Introducción a la Ciencia de Datos

Maestría en Ciencias de la Computación

Dr. Irvin Hussein López Nava

04

Clasificación

4.2 Decision tree learning


¿Qué son los Árboles de Decisión?

Definiciones

ChatGPT

Son un tipo de modelo de aprendizaje automático que se utiliza para tareas de clasificación. Son representaciones gráficas en forma de árbol en las que se toman decisiones basadas en la información contenida en los datos.

https://chat.openai.com/

Wikipedia

Es un modelo jerárquico que utiliza un modelo de decisiones similar a un árbol y sus posibles consecuencias, incluidos los resultados de eventos casuales, los costos de los recursos y la utilidad. Es una forma de mostrar un algoritmo que solo contiene declaraciones de control condicionales.

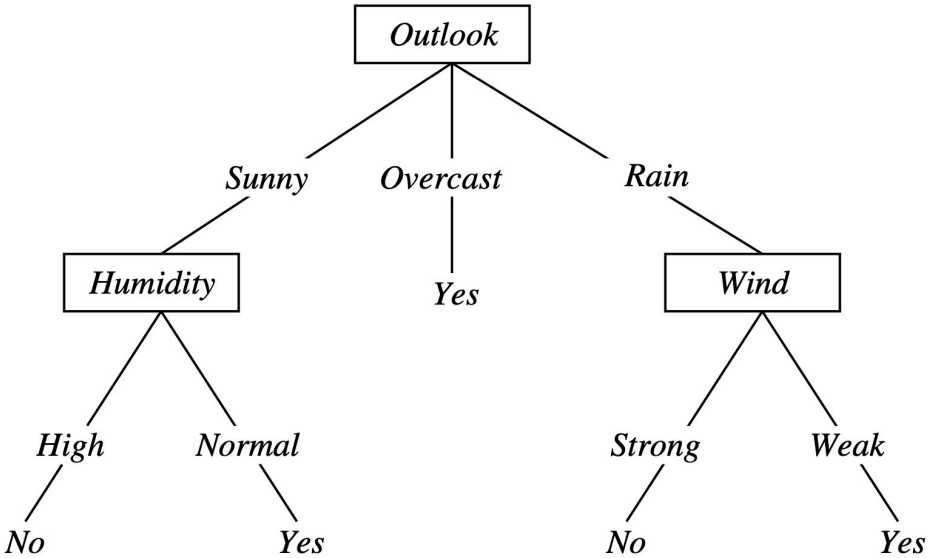
https://en.wikipedia.org/wiki/Decision_tree

Gemini

Son un tipo de algoritmo de aprendizaje supervisado que se utiliza para clasificación. Son modelos de predicción que se construyen a partir de una serie de reglas. Cada regla se divide en dos o más ramas, y cada rama representa un resultado posible.

https://gemini.google.com/

Play tennis


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Decision tree

- Un **árbol de decisión** representa una función que toma como entrada un vector de valores de atributos y devuelve una "decisión", i.e., un único valor de salida.
- Los valores de entrada y salida pueden ser **discretos** o **continuos**.
- Representación de un árbol de decisión:
  - Cada **nodo interno** prueba un atributo
  - Cada **rama** corresponde a un valor de atributo
  - Cada **nodo hoja** asigna una clasificación

¿Cuándo considerar los árboles de decisión?

- Instancias descriptibles por pares atributo-valor.
- La función objetivo es discreta.
- Puede ser necesaria una hipótesis disyuntiva.
- Datos de entrenamiento posiblemente ruidosos.

  Ejemplos:
  - Diagnóstico médico.
  - Análisis de riesgo de crédito y detección de fraudes.
  - Predicción de retención de empleados.
  - Recomendación de productos.

Árboles de decisión

- El algoritmo básico de aprendizaje de árboles de decisión, como el ID3 (Quinlan 1986) y su sucesor C4.5 (Quinlan 1993), utiliza una búsqueda codiciosa (*greedy*) de arriba hacia abajo a través del espacio de posibles árboles de decisión.
- Este enfoque comienza con la pregunta de qué atributo debe ser probado en la raíz del árbol.
    - Para decidir esto, cada atributo se evalúa mediante una prueba estadística para determinar qué tan bien clasifica los ejemplos de entrenamiento.

Árboles de decisión

- El mejor atributo se selecciona para ser probado en la raíz del árbol.
- Después, se crea un nodo descendiente para cada valor posible de este atributo, y los ejemplos de entrenamiento se distribuyen al nodo correspondiente según su valor.
- Este proceso se repite recursivamente en cada nodo descendiente, seleccionando el mejor atributo en cada punto del árbol, lo que constituye una búsqueda *greedy* sin retroceso para construir un árbol de decisión aceptable.

Inducción *Top-down*

Main loop:

1. **A** ← the “best” decision attribute for next node.
2. Assign **A** as decision attribute for node.
3. For each value of **A**, create new descendant of node.
4. Sort training examples to leaf nodes.
5. If training examples perfectly classified, Then STOP.
   Else iterate over new leaf nodes.

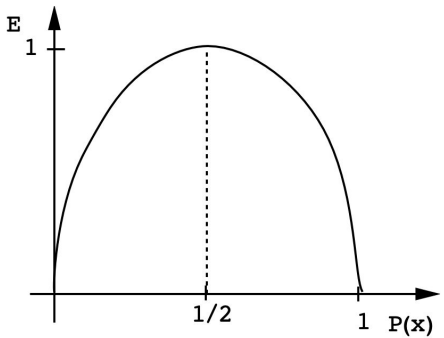
¿Cuál atributo es el mejor clasificador?

- La elección principal en el algoritmo ID3 es seleccionar qué atributo probar en cada nodo del árbol.
- Se desea elegir el atributo más útil para clasificar los ejemplos, ¿cuál es una buena medida cuantitativa del valor de un atributo?
- La propiedad estadística llamada **ganancia de información**, mide qué tan bien un atributo separa los ejemplos de entrenamiento según su clasificación objetivo.
- ID3 utiliza esta medida para seleccionar entre los atributos candidatos en cada paso mientras construye el árbol.

Entropía

- **S** es una muestra de ejemplos de entrenamiento.
- **S+** es la proporción de ejemplos positivos en S.
- **S–** es la proporción de ejemplos negativos en S.
- La **entropía** mide la impureza de S.

$$Entropy(S) \equiv -S_{\oplus} \log_2 S_{\oplus} - S_{\ominus} \log_2 S_{\ominus}$$



Aplicando la Entropía

- *Entropy(S)* = número esperado de bits necesarios para codificar la clase *(S+ o S−)* de un atributo de *S* extraído aleatoriamente.
- Teoría de la información: el código de longitud óptima asigna bits $-log_2 p$ al mensaje que tiene probabilidad *p*.

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \ log_2 \ p_i$$

Aplicando la Entropía

- Para ilustrar esto, suponga que *S* es una colección de 14 ejemplos de algún concepto booleano, incluidos 9 ejemplos positivos y 5 negativos (adoptando la notación [9+, 5−] para resumir dicha muestra de datos).
- Entonces, la entropía de *S* relativa a esta clasificación booleana es

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$Entropy([9+, 5-]) = -(9/14) \log_2(9/14) - (5/14) \log_2(5/14) = 0.940$$

Aplicando la Entropía

- Observe que la entropía es 0 si todos los miembros de *S* pertenecen a la misma clase.
- Por ejemplo, si todos los miembros son positivos *(p+ = 1)*, entonces *p−* es 0, y $Entropy(S) = -1 * log_2(1) - 0 * log_2(0) = -1 * 0 - 0 * log_2(0) = 0$.
- Notar además que la entropía es 1 cuando la colección contiene un número igual de ejemplos positivos y negativos.
- Si la colección contiene cantidades desiguales de ejemplos positivos y negativos, la entropía está entre 0 y 1.

Ganancia de información

- *Gain(S, A)* = reducción esperada de entropía debido a la clasificación en A causada por la partición de los ejemplos según este atributo.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

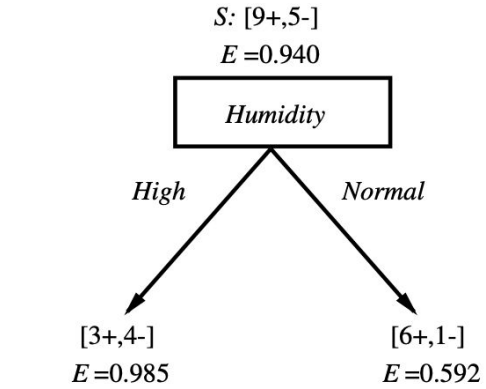
  donde *Values(A)* es el conjunto de todos los valores posibles para el atributo A, y S, es el subconjunto de S para el que el atributo A tiene valor *v*.
- Note que el primer término de la ecuación es la entropía de la colección original *S*, y el segundo término es el valor esperado de la entropía después de que *S* se particione utilizando el atributo *A*.

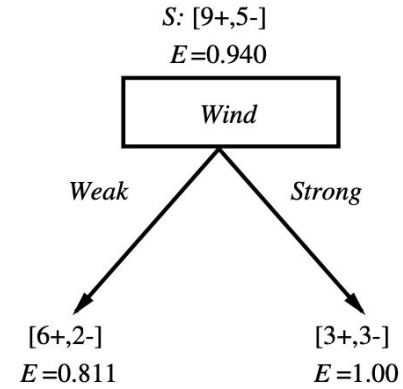
Attributes

day	outlook	temp	humidity	wind	play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Seleccionar el atributo siguiente

- ¿Cuál es el mejor atributo para clasificar?





Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

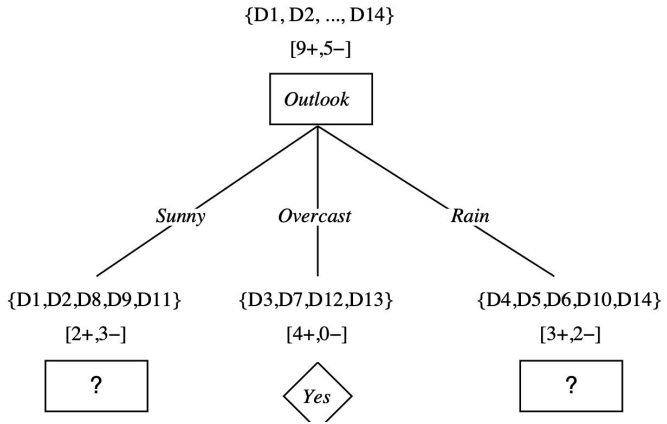
= .940 - (8/14).811 - (6/14)1.0
= .048

Outlook?

Temperature?

day	outlook	temp	humidity	wind	play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

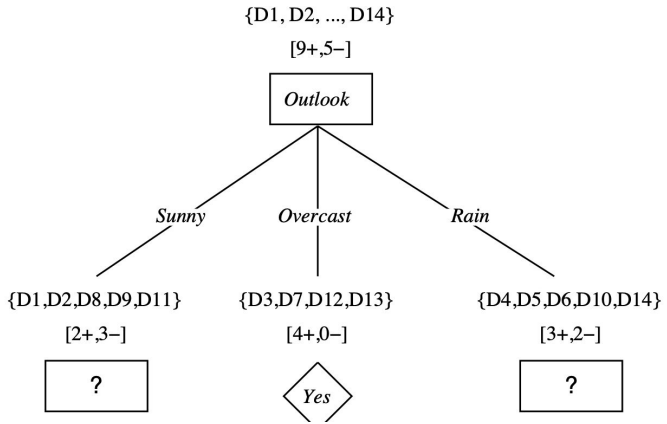
¿y el que sigue?



S_{Sunny} = {D1, D2, D8, D9, D11}

- Gain(S_{Sunny}, Temperature) =
- Gain(S_{Sunny}, Humidity) =
- Gain(S_{Sunny}, Wind) =

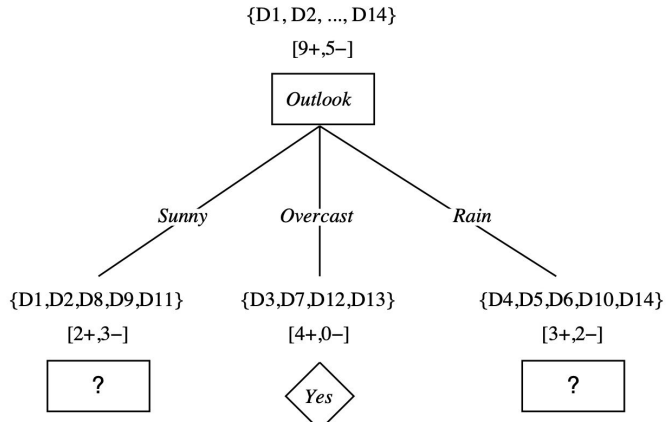
¿y el que sigue?



S_{Sunny} = {D1, D2, D8, D9, D11}

- Gain(S_{Sunny}, Temperature) = 0.97 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = 0.57
- Gain(S_{Sunny}, Humidity) =
- Gain(S_{Sunny}, Wind) =

¿y el que sigue?

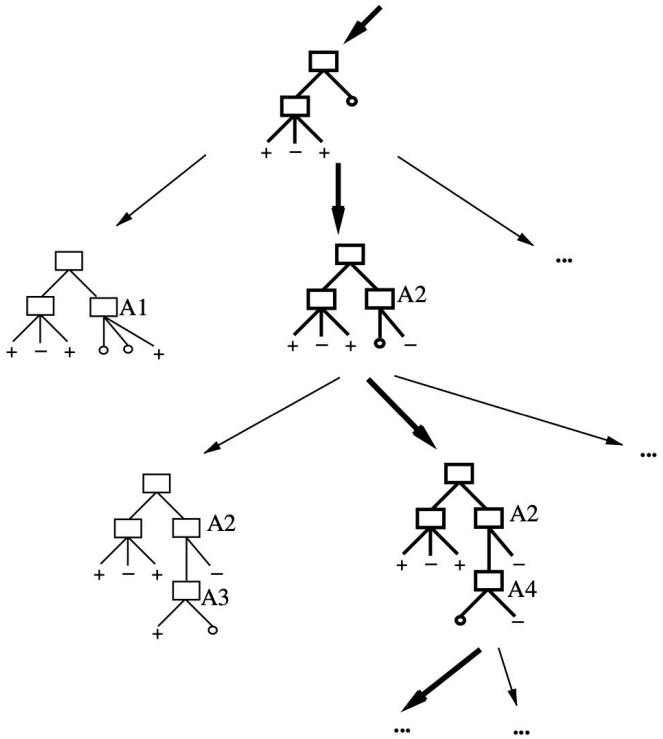


Rain

S_{Sunny} = {D1, D2, D8, D9, D11}

- Gain(S_{Sunny}, Temperature) = 0.97 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = 0.57
- Gain(S_{Sunny}, Humidity) = 0.97 - (3/5) 0.0 - (2/5) 0.0 = 0.97
- Gain(S_{Sunny}, Wind) = 0.97 - (2/5) 1.0 - (3/5) 0.91 = 0.019

Búsqueda de espacio de hipótesis por ID3



Algoritmo ID3

ID3 (Examples, Target_Attribute, Attributes)
Create a root node for the tree
If all examples are positive, **Return** the single-node tree Root, with label = +.
If all examples are negative, **Return** the single-node tree Root, with label = -.
If number of predicting attributes is empty, **then** Return the single node tree Root,
with label = most common value of the target attribute in the examples.
Otherwise Begin
A ← The Attribute that best classifies examples.
Decision Tree attribute for Root = A.
For each possible value, v_i, of A,
Add a new tree branch below Root, corresponding to the test A = v_i.
Let Examples(v_i) be the subset of examples that have the value v_i for A
If Examples(v_i) is empty
Then below this new branch add a leaf node with label = most common target value in the examples
Else below this new branch add the subtree ID3 (Examples(v_i), Target_Attribute, Attributes – {A})
End
Return Root

Búsqueda de espacio de hipótesis por ID3

- ¡El espacio de hipótesis está incompleto!
  - La función objetivo seguramente está allí...
- Genera una única hipótesis (¿cuál?)
  - No puedo manejar 20 preguntas...
- Sin retroceso
  - Se cae en mínimos locales...
- Opciones de búsqueda basadas en estadísticas
  - Datos resistentes al ruido...
- Sesgo inductivo: "preferir el árbol más corto"

Sesgo inductivo en ID3

Notar que H es el conjunto potencia de instancias X
→¿Imparcial?

No precisamente...

- Preferencia por árboles pequeños y por aquellos con atributos de alta ganancia de información cerca de la raíz.
- El sesgo es una preferencia por alguna hipótesis, en lugar de una restricción del espacio de hipótesis H.
- *Occam's razor*: preferir la hipótesis más corta que se ajuste a los datos.

¿Por qué preferir hipótesis pequeñas?

Argumento a favor:

- Hay menos hipótesis cortas que largas
- Es improbable que una hipótesis corta que se ajuste a los datos sea una coincidencia
    - Una hipótesis larga que se ajuste a los datos podría ser una coincidencia.

Argumento opuesto:

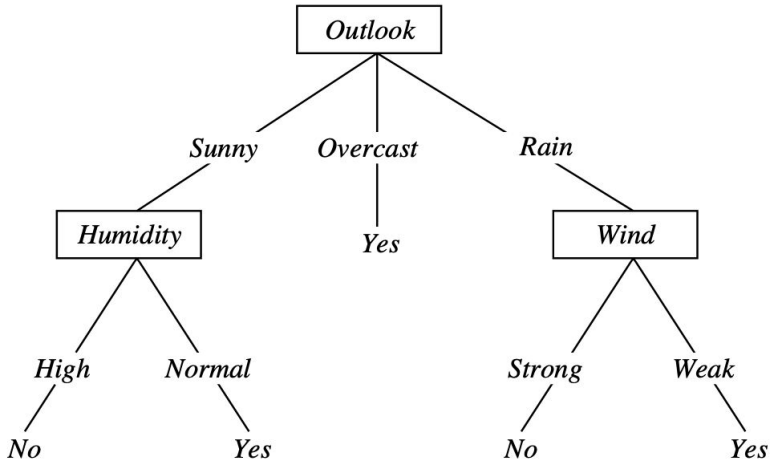
- Hay muchas formas de definir conjuntos pequeños de hipótesis
- ¿Qué tienen de especial los conjuntos pequeños basados en el tamaño de la hipótesis?

Sobreajuste en árboles de decisión

Considere agregar un ejemplo de entrenamiento **ruidoso** (n.º 15):

Sunny, Hot, Normal, Strong, Play = No

¿Qué efecto tiene sobre el árbol anterior?



Sobreajuste

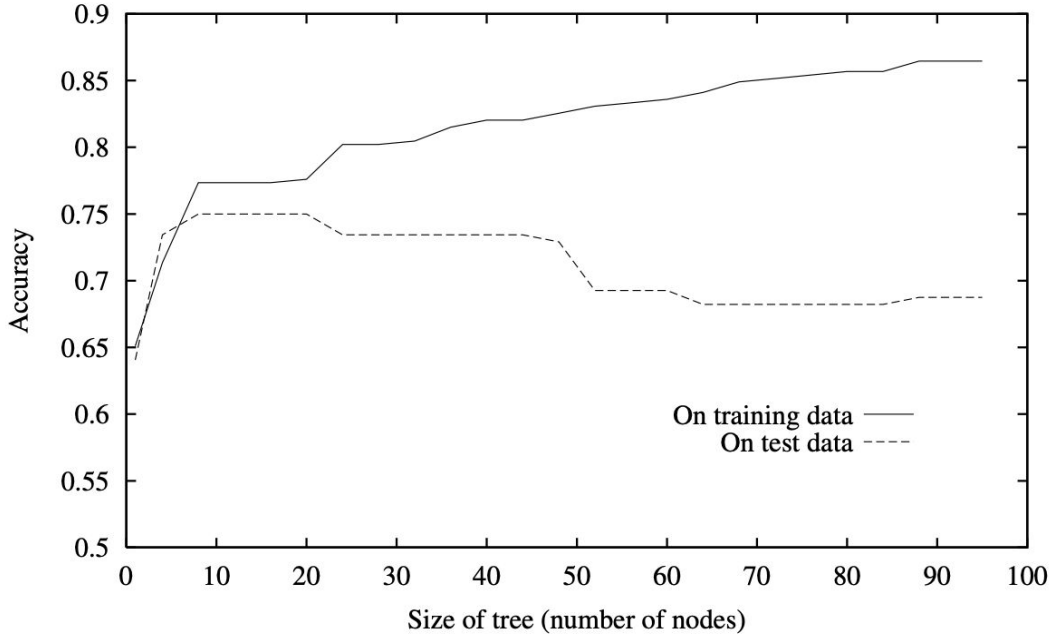
Considere el error de la hipótesis *h* sobre

- datos de entrenamiento: $error_{train}(h)$
- distribución completa D de datos: $error_{D}(h)$

La hipótesis $h \in H$ sobreajusta los datos de entrenamiento si existe una hipótesis alternativa $h' \in H$ tal que

$$error_{train}(h) < error_{train}(h')$$
y
$$error_{D}(h) > error_{D}(h')$$

Sobreajuste en árbol de decisiones



¿Cómo evitar el sobreajuste?

- Dejar de crecer el árbol cuando la división de datos no es estadísticamente significativa.
- Crecer el árbol completo y luego realizar una poda posterior.

¿Cómo seleccionar el "mejor" árbol?:

- Medir el rendimiento sobre los datos de entrenamiento.
- Medir el rendimiento en un conjunto de datos de validación independiente.
- MDL: minimizar

$$size(tree) + size(misclassifications(tree))$$

Podado de error-reducido

Dividir los datos en conjuntos de **entrenamiento** y **validación** (prueba).

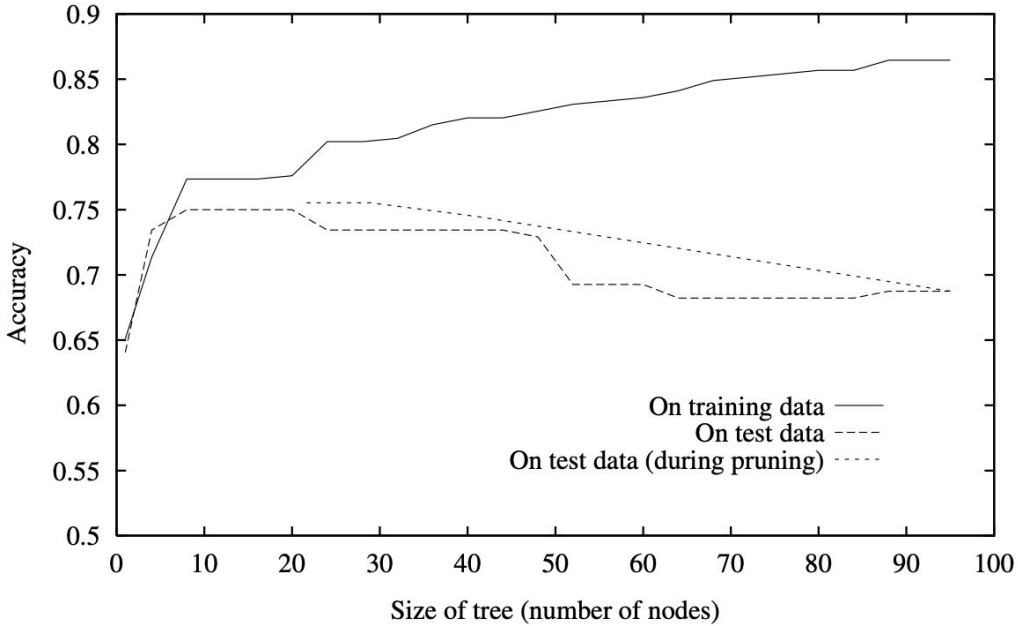
Do until further pruning is harmful:

1. Evaluate the impact on validation set of pruning each possible node (plus those below it).
2. Greedily remove the one that most improves validation set accuracy.

⇒ Produce la versión más pequeña del subárbol más preciso.

¿Qué pasa si los datos son limitados?

Efectos del podado



Reglas post - podado

1. Convertir el árbol en un conjunto equivalente de reglas.
2. Pode cada regla independientemente de las demás.
3. Ordenar las reglas finales en la secuencia deseada para su uso.

Quizá sea el enfoque más utilizado (e.g., C4.5 de Ross Quinlan).

https://www.rulequest.com/Personal/

Convertir árbol en reglas de clasificación

Definición: Las reglas de decisión son reglas con la siguiente forma:

if <conditions> **then** <actions>

```
if outlook = sunny then
|       if humidity <= 75 then yes (2.0)
|       if humidity > 75 then no (3.0)
if outlook = overcast then yes (4.0)
if outlook = rainy then
|       if windy = TRUE then no (2.0)
|       if windy = FALSE then yes (3.0)
```

Atributos continuos

Transformar a atributos discretos que representen el dominio continuo.

- (Humidity > 82) = High, Normal

- Temperature > 76 = Hot
- Temperature > 69 = Mild
- Temperature <70 = Cool

Atributos con muchos valores

Problema:

- Si un atributo tiene muchos valores, *Gain* lo seleccionará.
- Una solución: utilizar *GainRatio* en su lugar:

$$GainRatio(S, A) \equiv \frac{Gain(S,A)}{SplitInformation(S,A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} log_2 \frac{|S_i|}{|S|}$$

- donde S_i es el subconjunto de S para el que A tiene el valor v_i.

Atributos con pesos

- ¿Cómo aprender eficazmente un árbol con pesos?
- Un enfoque: sustituir *Gain* por

$$\frac{Gain^2(S,A)}{Cost(A)}$$

$$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$$

- donde w ∈ [0,1] determina la importancia del peso.

Valores de atributo desconocidos

¿Qué pasa si en algunos ejemplos faltan valores de A?

Use training example anyway, sort through tree:

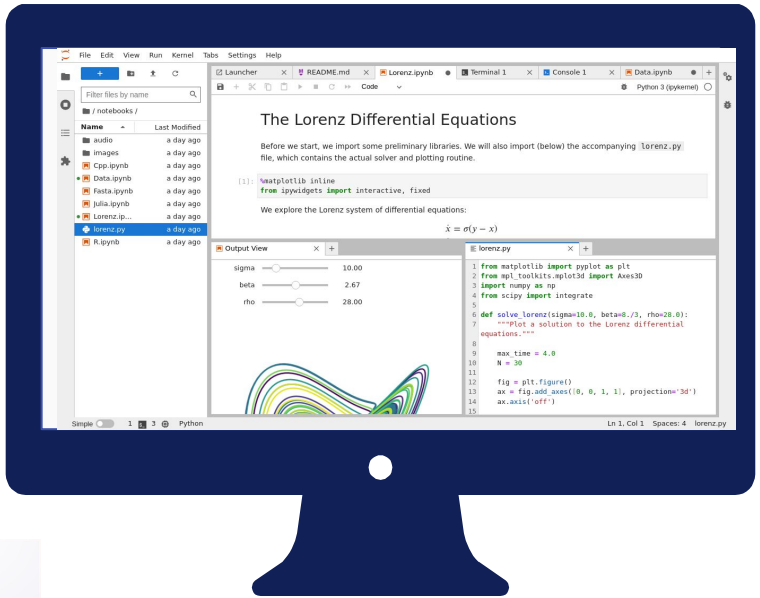
- If node *n* tests *A*, assign most common value of *A* among others sorted to node *n*.
- Assign most common value of *A* among other examples with same target value.
- Assign probability p_i, to each possible value v_i of *A*.
    - Assign fraction p_i of example to each descendant in tree.

Classify new examples in the same fashion.

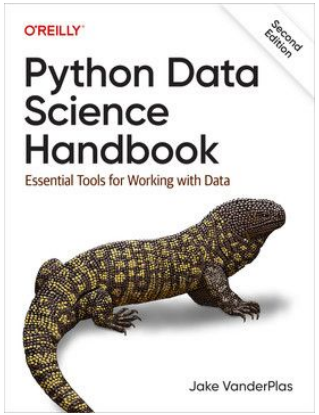
Mejoras C4.5 a ID3

- Maneja tanto atributos **continuos** como discretos.
  - C4.5 crea un umbral y, a continuación, divide la lista en aquellos cuyo valor de atributo es superior al umbral y aquellos que son inferiores o iguales a él.
- Los datos de entrenamiento pueden tener valores de atributo **nulos**.
  - C4.5 permite marcar los valores de los atributos como ? para indicar que faltan. Los valores de atributos que faltan simplemente no se utilizan en los cálculos de ganancia y entropía.
- Gestión de atributos con distintos pesos.
- Poda de árboles tras su creación.
  - C4.5 recorre el árbol una vez creado e intenta eliminar las ramas que no ayudan sustituyéndolas por nodos hoja.

(Go to live notebook)



Extra Libro



GitHub

- 05.08-Random-Forests.ipynb (primera parte)

Gracias!

¿Alguna pregunta?

hussein@cicese.mx

https://sites.google.com/view/husseinlopeznava

SCAN ME

CREDITS: This presentation was based on a template by **Slidesgo**, and includes icons by **Flaticon**.