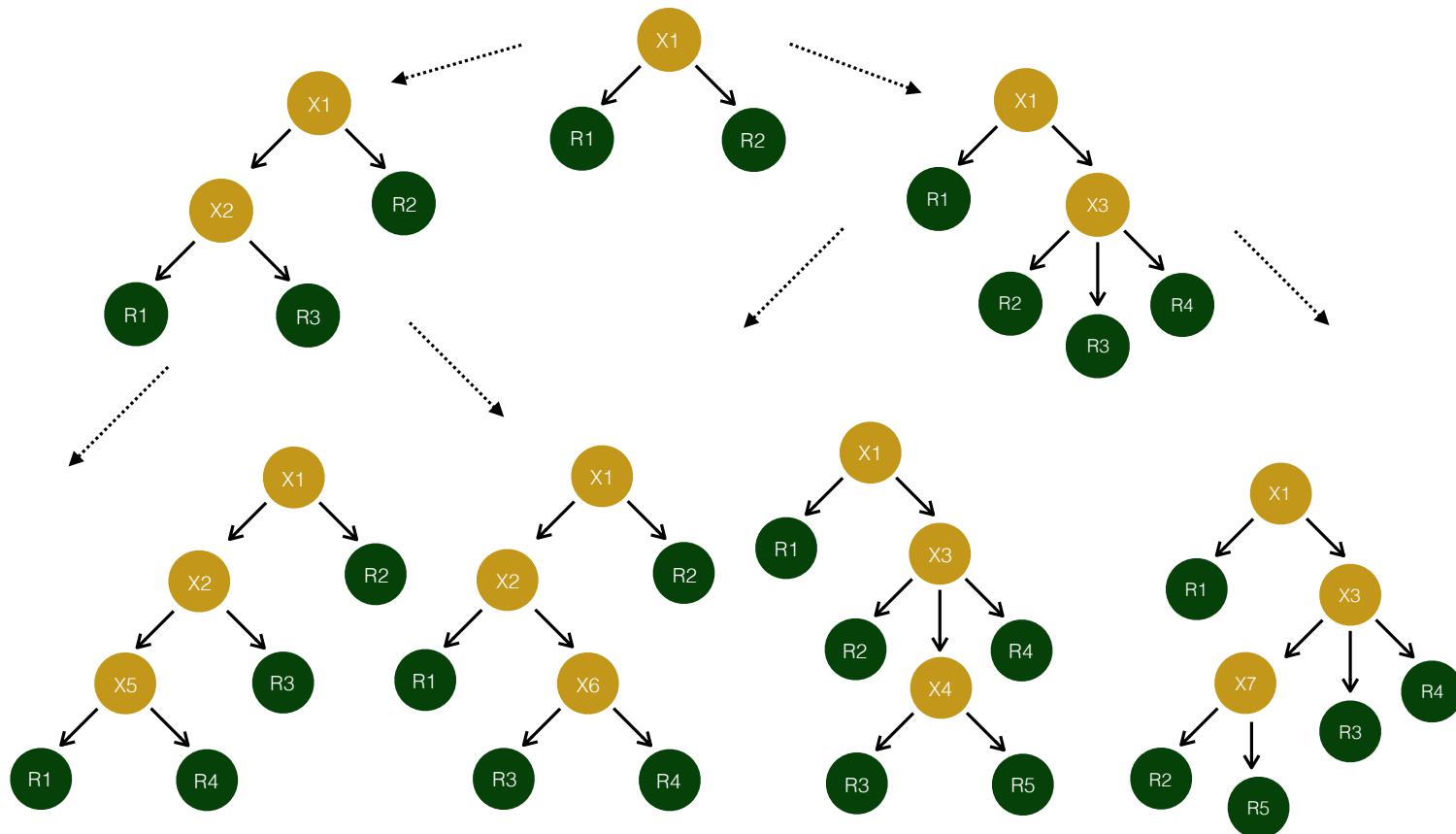
Top-Down + Greedy

- Para elegir un split, tomar la/una lista de movimientos posibles y elegir aquél que genera la mayor mejora en la función objetivo

$$J = \sum_{\ell=1}^n L(f(x^{(\ell)}), y^{(\ell)})$$

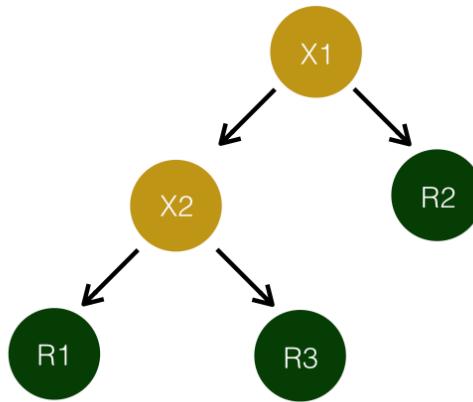


Top-Down + Greedy

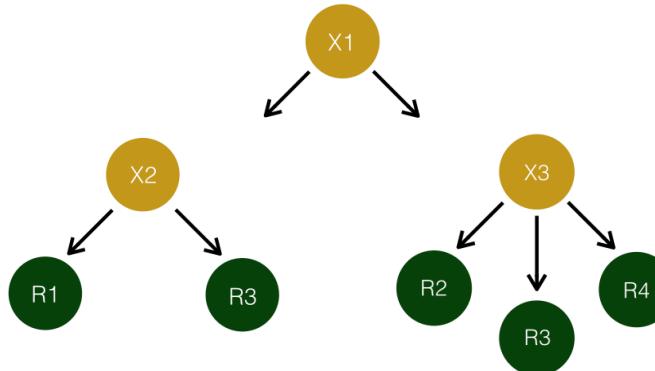


Top-Down + Greedy

crecimiento des-balanceado

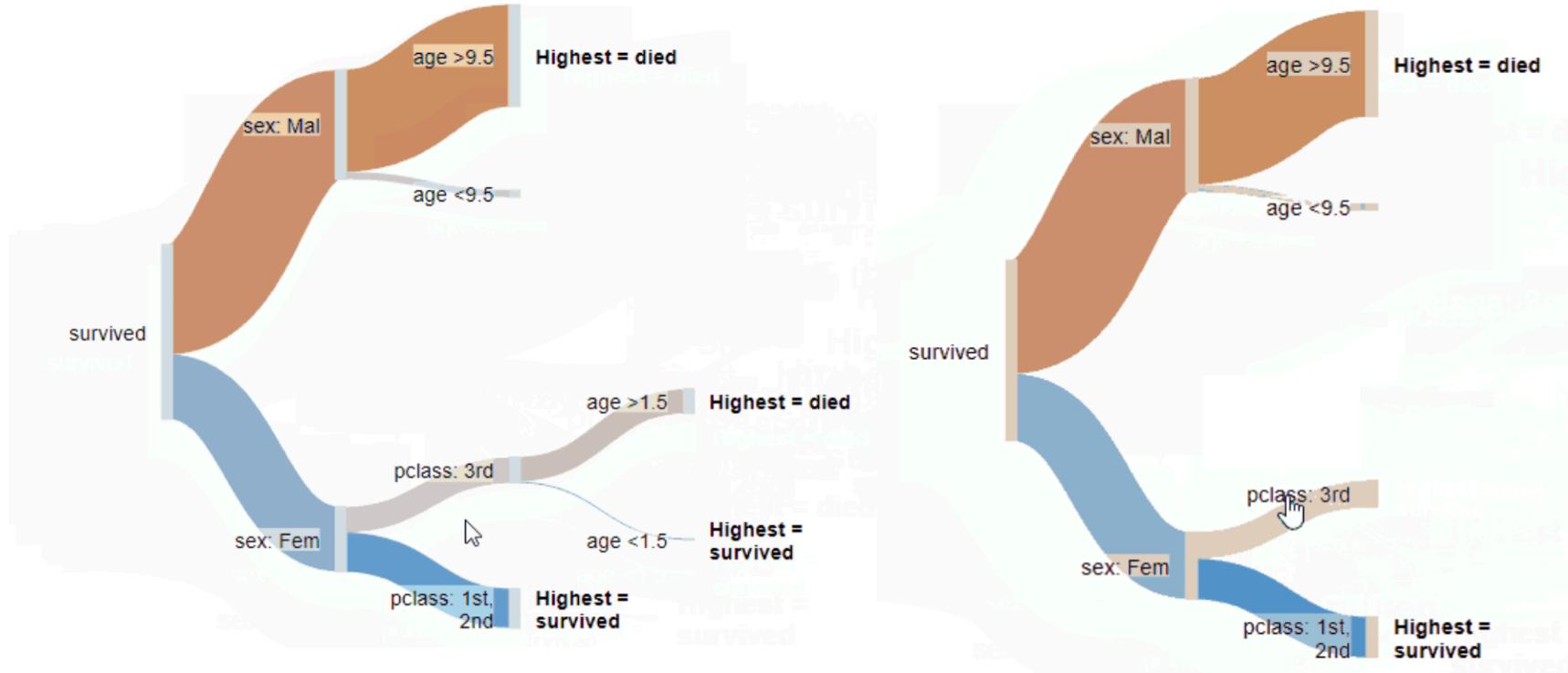


crecimiento balanceado



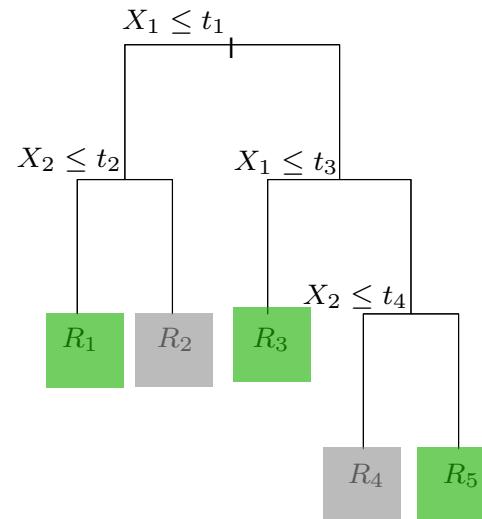
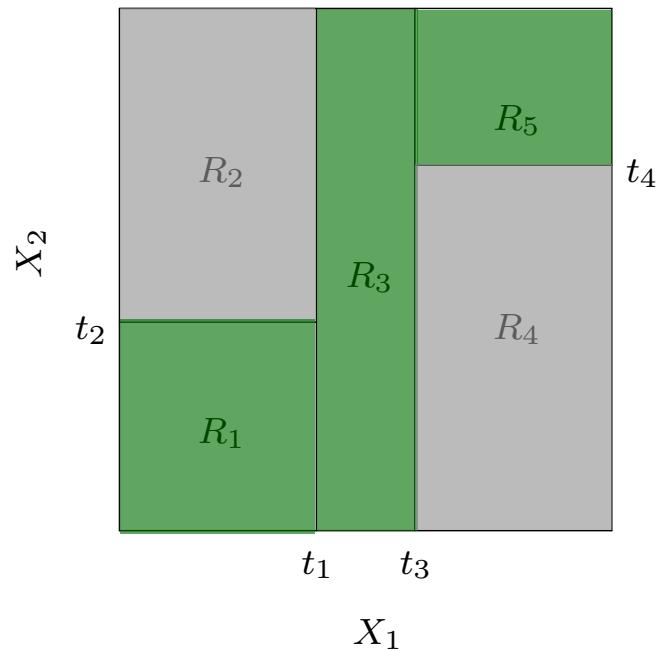
Poda

- Estrategia C4.5 + CART: Crecer de modo promiscuo (forward pass) y aplicar una segunda fase de poda (backward pass).



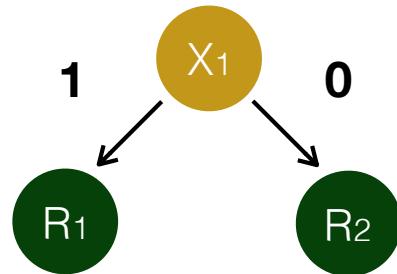
Criterio de Parada

- Continuar ramificando hasta que la función objetivo no mejora significativamente o hasta que se alcanza una profundidad máxima o hasta que las regiones correspondientes no tienen soporte suficiente.



Splits

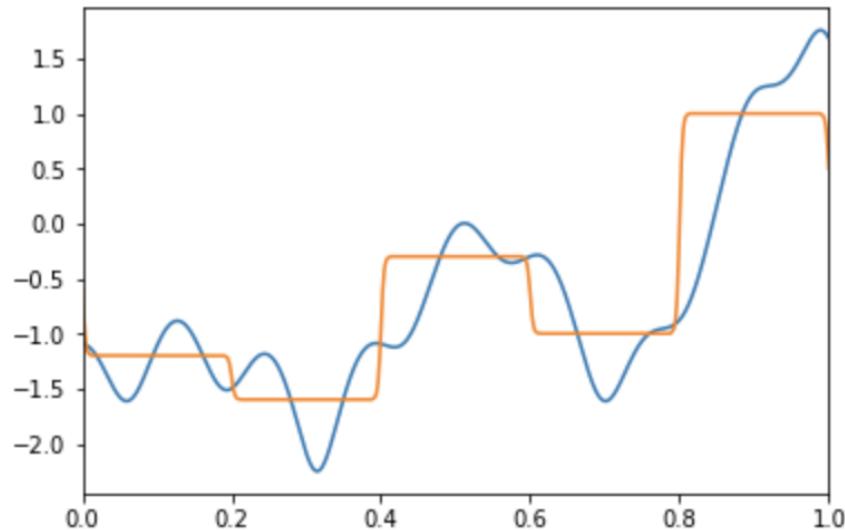
- **ID3 ó C4.5: Splits multivariados.** Cada rama indica un valor posible de la variable (categórica) o un rango posible.
- **CART: Splits binarios.** Rama derecha indica que se satisface la condición. Rama izquierda que no se satisface.



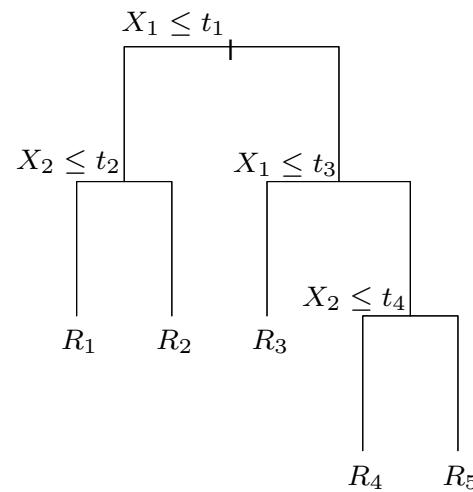
$$I(x_j \leq s) \quad b \in \mathbb{R}$$

Hojas en Regresión

- **Predicción** constante en cada región.



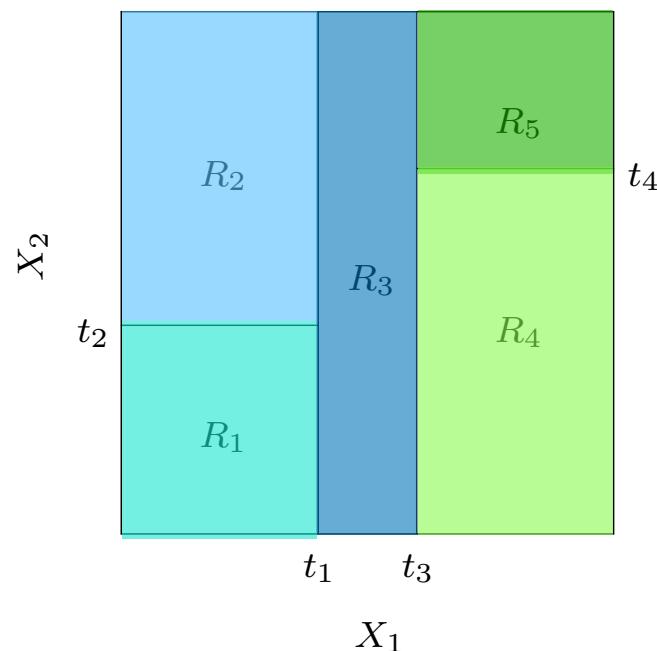
$$f(x) = \sum_{k=1}^K I(x \in R_k) c_k$$



Descomponibilidad del Error

- **¿Qué constante?** Como un dato pertenece a 1 y sólo 1 región, es fácil demostrar que

$$J = \sum_{\ell=1}^n L(f(x^{(\ell)}), y^{(\ell)}) = \sum_{k=1}^K L_k$$

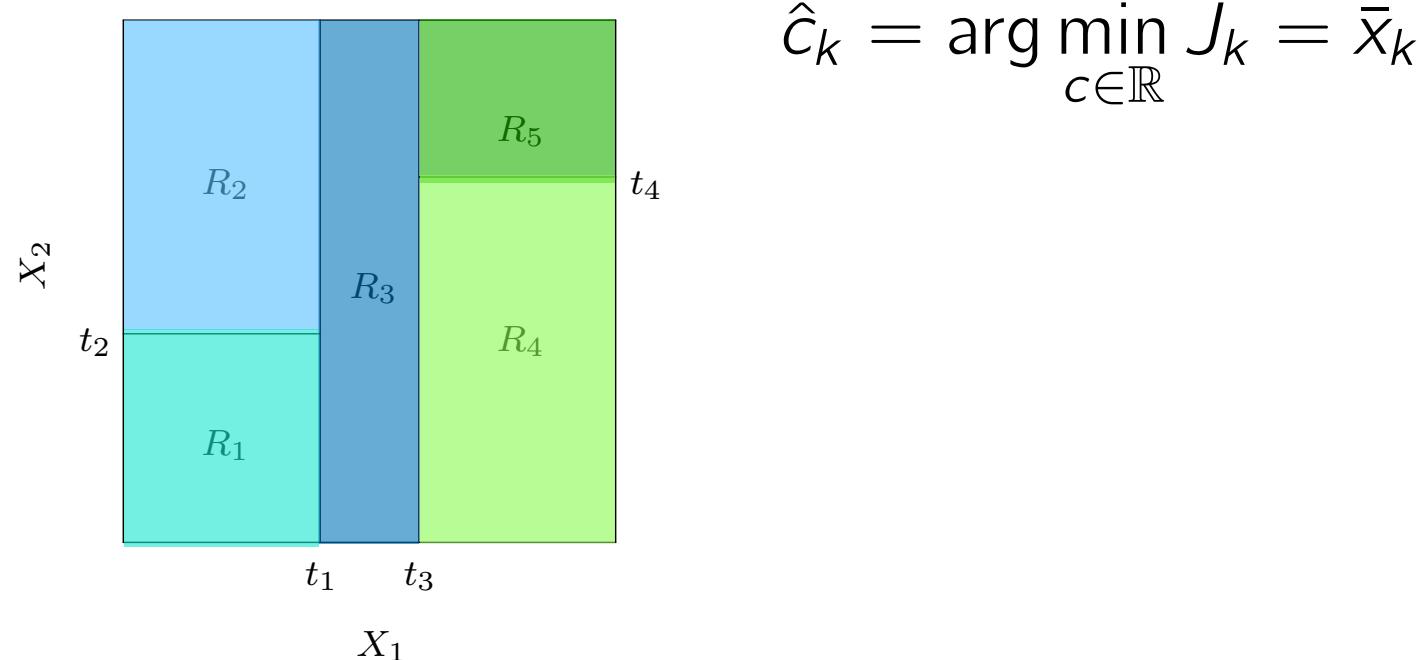


$$L_k = \sum_{\ell \in R_k}^n L(f(x^{(\ell)}), y^{(\ell)})$$

Descomponibilidad del Error

- **¿Qué constante?** Por lo tanto, podemos determinar la mejor predicción *localmente*. Por ejemplo, si

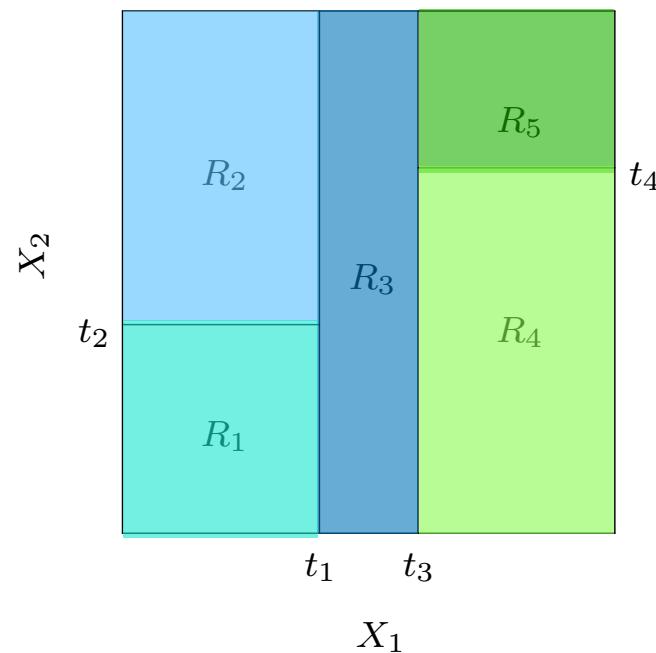
$$L(f(x^{(\ell)}), y^{(\ell)}) = \|f(x^{(\ell)}) - y^{(\ell)}\|^2$$



Descomponibilidad del Error

- La elección localmente óptima es globalmente óptima.

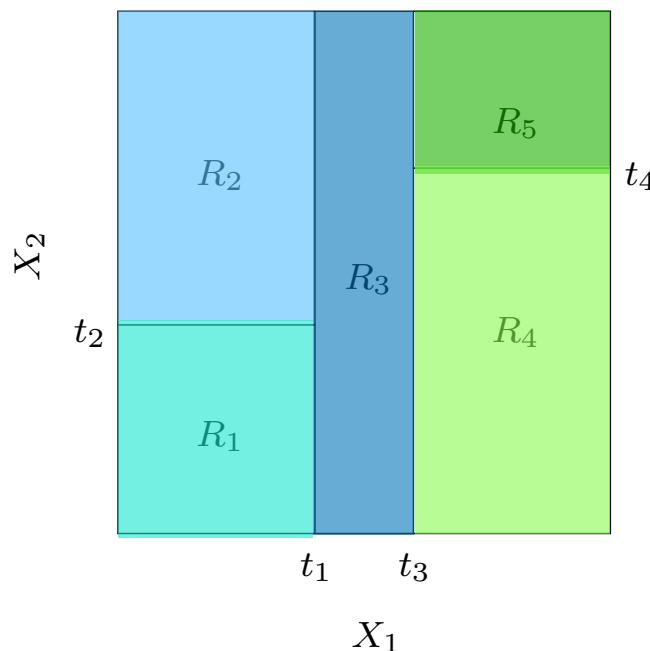
$$\hat{c}_1, \hat{c}_2, \dots, \hat{c}_K = \arg \min_{c_1, c_2, \dots, c_K \in \mathbb{R}} J$$



Descomponibilidad del Error

- Para otras funciones de error (no cuadrática) será otra la predicción óptima, pero seguiremos pudiendo elegir localmente.

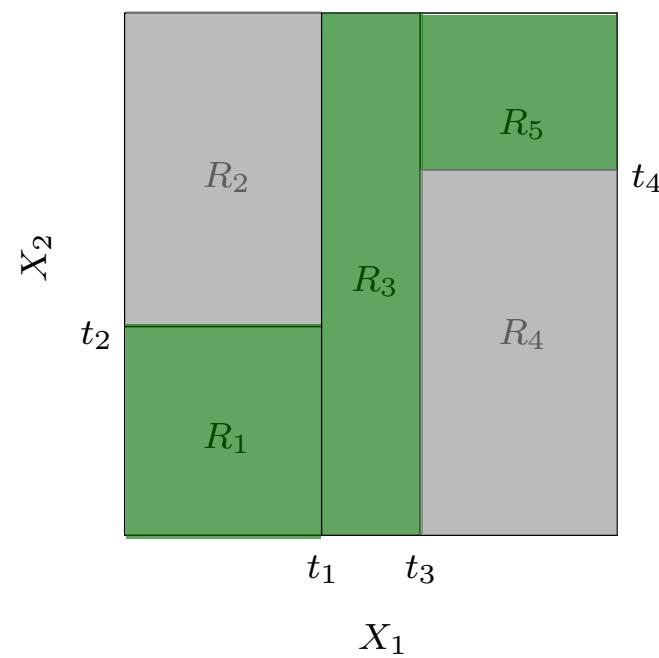
$$J = \sum_{\ell=1}^n L(f(x^{(\ell)}), y^{(\ell)}) = \sum_{k=1}^K L_k$$



$$L_k = \sum_{\ell \in R_k}^n L(f(x^{(\ell)}), y^{(\ell)})$$

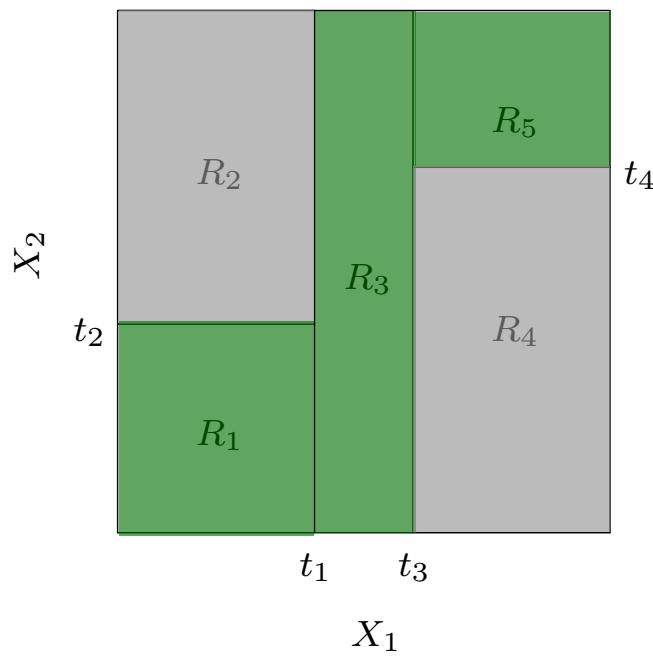
Hojas en Clasificación

- Clase?

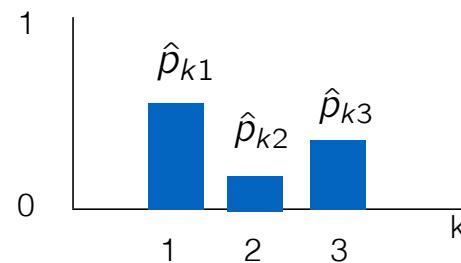


Hojas en Clasificación

- Una idea mucho mejor (al menos durante el entrenamiento) es manejar un estimador de la distribución de probabilidad sobre las clases.



predictor local:

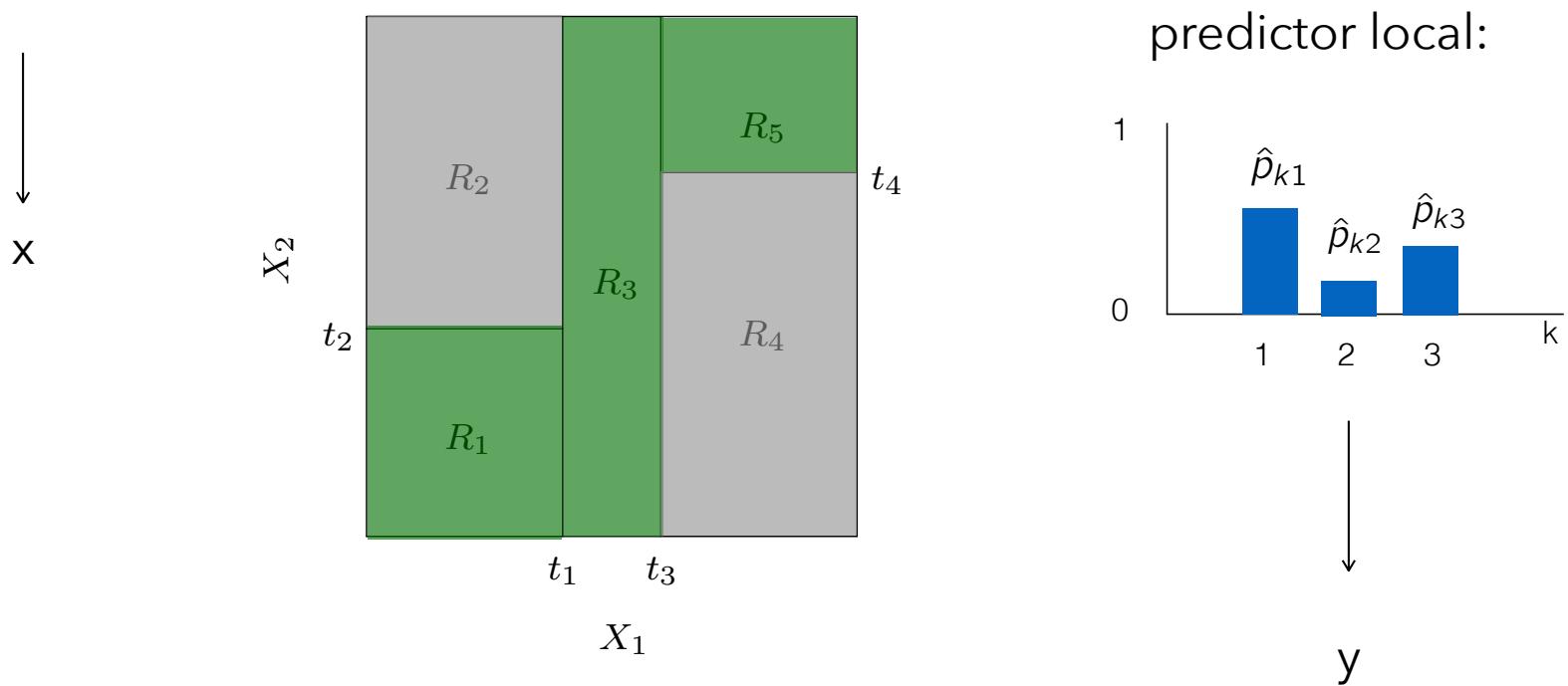


predictor global:

$$\hat{p}(y = j|x) = \hat{p}_j = \sum_{k=1}^K I(x \in R_k) \hat{p}_{kj}$$

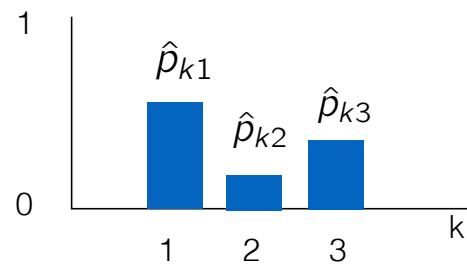
Hojas en Clasificación

- Si queremos un estimador puntual (y queremos minimizar a probabilidad de error) basta tomar la moda de la distribución.

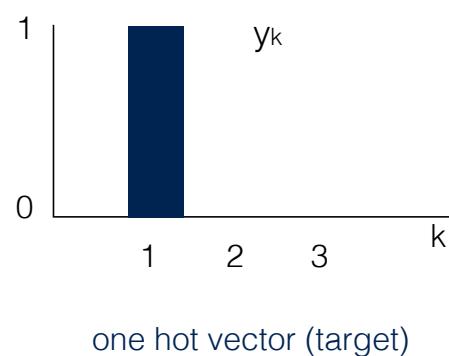


Hojas en Clasificación

- Qué **estimador local**? Nuevamente por la "descomponibilidad" de la f.o. basta encontrar el óptimo local de la función de costo que se deseé optimizar.



- Por ejemplo (CART) podríamos decidir minimizar la divergencia KL entre el estimador y la distribución "real" de los datos.



$$J = \sum_{\ell=1}^n D_{KL}(p(x^{(\ell)}) || \hat{p}(x^{(\ell)}))$$

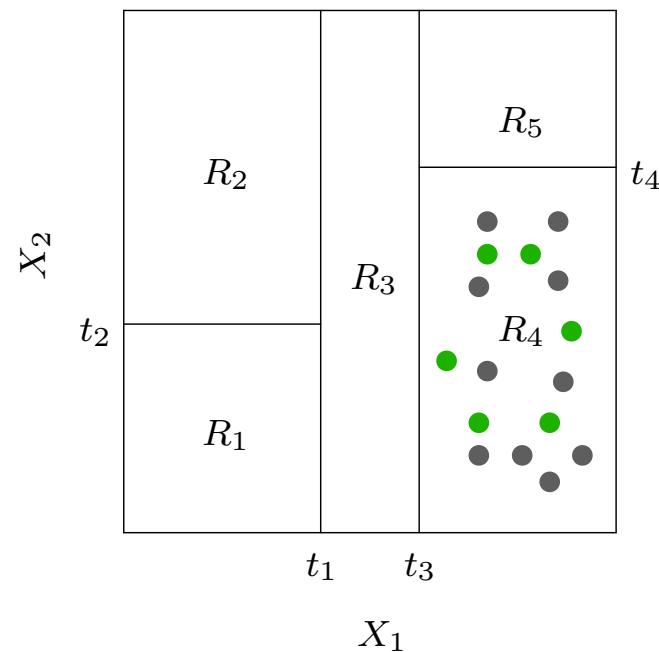
- En caso (usual) de etiquetado determinista, interpretamos la etiqueta como una distribución one-hot.

Hojas en Clasificación

- En este caso, no es difícil demostrar que el estimador local óptimo de $p(y = j | x)$ corresponde a la proporción de datos (de esa región) que pertenecen a la clase j .

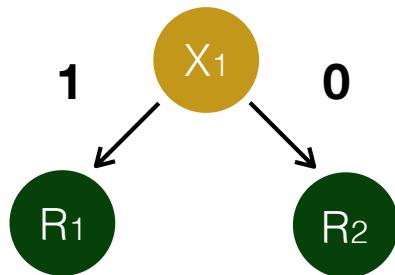
$$\hat{p}_{kj} = \frac{\sum_{\ell} I(x^{(\ell)} \in R_k) I(y^{(\ell)} = j)}{n}$$

$$\hat{p}_{kj} = \frac{n_{kj}}{n}$$



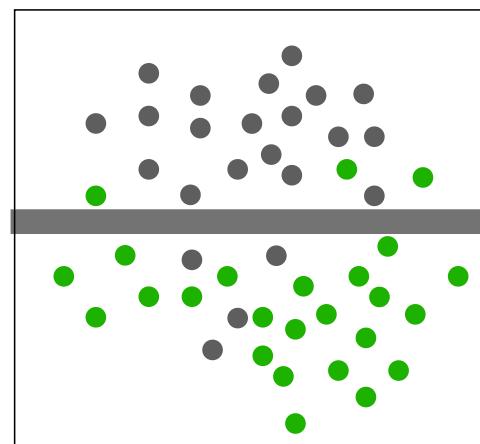
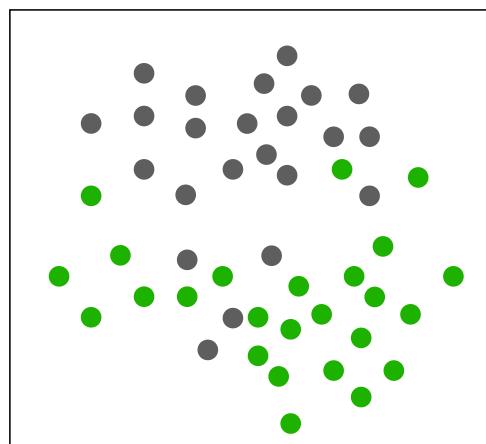
¿Cómo elegir el split?

- La elección de CART reduce la elección a una combinación de variable j y valor de corte s .

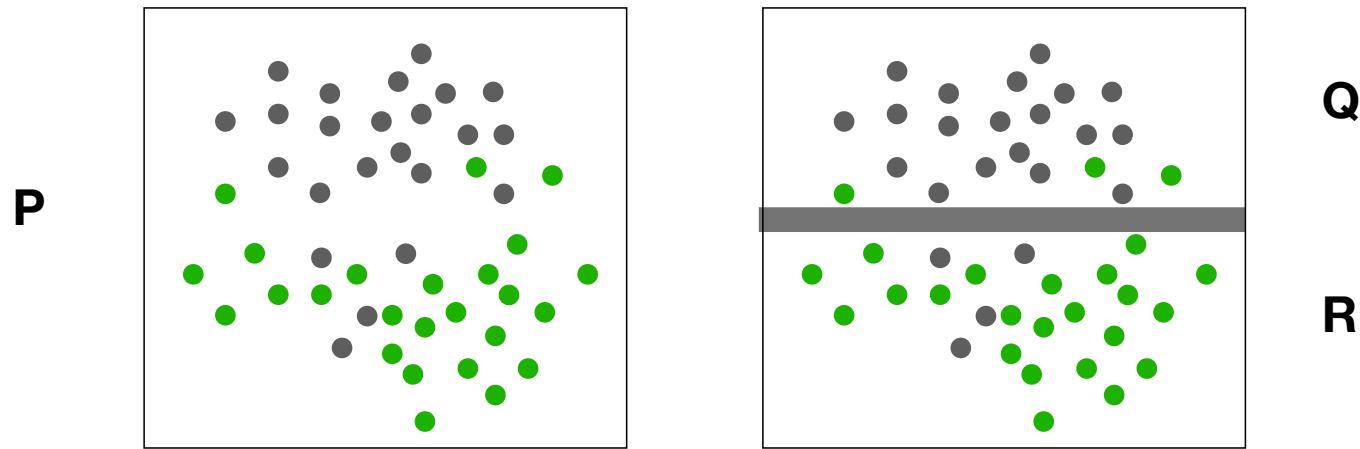


$$I(x_j \leq s) \quad b \in \mathbb{R}$$

- ¿Qué sucede la f.o. al dividir un nodo?



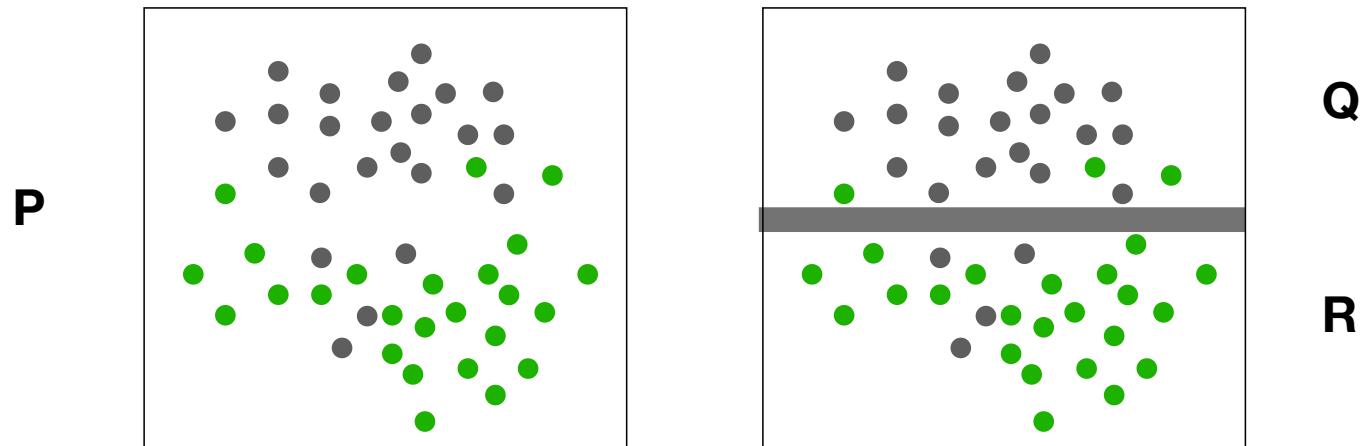
¿Cómo elegir el split?



- La división de una región en 2 **sólo puede mejorar** la f.o. (error de entrenamiento). En efecto, para $f(x)$ fija

$$\sum_{\ell \in P}^n L(f(x^{(\ell)}), y^{(\ell)}) = \sum_{\ell \in R}^n L(f(x^{(\ell)}), y^{(\ell)}) + \sum_{\ell \in S}^n L(f(x^{(\ell)}), y^{(\ell)})$$

¿Cómo elegir el split?

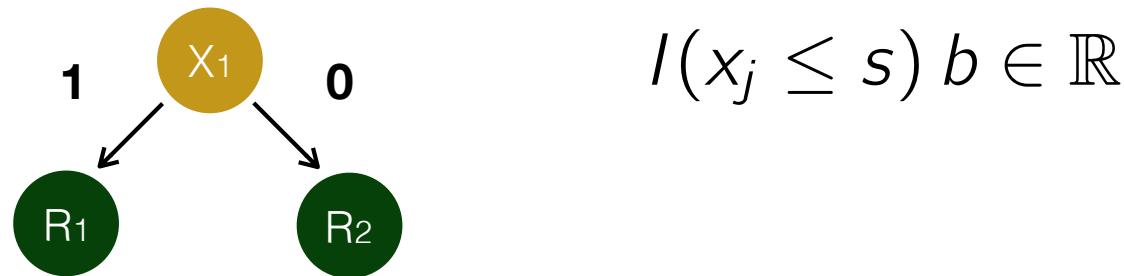


- Pero, como la división de la región cambia el predictor global, tenemos

$$\sum_{\ell \in P}^n L(f^{\text{old}}(x^{(\ell)}), y^{(\ell)}) \geq \sum_{\ell \in R}^n L(f^{\text{new}}(x^{(\ell)}), y^{(\ell)}) + \sum_{\ell \in S}^n L(f^{\text{new}}(x^{(\ell)}), y^{(\ell)})$$

¿Cómo elegir el split?

- Estrategia clásica (greedy): se evalúan todas las opciones (variable j + valor s) y se elige aquella que minimiza más la f.o.
- La generación de splits candidatos puede hacerse de modo relativamente eficiente gracias a la elección de CART.



- Como en el conjunto de ejemplos, los valores observados de la variable son finitos sólo tiene sentido explorar un número finito de valores (cuáles?)

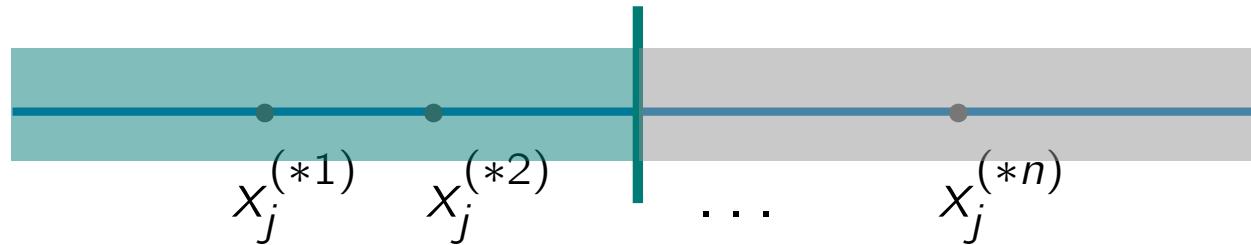
¿Cómo elegir el split?



- La decisión de qué split aplicar sobre una hoja tiene entonces costo $\mathcal{O}(nd)$: caro, pero manejable (hay métodos no exhaustivos).
- Elegir un split de la forma $I(t \leq x_j \leq s)$ eleva el costo a $\mathcal{O}(n^2d)$. Un split de esa forma se puede obtener en dos pasos de CART.



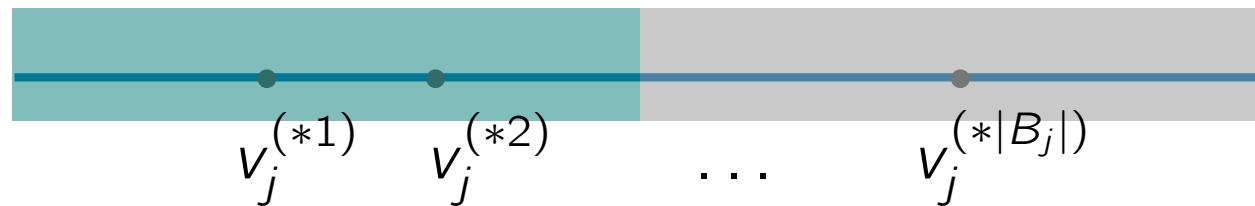
¿Cómo elegir el split?



- **Non-free lunch:** Es posible demostrar que, sin una simplificación sobre la forma del split (CART, ID3, C4.5) el problema de elegir el split óptimo (univariado y multivariado) es NP completo.
- En el caso de atributos discretos, CART aplica el mismo tipo de split, pero una pequeña modificación en la forma de generar los candidatos puede, con frecuencia, garantizar el **óptimo**.

¿Cómo elegir el split?

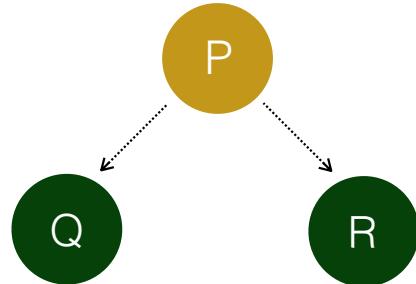
- Si x_j es categórica con valores, B_j , existen $2^{|B_j|-1} - 1$ posibles splits. Sin embargo, si y es continua y el objetivo es minimizar el error cuadrático, es posible demostrar que el/un split óptimo tiene siempre la forma



donde el orden se hace según el valor medio de y . Lo mismo vale en clasificación binaria con la gran mayoría de las funciones de costo empleadas en las prácticas.

Evaluación de un Split

- En problemas de regresión, la función más utilizada para medir el beneficio de un split es el error cuadrático (suma, no media)



$$\Delta = L_P - L_R - L_S$$
$$L_R = \sum_{\ell \in R}^n \|f(x^{(\ell)}), y^{(\ell)}\|^2$$

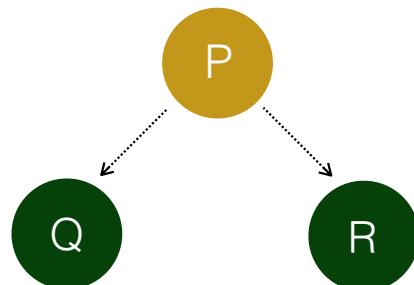
- Sin embargo, se han escrito muchos papers sobre qué criterio utilizar para medir el beneficio de un split en problemas de clasificación. Como hemos presentado la historia nosotros, ese criterio depende simplemente de la loss que se quiera optimizar.

Evaluación de un Split

- Si usamos la divergencia KL ó, equivalentemente, la función cross-entropy para implementar la f.o. de entrenamiento, es decir

$$J = \sum_{\ell}^n D_{KL}(y^{(\ell)} || \hat{p}(x^{(\ell)})) \propto - \sum_{\ell}^n \sum_j y_j^{(\ell)} \log \hat{p}_j(x^{(\ell)})$$

la reducción del error de entrenamiento viene dada por



$$\Delta = L_P - L_R - L_S$$
$$L_R = - \sum_{\ell \in R}^n \sum_j y_j^{(\ell)} \log \hat{p}_{rj}$$

(con r e índice de la región R).

Information Gain

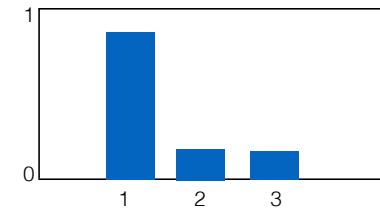
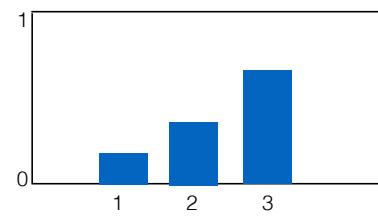
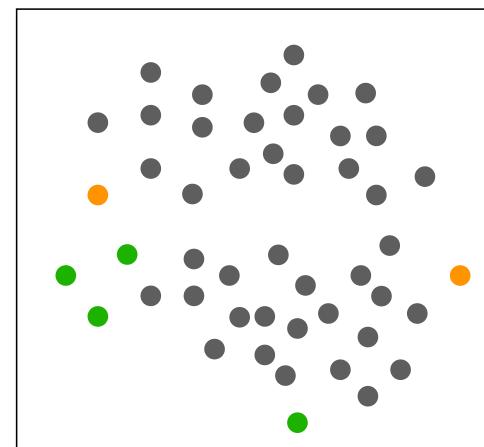
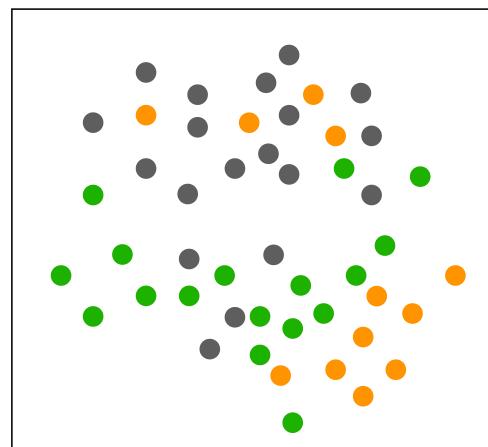
- Notando que el estimador es constante dentro de cada región

$$\begin{aligned}L_R &= - \sum_{\ell \in R}^n \sum_j y_j^{(\ell)} \log \hat{p}_{rj} = - \sum_j n_{rj} \log \hat{p}_{rj} \\&= -n_r \sum_j \frac{n_{rj}}{n_r} \log \hat{p}_{rj} \\&= -n_r \sum_j \hat{p}_{rj} \log \hat{p}_{rj} = n_r H_r\end{aligned}$$

donde H_r es la entropía de la distribución de probabilidad correspondiente a la región/hoja R.

Pureza e Information Gain

- La entropía mide qué tan aleatoria es la distribución de las clases en la región o bien (su negativo) qué tanta información sobre la clase entrega conocer la región donde cae un punto.



Information Gain

- Reemplazando el resultado en la ecuación que mide el beneficio del split (reducción de la f.o.), obtenemos

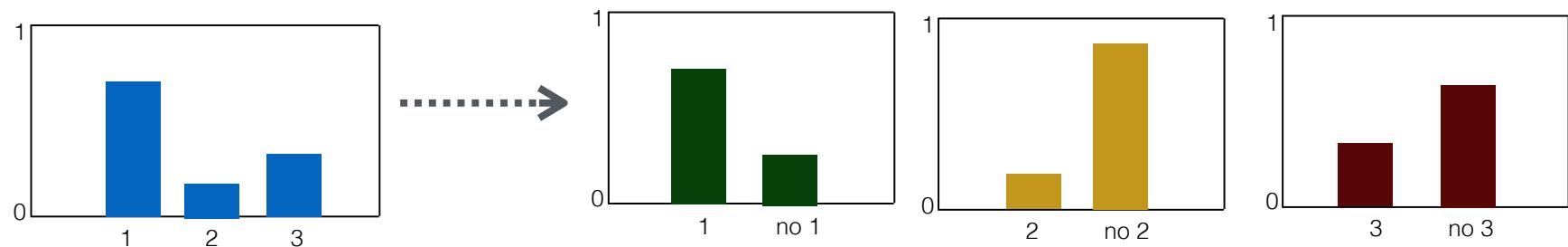
$$\begin{aligned}\Delta &= L_P - L_R - L_S \\ &= n_p H_p - (n_r H_r + n_s H_s)\end{aligned}$$

expresión que se conoce como information gain en la literatura de árboles (y que en muchas ocasiones se usa, de modo equivocado, sin el escalamiento proporcional al número de puntos en cada región).

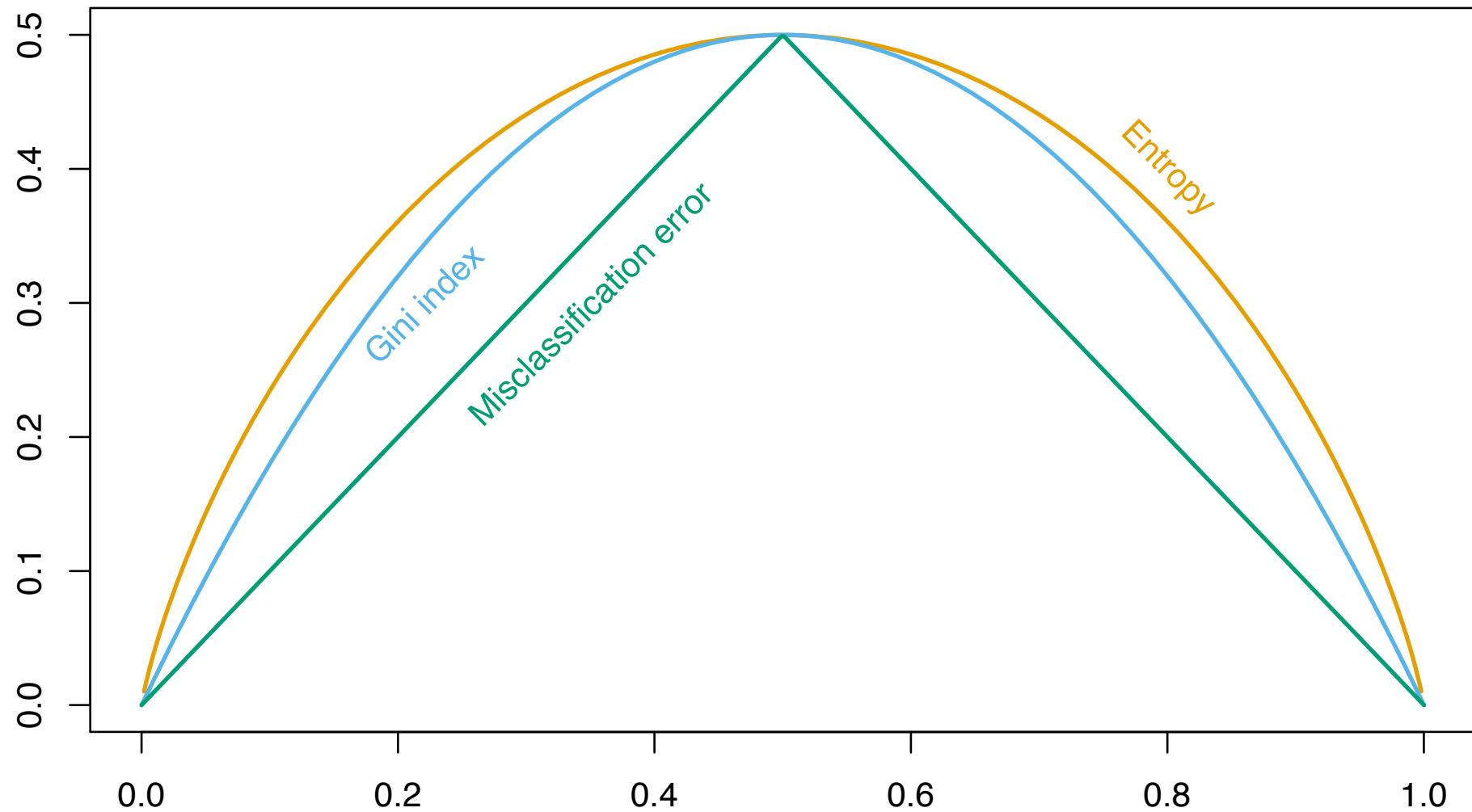
Gini Index

- Otro criterio históricamente relevante (primeros árboles) para medir la pureza de una región y luego el beneficio de un split es el índice de Gini:

$$L_R = \sum_j \hat{p}_{rj}(1 - \hat{p}_{rj})$$

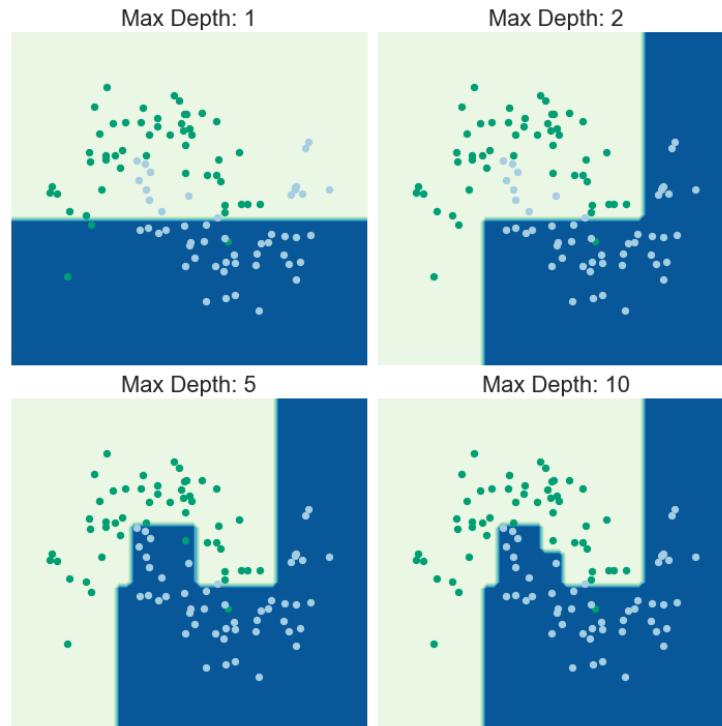


Gini Index



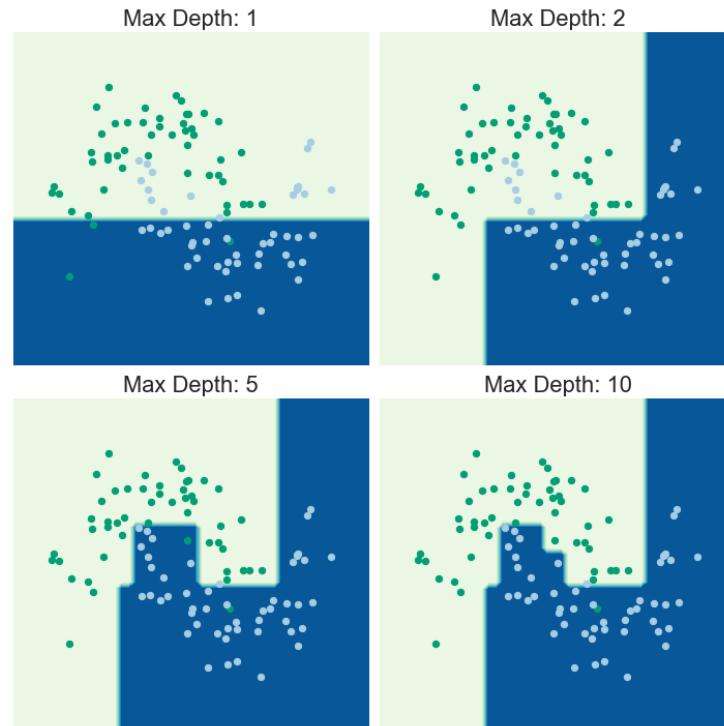
Poda

- Seguramente una componente de los algoritmos actuales más relevante es el mecanismo de poda.
- La poda es un mecanismo que se usa para reducir la profundidad de un árbol: por lejos el mecanismo más utilizado para evitar overfitting en este tipo de modelos.
- El método empleado por CART es particularmente poderoso.



Poda

- Mala noticia: encontrar el árbol más compacto equivalente a un determinado árbol entrenado es un problema NP-completo.
- Buena noticia: El método empleado por CART es particularmente eficiente y poderoso.



Poda & Regularización

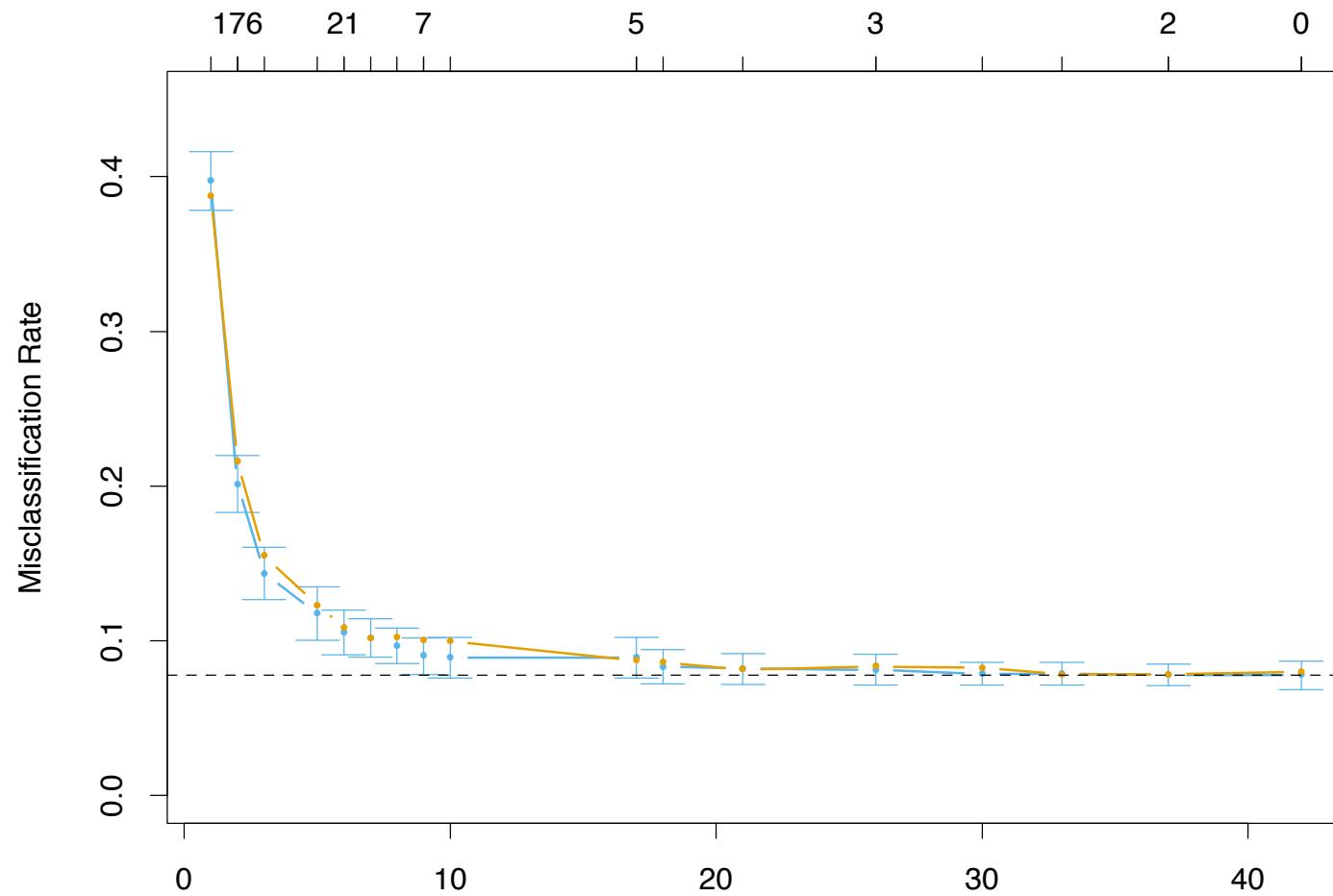
- Desde una óptima muy tradicional, el problema de **cuánto podar** un árbol se podría plantear como el de resolver una f.o. regularizada. Por ejemplo:

$$J_\alpha = J + \alpha|T|$$

con $|T|$ a profundidad del árbol y α el parámetro de regularización.

- Podríamos intentar modificar el algoritmo de crecimiento para considerar el regularizador (nada fácil porque $|T|$ no es local); ejecutar con muchos valores de α y luego usar cross-validation (u otro método) para elegir el modelo que generaliza mejor.

Poda & Regularización



Poda & Regularización

- CART no hace eso (muy costoso). Se demuestra en cambio el siguiente teorema:

Teorema

Sea T_α el sub-árbol de T que optimiza J_α . Supongamos que tomamos el árbol entrenado y ejecutamos una algoritmo greedy bottom-up que en cada iteración elige exactamente 1 nodo interno para transformar en hoja y que el nodo elegido es aquél que causa el menor empeoramiento de la f.o. Sea $\tilde{T}_1, \tilde{T}_2, \dots, \tilde{T}_m$ la secuencia de árboles obtenidos. Entonces,

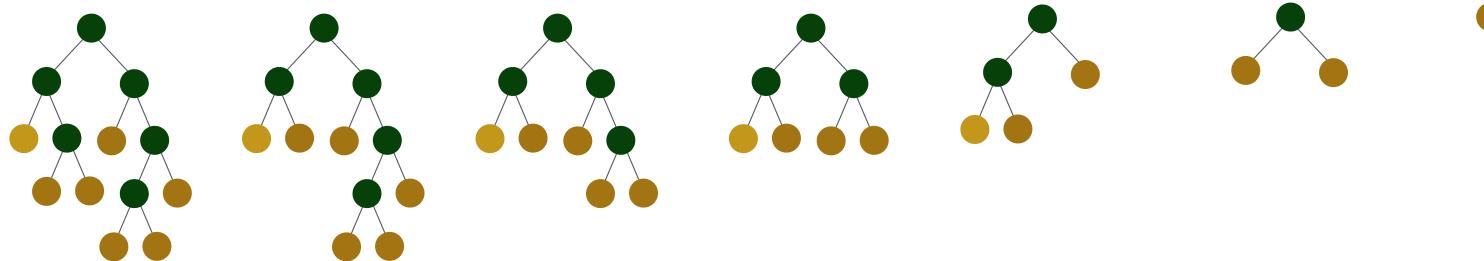
$$\forall \alpha \geq 0, \exists i : \tilde{T}_i = T_\alpha$$

Poda & Regularización

Teorema

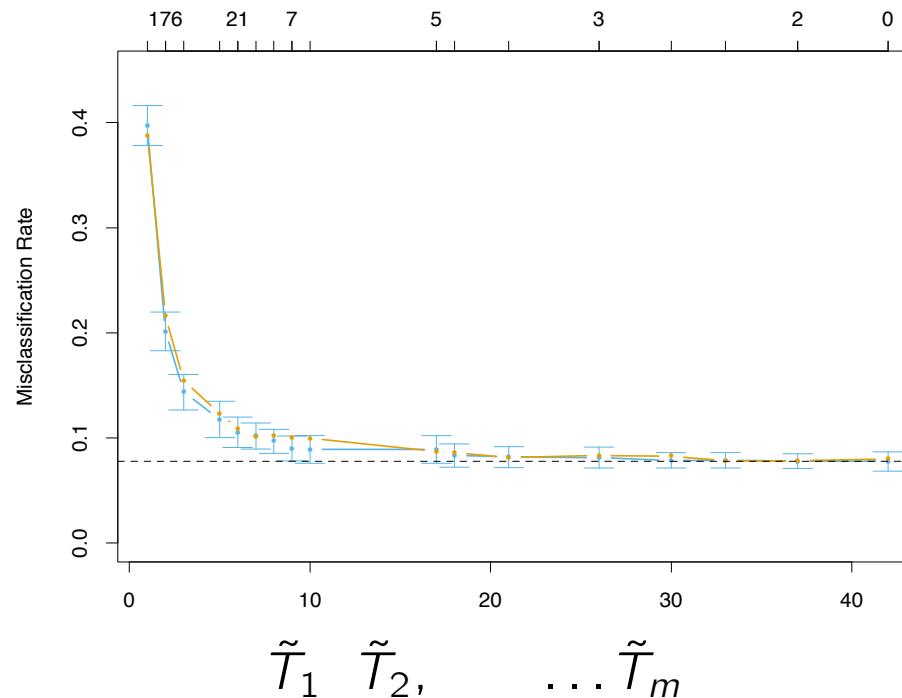
Sea T_α el sub-árbol de T que optimiza J_α . Supongamos que tomamos el árbol entrenado y ejecutamos una algoritmo greedy bottom-up que en cada iteración elige exactamente 1 nodo interno para transformar en hoja y que el nodo elegido es aquél que causa el menor empeoramiento de la f.o. Sea $\tilde{T}_1, \tilde{T}_2, \dots, \tilde{T}_m$ la secuencia de árboles obtenidos. Entonces,

$$\forall \alpha \geq 0, \exists i : \tilde{T}_i = T_\alpha$$



Poda & Regularización

- La consecuencia del teorema anterior es que no necesitamos modificar el algoritmo de crecimiento o considerar, como usualmente un rango de valores de α . Basta obtener la secuencia de subárboles y elegir entre ellos.



Resumen

- Partición recursiva del espacio de características.
- Las hojas identifican regiones; los nodos internos pruebas sobre un atributo y ramas resultados de una prueba.
- A cada hoja le corresponde una decisión.
- Manejan naturalmente variables continuas, categóricas y datos faltantes.
- Altamente interpretables.
- Aproximadores universales.
- Sensibles a overfitting.