Arboles de Decisión

Dra Ana Georgina Flesia

Diplomatura en Ciencia de Datos
FaMAF-UNC
Oficina 370
georgina.flesia@unc.edu.ar

2021

Aprendizaje de conceptos

- Usualmente aprendemos, a partir de situaciones y ejemplos, conceptos o categorías del estilo de
  - `pájaro`,
  - `auto`,
  - `temas que tengo que estudiar para aprobar`
- Cada concepto puede ser visto como descriptor de un subconjunto de objetos o eventos de un conjunto mas grande (`el conjunto de animales que son aves`).
- Un concepto es una función booleana sobre el conjunto mas grande que es verdadera sobre el subconjunto y falsa en otra parte.

Aprendizaje de conceptos generalizado

Objetivo

Se busca inferir la definición de un concepto a partir de ejemplos rotulados mediante la creacion de una regla o modelo que representa al concepto y que puede definir la pertenecia al concepto de un nuevo elemento

Aprendizaje de conceptos generalizado

Modelo

la regla a inferir a partir de los ejemplos, que caracteriza a el concepto asignando etiquetas.

Instancia

cada uno de los ejemplos.

Atributo

cada una de las variables medidas a un ejemplo.

Clase

el atributo producido como salida del modelo.

Ejemplo de aprendizaje de concepto por inferencia inductiva:

Concepto

determinar la clase de fruta en la frutera

Atributos

- tamaño
- color
- sabor

Muestra

Frutas previamente inspeccionadas, con sus atributos medidos y clase determinada.

Aprendizaje de conceptos generalizado

Atributos

- Real: puede tomar cualquier valor dentro de un cierto rango.
  - ej. temperatura como un número real (grados).
- Discreto: toma valores discretos ordenados.
  - ej. temperatura como alta, media, baja.
- Categórico: toma valores discretos no ordenados.
  - ej. color como azul, rojo, amarillo

Problemas

- Identificación de las clases en forma unívoca
- Eficiencia en el cálculo
- Manejo del ruido y errores en la muestra

Aprendizaje de conceptos generalizado

Conceptos

- Los conceptos se pueden inferir de diversas formas:
  - Árboles de decisión
  - Combinación de reglas
  - Redes neuronales
  - Modelos bayesianos o probabilísticos

Los árboles de decisión son uno de los modelos más usados en aprendizaje automático.

Arboles de decisión binarios

Arboles binarios

Arboles de decisión se basan en los juegos de 20 preguntas

- La fruta es verde, si o no?, la fruta es amarilla si o no?, la fruta es redonda si o no? , la fruta es larga si o no?
- Las respuestas caracterizan el concepto: el patrón amarillo, no redondo y largo corresponde a una banana.

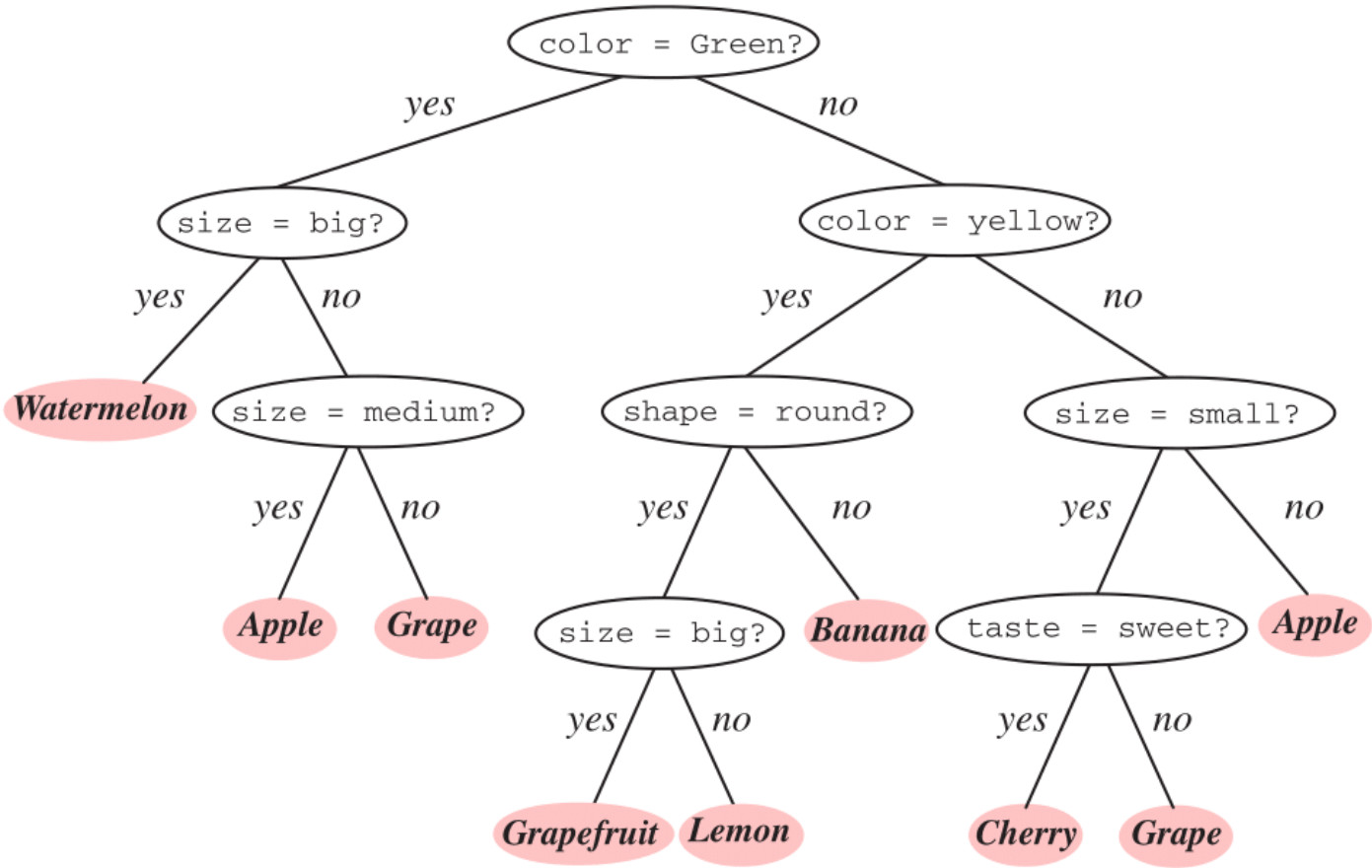
Arboles de decisión

Arbol de decisión

Es un grafo especial con una estructura jerárquica donde cada nodo tiene solo un padre y no hay ciclos en el grafo.

- Cada nodo parte el espacio en zonas basadas en el valor de la variable
- Cada nivel usa una variable diferente o un valor de la variable diferente a las usadas en los niveles anteriores
- Un árbol binario es un árbol con solo dos posibles hijos por cada nodo padre

Árbol Binario para el problema de la frutera

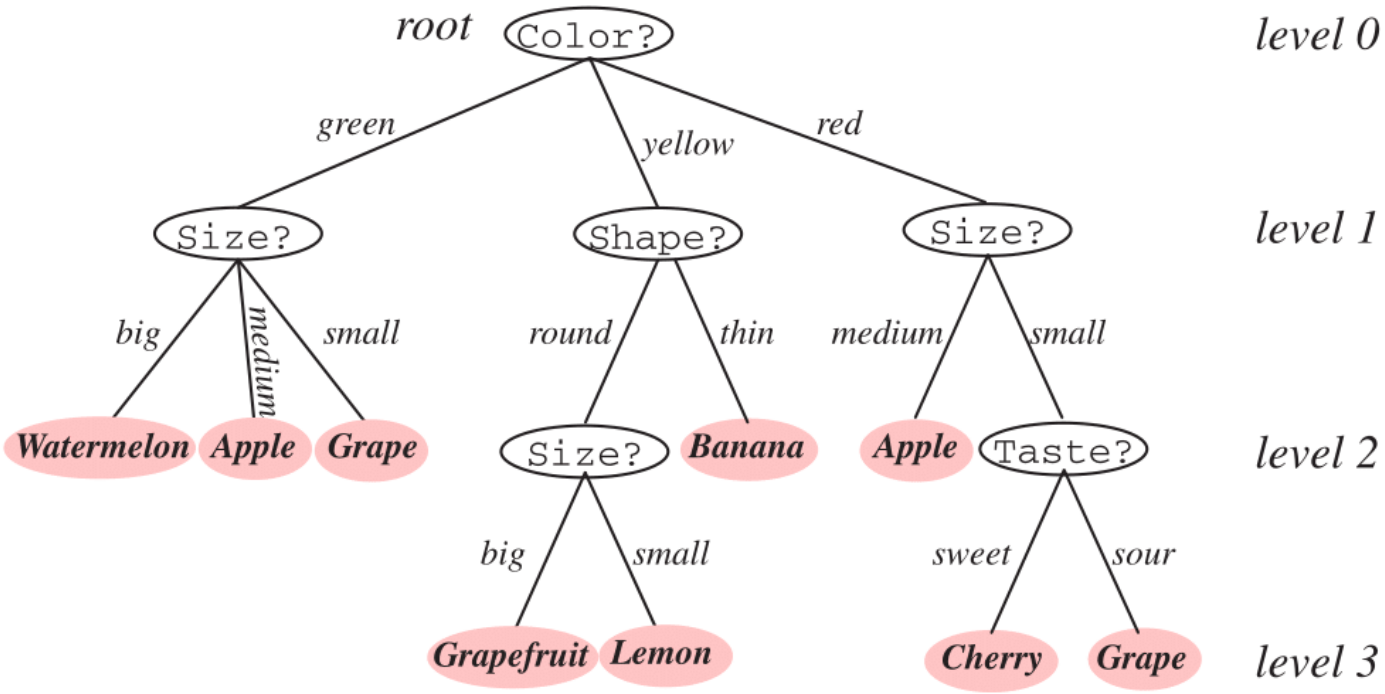


Árbol de Decisión

Generalidades

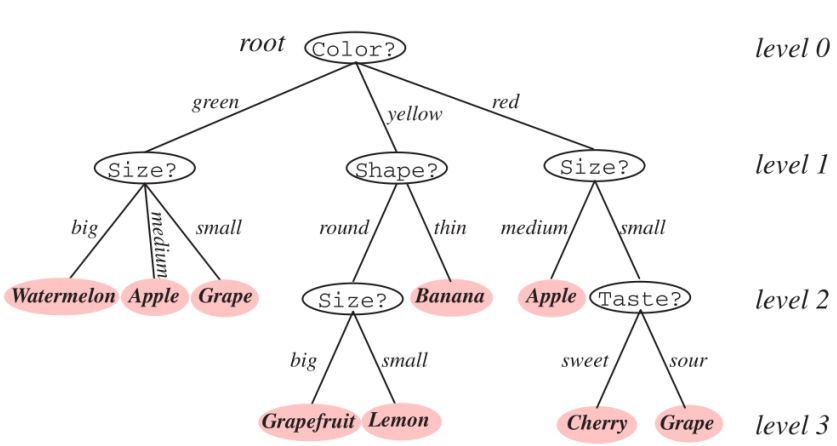
- Cada nodo interno corresponde a una pregunta
- Cada arista corresponde a un resultado de la pregunta
- Cada hoja asigna una clasificación

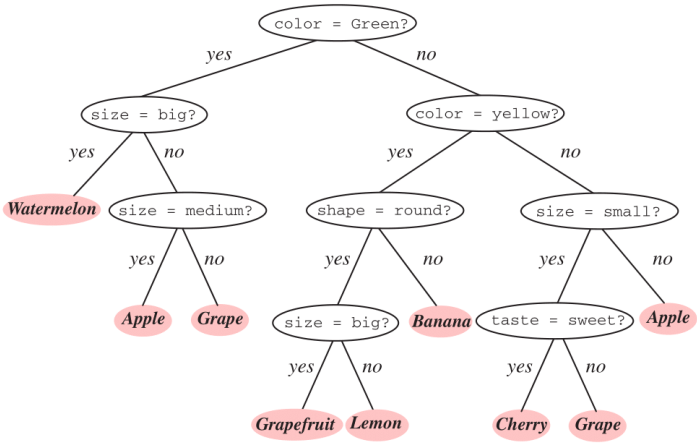
Arbol de Decisión no binario para el problema de la frutera



If (Color=green AND Size=small) OR (Color=red AND Size=small AND Taste=sour) then Grape

Árbol Binario para el problema de la frutera





Múltiples formas de inferir el árbol

Trivial

se crea una ruta del árbol por cada instancia de entrenamiento.

- Árboles excesivamente grandes.
- No funcionan bien con instancias nuevas.

Óptimo

el árbol más pequeño posible compatible con todas las instancias.

- Inviable computacionalmente

Pseudo-óptimo (heurístico)

se selecciona el atributo de los nodos internos de cada nivel del árbol en función de la calidad de la división que produce.

- Los principales programas de generación de árboles utilizan procedimientos similares (C4.5, CART, etc)

Árboles de Decisión

- Para una muestra de entrenamiento, hay muchos árboles que codifican sin error
- Encontrar el árbol mas chico es un problema NPcomplete (Quinlan 1986), por lo cual estamos forzados a usar algún algoritmo local de búsqueda para encontrar soluciones razonables.
- Aprendizaje heurístico es voraz; encuentra el mejor split recursivamente (Breiman et al, 1984; Quinlan, 1986, 1993)
- Si las decisiones son binarias , entonces en el mejor caso, cada decisión elimina la mitad de las regiones (hojas).
- Si hay b regiones, la región correcta puede encontrarse en log2b decisiones, en el mejor caso.

Árboles de Decisión

Preguntas

- ¿Deberíamos restringirnos a preguntas binarias?
- ¿Cuantos atributos deberían ser testeados por nodo?
- ¿Cuando un nodo debe ser considerado hoja?
- ¿Como se puede recortar un árbol muy largo?
- ¿Como se asigna una etiqueta a un nodo hoja?

Árboles de Decisión: ejemplo básico

Un árbol de decisión puede ser construido considerando los atributos de las instancias una por una en forma recursiva

Ciclo central

- Decidir el mejor atributo para el próximo nodo
- Asignar A como el atributo del nodo
- Para cada valor de A, crear un descendiente del nodo
- Elegir las instancias que corresponden al nodo
- Si todas las instancias están perfectamente clasificadas, PARAR y declarar los nodos descendientes como hojas, si no iterar sobre ellos

Árboles de Decisión: ejemplo

Atributos y clase:

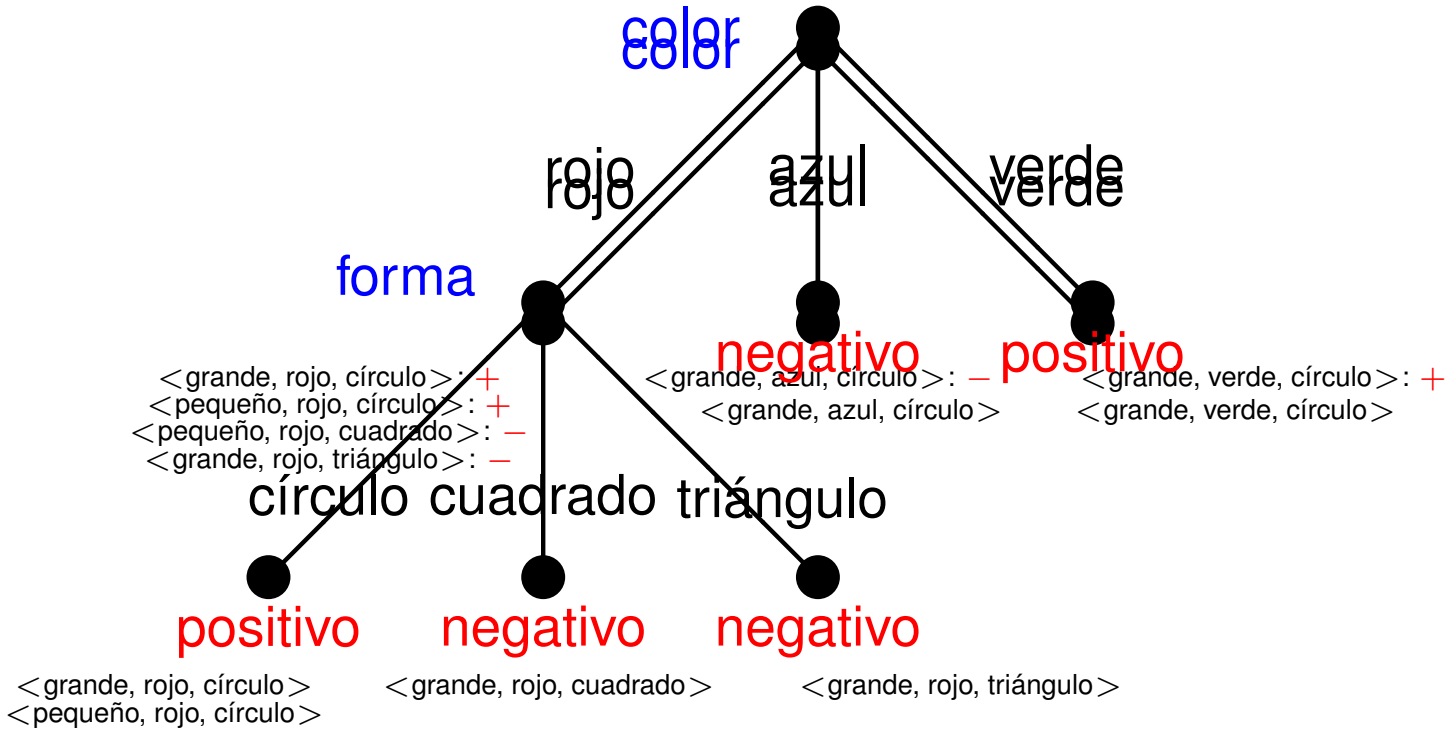
tamaño	color	forma	clase
pequeño	rojo	círculo	$+$
grande	azul	cuadrado	$-$
	verde	triángulo	

Instancias:

<grande, rojo, círculo>: $+$
<pequeño, rojo, círculo>: $+$
<pequeño, rojo, cuadrado>: $-$
<grande, azul, círculo>: $-$
<grande, verde, círculo>: $+$
<grande, rojo, triángulo>: $-$

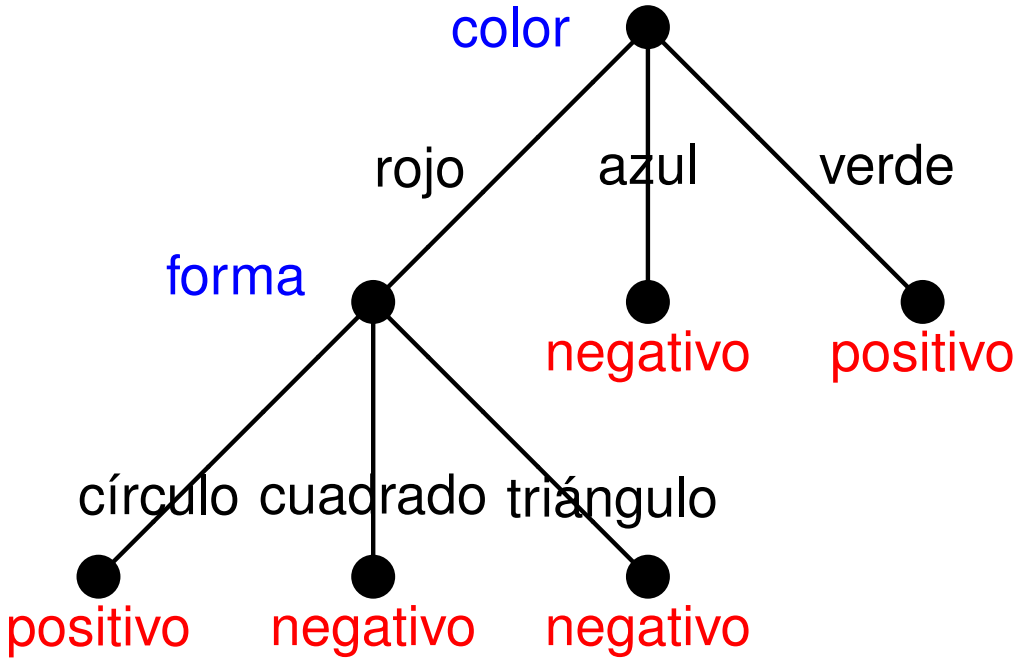
Construcción de un árbol a partir de instancias

Los nodos evalúan atributos, con una arista para cada posible valor del atributo, y se continúa hasta que las hojas especifican la categoría:



Construcción de un árbol a partir de instancias

El árbol construído puede codificarse como una cascada
`if, then, else`

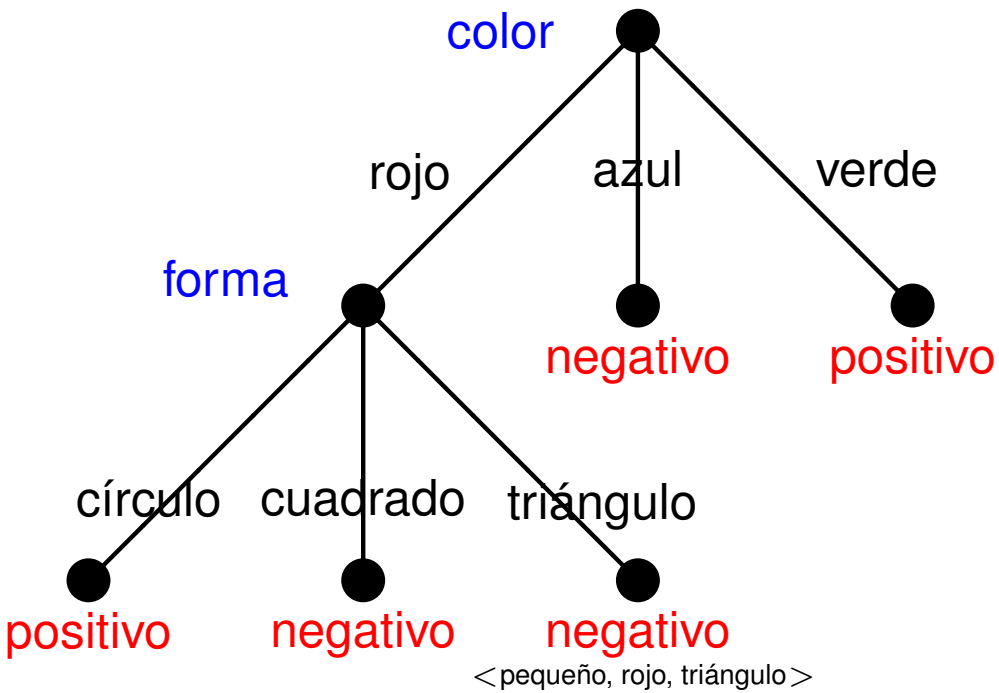


Árboles de Decisión

Ante una nueva instancia no etiquetada:

<pequeño, rojo, triángulo>

el árbol construído funciona como clasificador:



Particularidades de los Árboles de Decisión

- Existen métodos para tratar datos faltantes
- Los árboles de clasificación tienen etiquetas de clase discretas en las hojas, mientras que los árboles de regresión tienen valores contínuos
- Los algoritmos para construir árboles son eficientes para el procesamiento de grandes cantidades de datos
- Existen métodos para tratar datos de entrenamiento ruidosos, con errores tanto en en las características como en la clase
- Los árboles pueden representar cualquier concepto

Particularidades de los Árboles de Decisión

Variables continuas

- Las características con valores contínuos se pueden partir en dos o más rangos, mediante un umbral (p.e. longitud < 3 y longitud ≥ 3)
- Podemos dividir cualquier variable , con cualquier umbral.
- Sin embargo, para cada variable, los únicos umbrales que consideramos son los n valores de la variable en la muestra de entrenamiento
- Si ordenamos cada variable con esos n valores, se puede calcular fácilmente cualquier métrica elegida para evaluar la división

Elección del atributo de un nodo

- El objetivo es obtener el árbol más chico posible, pues es el modelo mas parsimonioso
- El método recién empleado (top-down) hace una búsqueda voraz (greedy), por lo cual no garantiza encontrar el árbol más chico posible, si bien en general encuentra una buena solución
- La raíz es el atributo que mejor clasifica a los datos
- Se elige la característica que crea subconjuntos de ejemplos relativamente “puros” en una sola clase, de forma que las hojas queden más cerca de la raíz
- Hay muchas heurísticas para elegir una característica. La más popular se basa en Ganancia de Información (Information Gain) propuesta por Quinlan (1979)

Entropía de Shannon

Hay ganancia de información cuando la división envía instancias con clases distintas a distintos nodos. Se mide como reducción de la entropía

- La entropía de un conjunto de instancias S, relativo a una clasificación binaria (0 y 1) es

$$\text{Entropy}(S) = -p_0 \log_2(p_0) - p_1 log_2(p_1)$$

  donde p_1 es la fracción de instancias positivos en S y $p_0 = 1 - p_1$ es la fracción de negativos
- Si todos las instancias están en una categoría, la entropía es 0
- Si las instancias están mezclados en partes iguales ($p_1 = p_0 = 0.5$), la entropía alcanza su máximo en 1
- La entropía representa el número medio de bits que se necesitan para codificar la clase en S

Ganancia de Información

Para problemas multi-clase con c categorías, la entropía se generaliza según

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

La ganancia de información de un conjunto de instancias respecto de un atributo F es la información mutua que resulta al dividir según este atributo

$$\text{Gain}(S, F) = \text{Entropy}(S) - \sum_{v=\text{values(F)}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

donde S_v es el subconjunto de S que tiene valor v para el atributo F
La entropía de cada subconjunto resultante está ponderado por su tamaño (cantidad de elementos que contiene)

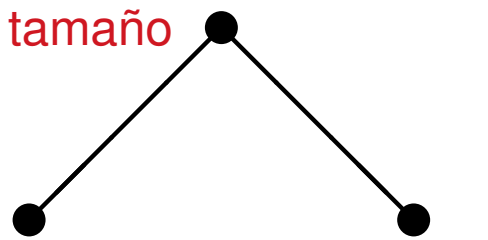
Ejemplo de Ganancia de Información:

S:

e_1 =<grande, rojo, círculo>: + e_2 =<pequeño, rojo, círculo>: +

e_3 =<pequeño, rojo, cuadrado>: − e_4 =<grande, azul, círculo>: −

$$\text{Entropy}(S) = -p_0 \log_2(p_0) - p_1 log_2(p_1) = -2(\frac{1}{2}\log_2(\frac{1}{2})) = 1$$



e_1+, e_4- e_2+, e_3-

$E_G = 1$ $E_P = 1$

$$G = 1 - (0.5 \times 1 + 0.5 \times 1) = 0$$

$$\text{Entropy}(S_G) = -2(\frac{1}{2}\log_2(\frac{1}{2})) = 1$$

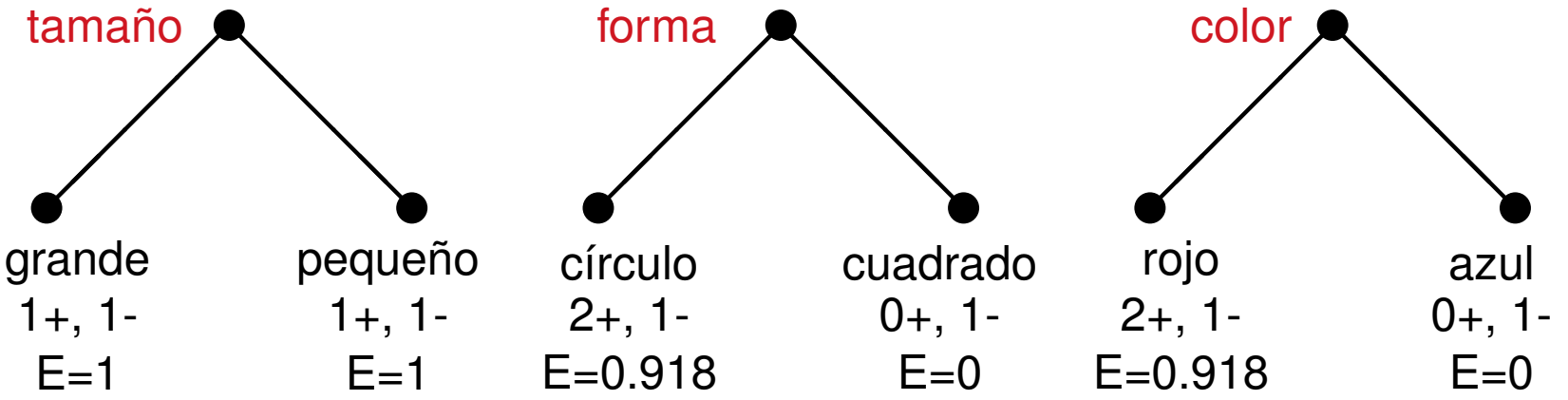
$$\text{Entropy}(S_P) = -2(\frac{1}{2}\log_2(\frac{1}{2})) = 1$$

$$\text{Gain} = \text{Entropy}(S) - \frac{|S_G|}{|S|}\text{Entropy}(S_G) - \frac{|S_P|}{|S|}\text{Entropy}(S_P)$$

$$= 1 - \frac{2}{4}1 - \frac{2}{4}1 = 0$$

Ejemplo de Ganancia de Información:

S: <grande, rojo, círculo>: $+$ <pequeño, rojo, círculo>: $+$
<pequeño, rojo, cuadrado>: $-$ <grande, azul, círculo>: $-$



$G = 1 - (0.5 \times 1 + 0.5 \times 1) = 0$

$G = 1 - (0.75 \times 0.918 + 0.25 \times 0) = 0.311$

$G = 1 - (0.75 \times 0.918 + 0.25 \times 0) = 0.311$

Uso de las características

- Los atributos (no necesariamente todas) aparecen sólo una vez en los nodos (no se repiten)
- Un atributo con valores contínuos, puede aparecer en más de un nodo pero con diferentes valores de corte
- Ejemplo: Deporte al aire libre

Temperatura (^{o}C)	5	12	18	22	25	33
Práctica	no	no	sí	sí	sí	no

Búsqueda en el Espacio de Hipótesis

- Se trata de aprendizaje en batch, ya que las instancias de entrenamiento se procesan todas juntos, en contraste con un aprendizaje incremental que actualizaría la hipótesis después de cada instancia
- Aplica búsqueda voraz que puede quedar limitada a una solución óptima local
- Se encuentra un árbol consistente con un conjunto de entrenamiento sin conflictos (de clase), pero no necesariamente el más simple
- La Ganancia de Información tiene sesgo hacia los árboles poco profundos

Complejidad computacional

- Supongamos n ejemplos y m características
- En el peor caso se tiene un árbol donde para alcanzar las hojas se tienen que evaluar todas los atributos
- Al nivel i se evaluan las $(m - i)$ características restantes y para calcular la ganancia de información se usan todas las instancias

$$\sum_{i=1}^{m} (m - i)\, n \sim O(nm^2)$$

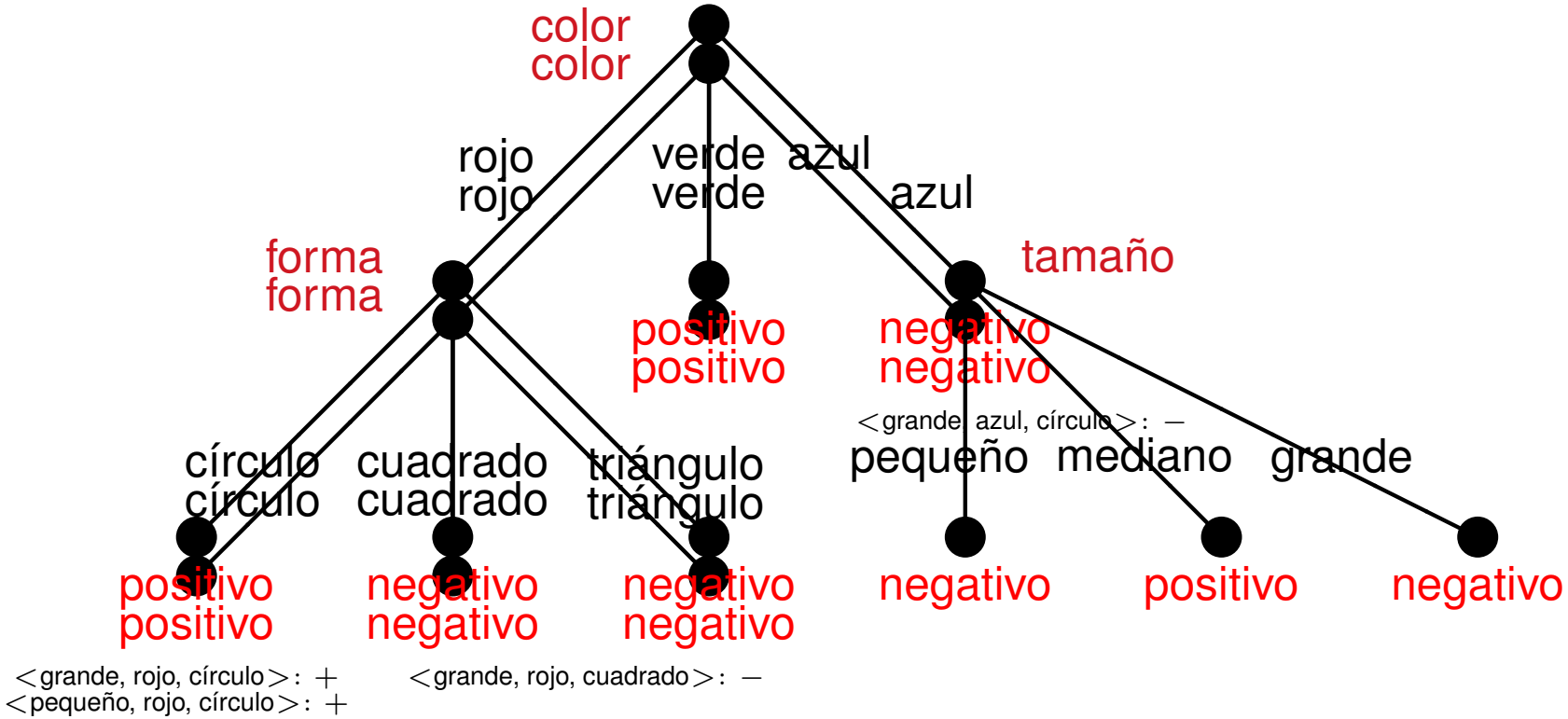
- En la práctica rara vez el árbol será completo y la complejidad usualmente resulta lineal en m y en n

Problema de sobreajuste (overfitting)

- Ocurre al aprender a clasificar perfectamente las instancias de entrenamiento pero a costa de fallar en la tarea de generalizar
- Potencia los problemas:
  - Ruido en las instancias de entrenamiento
  - El algoritmo puede tomar decisiones basadas en datos que no reflejen la distribución de una mayor cantidad de instancias
- Decimos que un clasificador sobreajusta si su performance es muy buena en el set de entrenamiento, pero en comparación desmejora mucho sobre un set de evaluación independiente

Sobreajuste de ruido

- El ruido en una instancia de un atributo puede causar sobreajuste. Por ejemplo añadir la instancia ruidosa:
  $<$mediano, azul, círculo$>$: $+$ (que debe ser $-$!)



Prevención del sobreajuste: Pruning (Poda)

- Dos métodos básicos:
  - Prepruning: Detener en algún momento el crecimiento del árbol durante la construcción top-down cuando ya no hay suficientes datos para hacer decisiones criteriosas
    (por ejemplo tener un mínimo número de ejemplo por hoja)
  - Postpruning: Luego de obtener el árbol completo, eliminar subárboles que no contienen suficiente evidencia.
- Luego de la poda, rotular la hoja resultante con la clase mayoritaria

Métodos para determinar qué ramas podar

- Validation: Reservar algunas instancias de entrenamiento como conjunto de validación (validation set, tuning set) para evaluar si el error de clasificación postpruning no es peor que el anterior:
  reduced error-pruning method
- Evaluación estadística: usando las instancias de entrenamiento, implementar un test χ^2 para determinar si hay o no mejora de performance al retener una arista
- Mínima longitud de descripción (MDL): Determinar si la complejidad adicional de la hipótesis es menos compleja que simplemente recordar explícitamente todas las excepciones que resultan de la poda

Problemas usuales con la poda

- La evaluación estadística con los mismos datos de entrenamiento es poco confiable
- El problema de la validación es que potencialmente “gasta” instancias de entrenamiento en el conjunto de validación
- La severidad de este problema depende de dónde nos encontramos en la curva de aprendizaje:

Validación cruzada

- Uso de una métrica MDL
- Se realizan pruebas de reduced error-pruning usando diferentes particiones aleatorias de los datos para obtener los conjuntos de aprendizaje y validación (usualmente 10-fold cross-validation)
- Registrar la complejidad del árbol podado en cada fold de aprendizaje. Sea C el promedio de las complejidades medidas
- Construir un árbol final a partir de todos los datos de entrenamiento y detener la construcción al alcanzar la complejidad C
- No hay pérdida de datos de entrenamiento

Saga de algoritmos

- Algoritmo ID3 (Quinlan 1986) utiliza Ganancia de Información
- Algoritmo C4.5 (Quinlan 1993)
  - incorpora pruning
  - maneja atributos con diferentes costos
  - maneja características con datos faltantes
  - maneja características con datos discretos y contínuos
  - J48 una implementación java open source del C4.5 en WEKA
  - El costo numérico de computar logaritmos se puede solucionar usando la impureza de Gini

$$\text{Gini}(S) = \sum_{i=1}^{c} p_i \left(1 - p_i\right)$$

Saga de algoritmos

- Costes en los errores de clasificación $\rightarrow$ C5.0
- Clase con valores contínuos (árboles de regresión) $\rightarrow$ CART
- Aprendizaje Incremental $\rightarrow$ ID4 ID5 ID5R ID6MDL