Árboles de Decisión

Tomás Arredondo Vidal

26/3/08

Árboles de Decisión

Contenidos

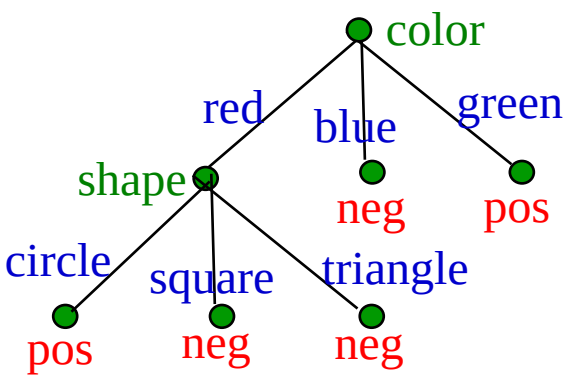
- Árboles de Decisión
- Sobreajuste
- Recorte (Pruning)

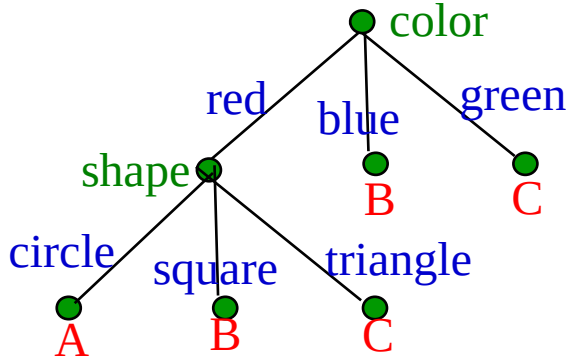
Investigación Relacionada a los Árboles de Decisión

- William of Occam inventa el criterio de Occam's razor - 1320
- Hunt usa el método (CLS) para modelar aprendizaje humano de conceptos - 1960's.
- Quinlan desarrolla ID3 con la heurística de ganancia de información para desarrollar sistemas expertos desde ejemplos – 1970s
- Breiman, Friedman y otros desarrollan CART (Classification and Regression Trees) similar a ID3 – 1970s
- Una variedad de mejorar se introducen para acomodar ruido, características continuas, características incompletas, mejoras en métodos de división – 1980s
- Quinlan's actualiza sistemas de decision (C4.5) -1993.
- Weka se incluye una versión en Java de C4.5 llamado J48.

Árboles de Decisión

- Clasificadores basados en árboles para instancias (datos) representados como vectores de características (features).
- Nodos prueban características, hay una rama para cada valor de la característica, las hojas especifican la categoría.





- Pueden representar cualquier conjunción (AND) y disyunción (OR)
- Pueden representar cualquier función de clasificación de vectores de características discretas.
- Pueden ser rescritas como reglas, i.e. disjunctive normal form (DNF).
  - red $\wedge$ circle $\rightarrow$ pos
  - red $\wedge$ circle $\rightarrow$ A

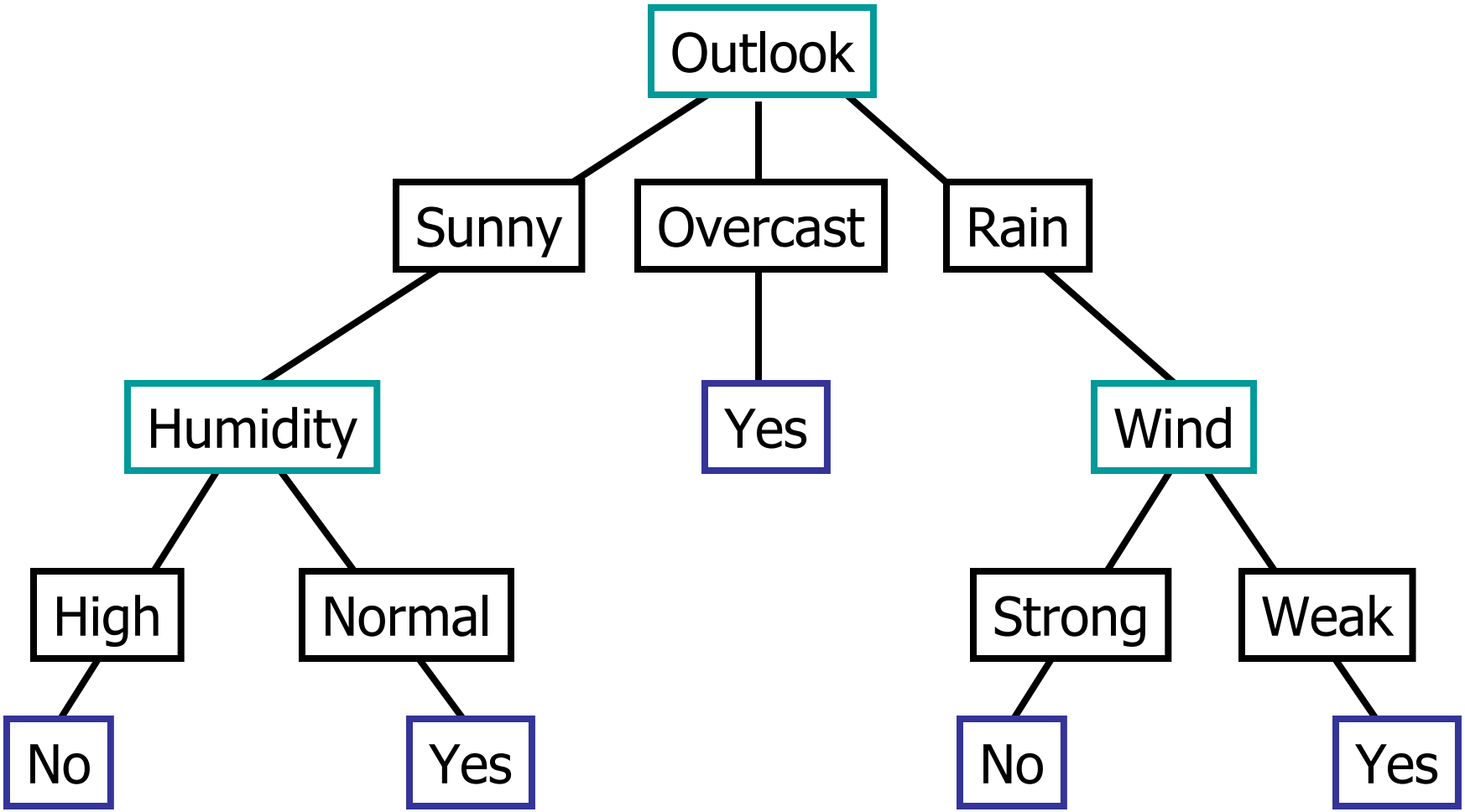
    blue $\rightarrow$ B; red $\wedge$ square $\rightarrow$ B

    green $\rightarrow$ C; red $\wedge$ triangle $\rightarrow$ C

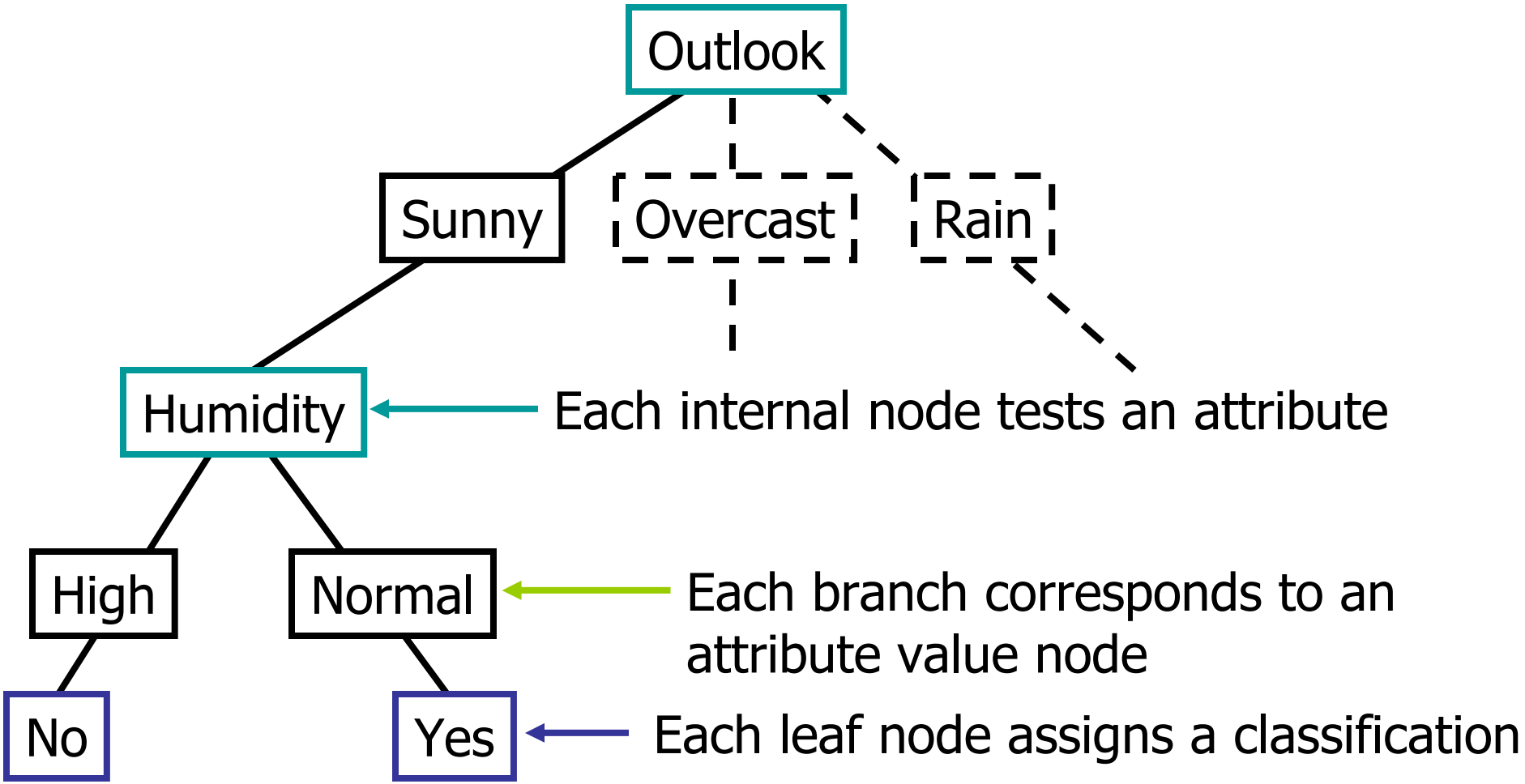
Propiedades de Árboles de Decisión

- Características (features) continuas (reales) pueden ser clasificadas al permitir nodos que dividan una característica real en dos rangos basados en umbrales (e.g. largo < 3 and largo ≥3)
- Árboles de clasificación tienen valores discretos en las ramas, árboles de regresión permiten outputs reales en las hojas
- Algoritmos para encontrar árboles consistentes son eficientes para procesar muchos datos de entrenamiento para tareas de datamining
- Pueden manejar ruido en datos de entrenamiento
- Ejemplos:
  - Diagnostico medico
  - Análisis de riesgo en crédito
  - Clasificador de objetos para manipulador de robot (Tan 1993)

Árbol de Decisión para PlayTennis

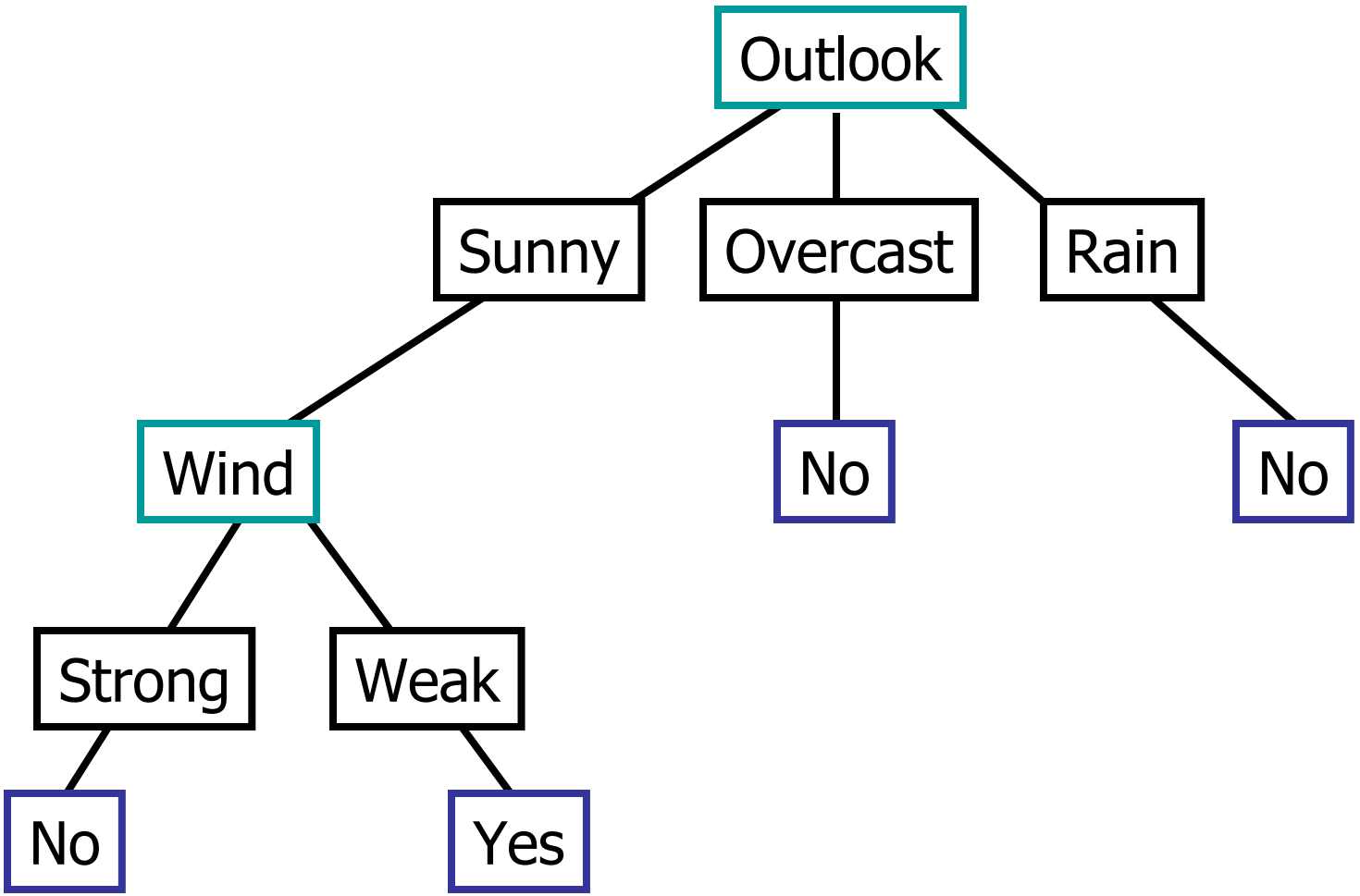


Árbol de Decisión para PlayTennis

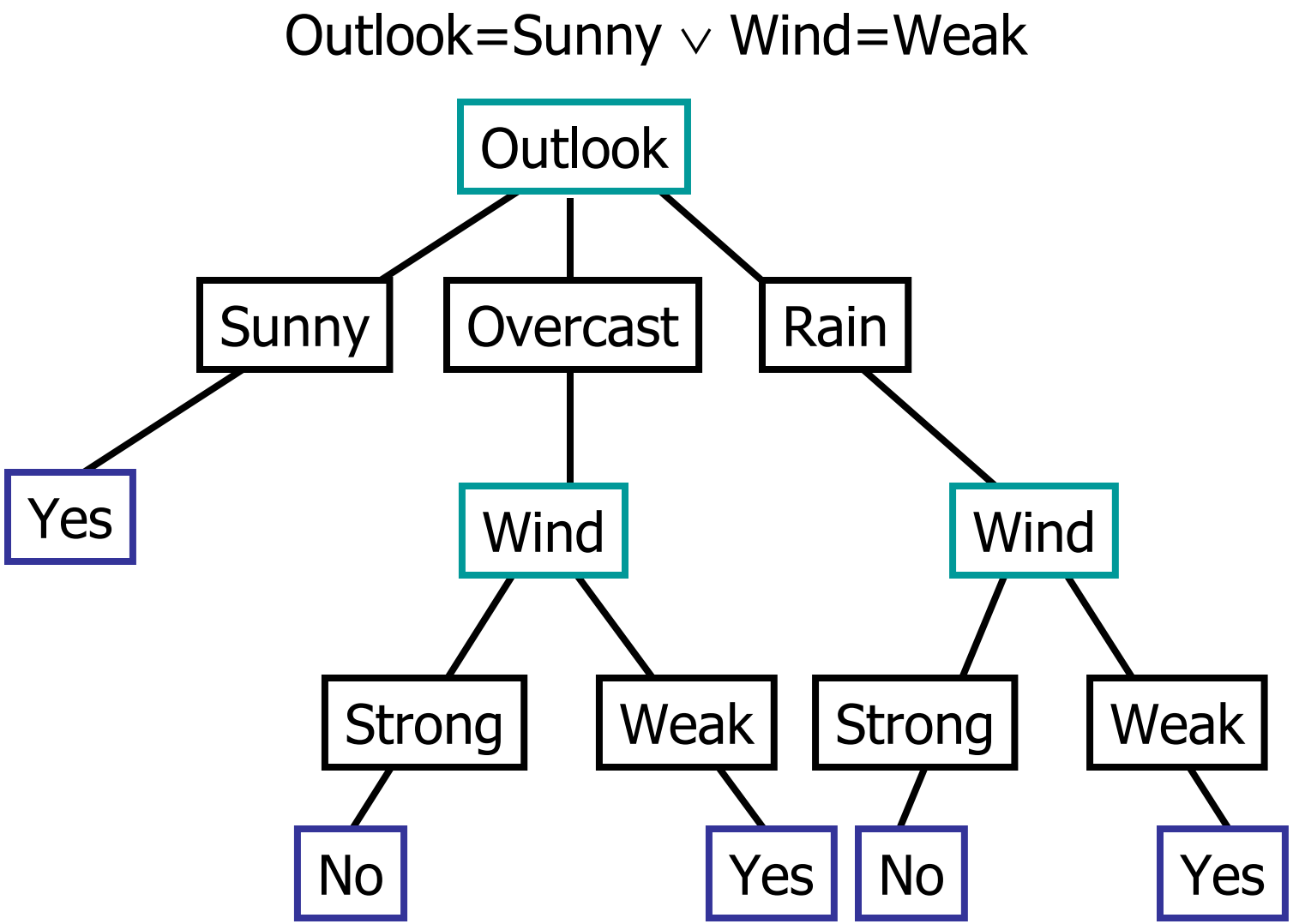


Árbol de Decisión para Conjunción

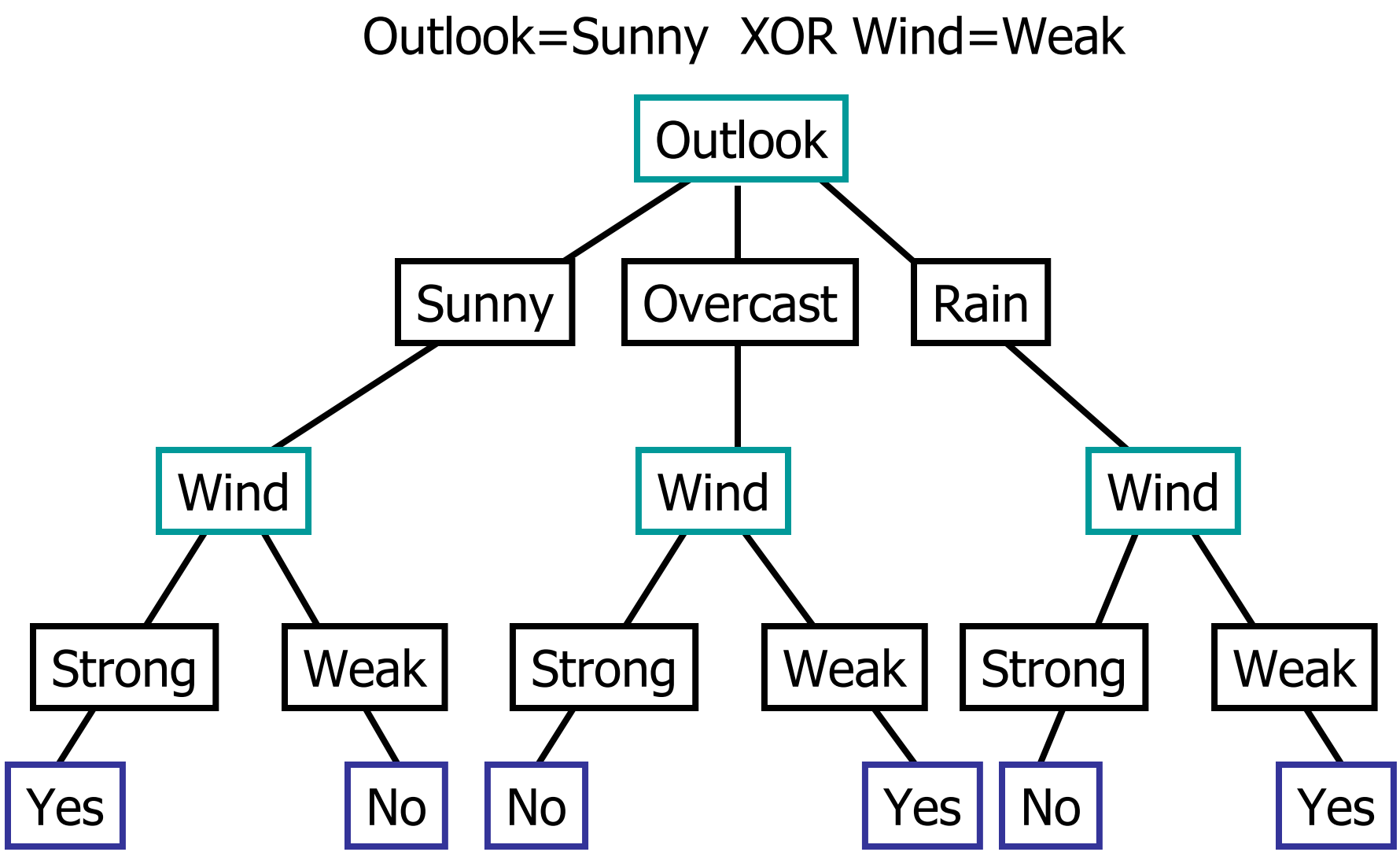
Outlook=Sunny $\wedge$ Wind=Weak



Árbol de Decisión para Disyunción

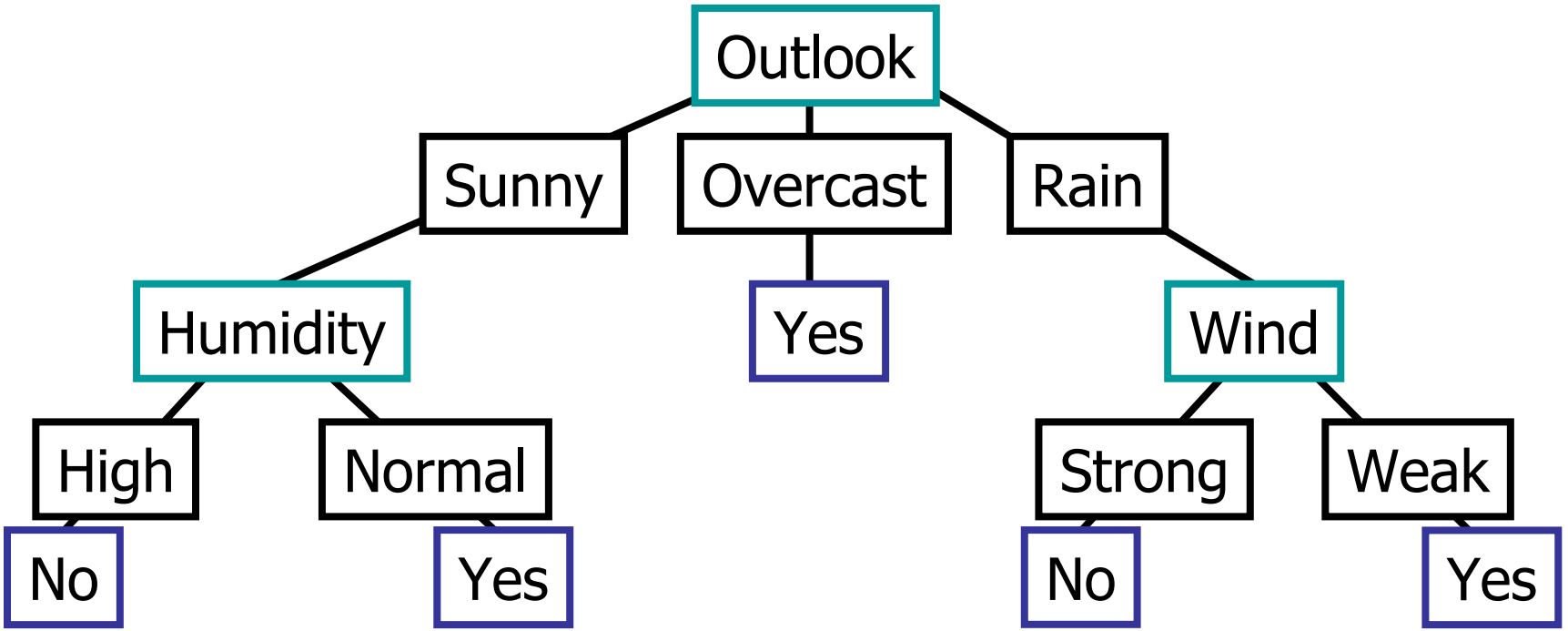


Árbol de Decisión para XOR



Árbol de Decisión

- Árboles de decisión representan disyunciones de conjunciones

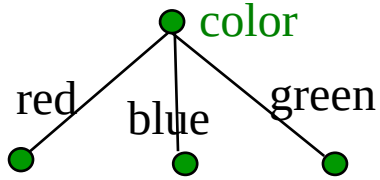


(Outlook=Sunny $\wedge$ Humidity=Normal)
$\vee$ (Outlook=Overcast)
$\vee$ (Outlook=Rain $\wedge$ Wind=Weak)

Método Top-Down de Construcción

- Construir un árbol Top-Down usando dividir para conquistar.

<big, red, circle>: +
<small, red, square>: –
<small, red, circle>: +
<big, blue, circle>: –
<small, red, triangle>: +
<small, green, triangle>: -

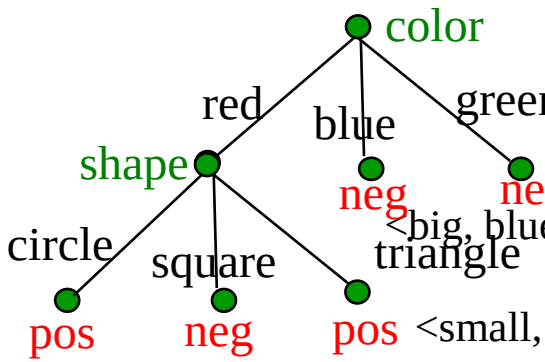


<big, red, circle>: +
<small, red, circle>: +
<small, red, square>: –
<small, red, triangle>: +

Método Top-Down de Construcción

- Construir un árbol Top-Down usando dividir para conquistar.

<big, red, circle>: +
<small, red, square>: –
<small, red, circle>: +
<big, blue, circle>: –
<small, red, triangle>: +
<small, green, triangle>: -



<big, red, circle>: +
<small, red, circle>: +
<small, red, square>: –
<small, red, triangle>: +

<small, green, triangle>: +
<big, blue, circle>: –

<big, red, circle>: +
<small, red, circle>: +

<small, red, square>: –

<small, red, triangle>: +

Pseudocódigo para Construcción de Árbol

DTree(*examples*, *features*) returns a tree
  If all *examples* are in one category, return a leaf node with that category label.
  Else if the set of *features* is empty, return a leaf node with the category label that
      is the most common in examples.
  Else pick a feature *F* and create a node *R* for it
    For each possible value v_i of *F*:
      Let $examples_i$ be the subset of examples that have value v_i for *F*
      Add an out-going edge *E* to node *R* labeled with the value v_i.
      If $examples_i$ is empty
        then attach a leaf node to edge *E* labeled with the category that
          is the most common in *examples*.
        else call DTree($examples_i$, *features* – {*F*}) and attach the resulting
          tree as the subtree under edge *E*.
    Return the subtree rooted at *R*.

Seleccionar una Buena Característica para Dividir

- Objetivo es tener un árbol que sea lo mas simple posible (de acuerdo a Occam's Razor)
- Encontrar un árbol mínimo de decisión (nodos, hojas o profundidad) es un problema de optimización NP-hard.
- El método top-down que divide para conquistar hace una búsqueda codiciosa (greedy) para obtener un árbol simple pero no garantiza que sea el mas simple.
- Se quiere seleccionar una característica que genera subconjuntos de ejemplos que son similares en una clase para que estén cerca de ser nodos hojas.
- Hay una variedad de heurísticas para elegir un buen test o condición, un método popular esta basado en el incremento de la información (information gain) que se origino con el sistema ID3 (Quinlan 1979).

Occam's Razor

- Occam's razor: Se debe preferir la hipótesis más corta que tenga coincidencia (o describa) con los datos

Porque preferir hipótesis cortas?

Argumentos a favor:

- Hay menos hipótesis cortas que largas
- Es poco probable que una hipótesis corta que describa los datos sea una coincidencia
- Una hipótesis larga que describa los datos puede ser una coincidencia (sobreajuste o overfitting)

Argumentos en contra:

- Hay muchas maneras de definir hipótesis cortas y una definición depende de la representación interna del que aprende (i.e. una representación interna corta para uno puede ser larga para otro)
- Porque el tamaño de la hipótesis es importante?

Entropía

- Entropía (incertidumbre, desorden, impureza) de un conjunto de ejemplos S, relativo a una clasificación binaria es:

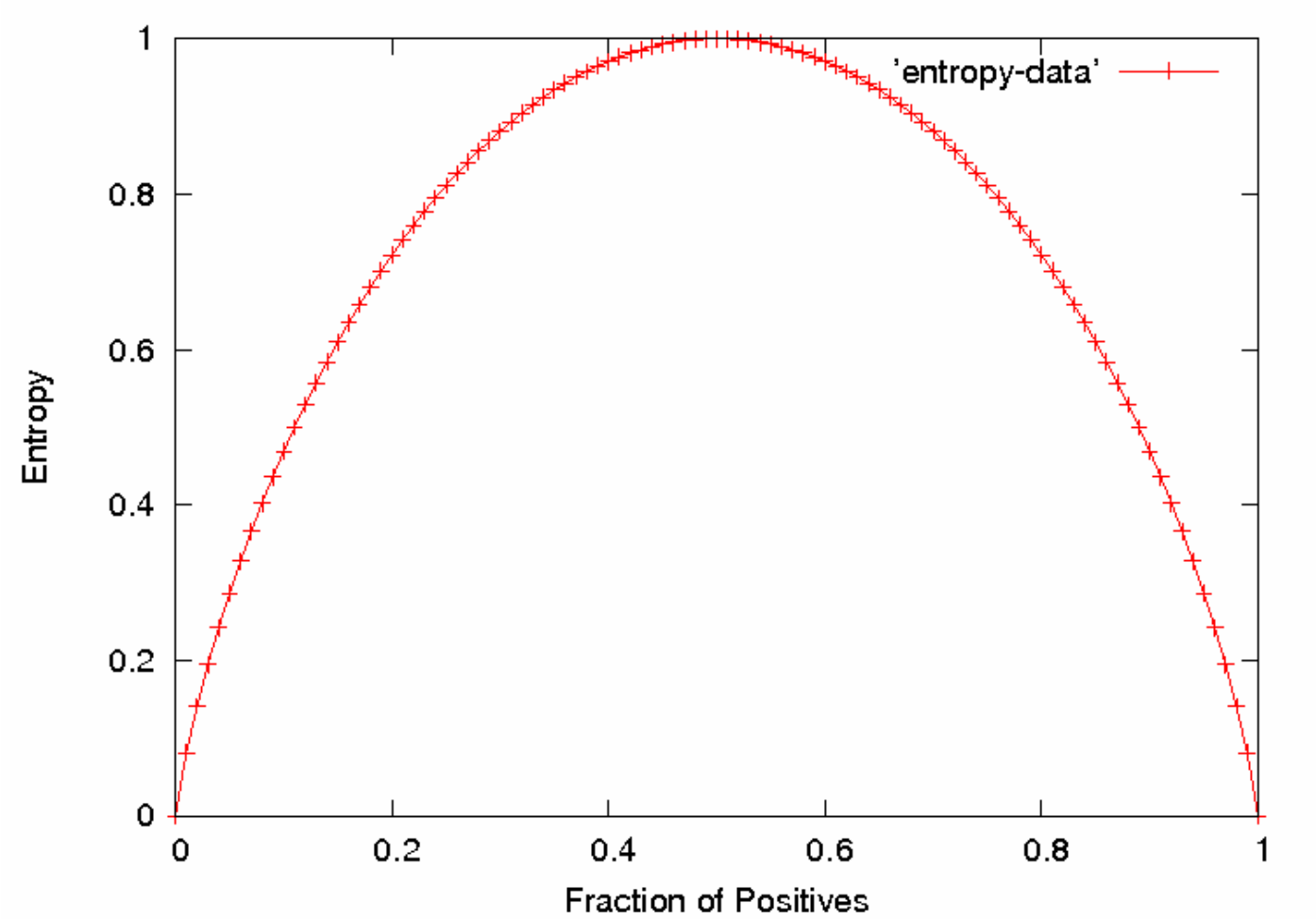
$$Entropy\ (S) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$$

en el cual p_1 es la fracción de ejemplos positivos en S y p_0 es la fracción de negativos.

- Si todos los ejemplos están en una categoría, entropía es zero (definimos $0 \cdot \log(0)=0$)
- Si los ejemplos están igualmente mezclados ($p_1=p_0=0.5$), entropía es una máxima de 1.
- Entropía se puede ver como el máximo numero de bits requeridos en promedio para codificar la clase de un ejemplo en S.
- Para problemas de múltiples clases (multi-class) con c categorías, la entropía generaliza a:

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

Grafico de Entropía para Clasificación Binaria



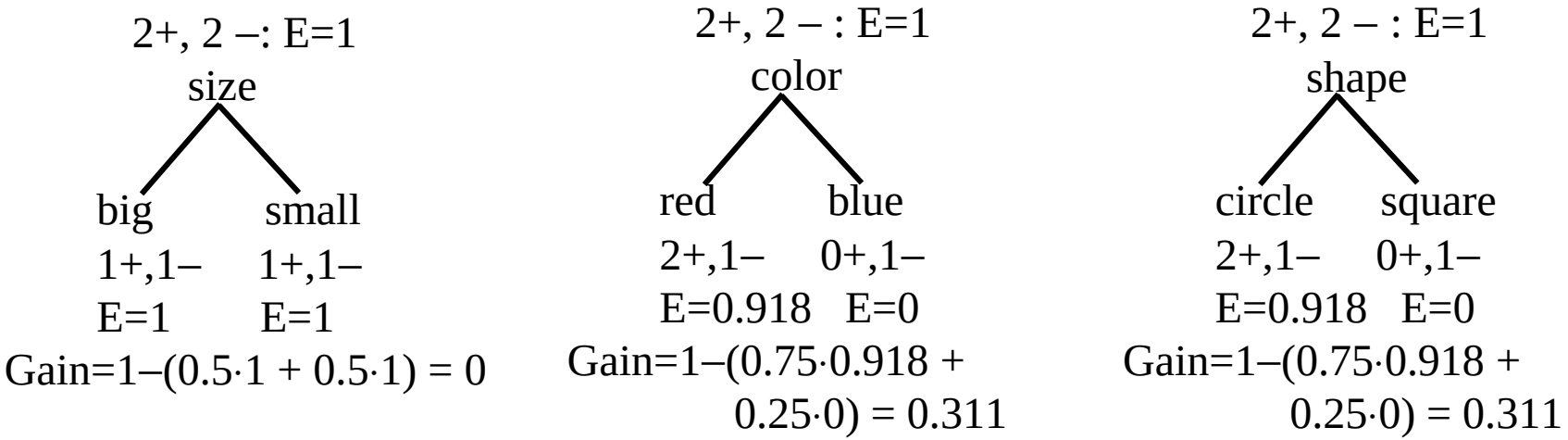
Incremento de la Información

- El incremento de información Gain(S, F) ocurre por reduccion en la entropia esperada al ordenar S basado en atributo F

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

en el cual S_v es el subconjunto de S que tiene el valor v para característica F.

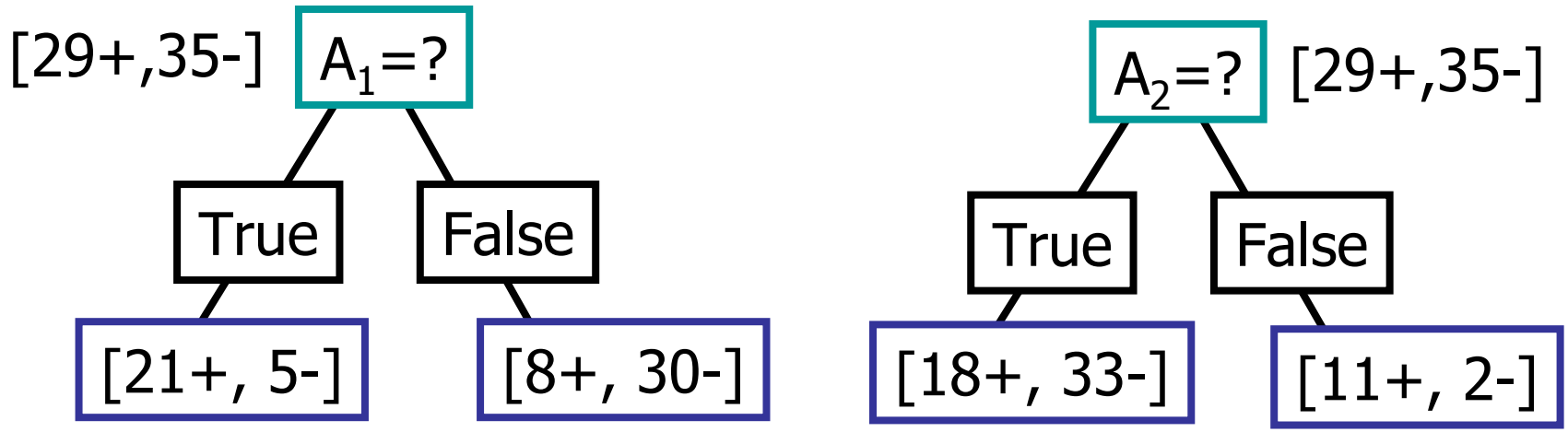
- Entropía de cada subconjunto resultante se pondera por su tamaño relativo.
- Ejemplo:
  - <big, red, circle>: + <small, red, circle>: +
  - <small, red, square>: – <big, blue, circle>: –



Ejemplo: Incremento de la Información

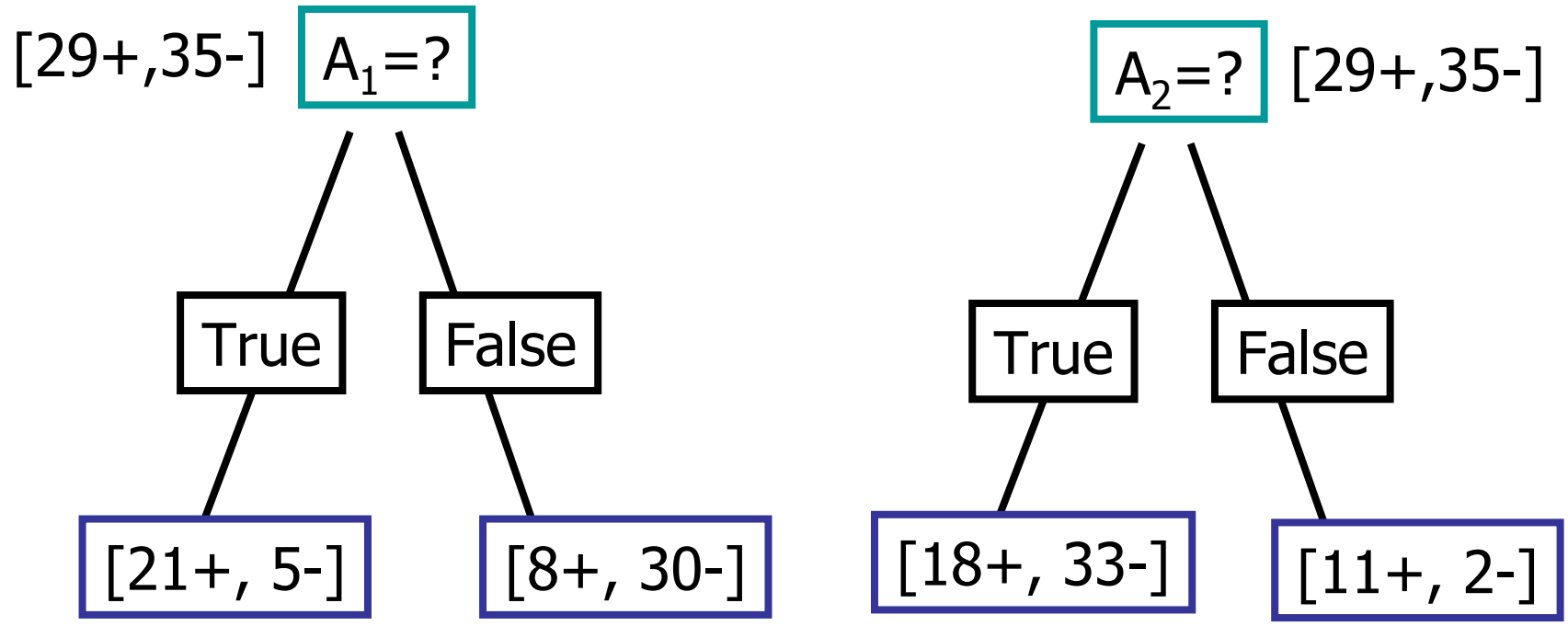
$$Gain(S,A)=Entropy(S) - \sum_{v \in values(A)} |S_v|/|S| \; Entropy(S_v)$$

$$Entropy([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$$



Ejemplo: Incremento de la Información

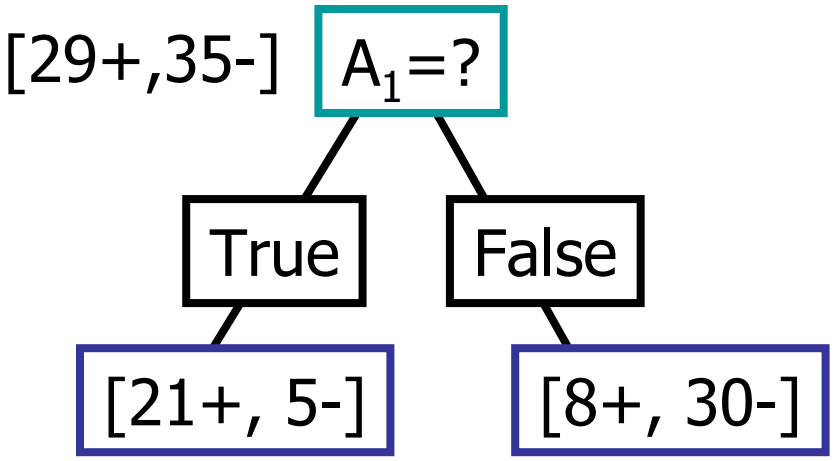
Cual atributo es mejor?

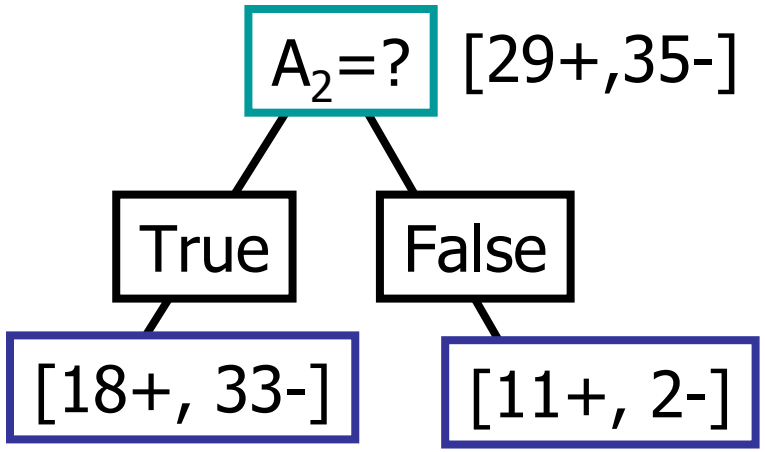


Ejemplo: Incremento de la Información

Entropy([21+,5-]) = 0.71
Entropy([8+,30-]) = 0.74
Gain(S,A_1)=Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
=0.27

Entropy([18+,33-]) = 0.94
Entropy([11+,2-]) = 0.62
Gain(S,A_2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12

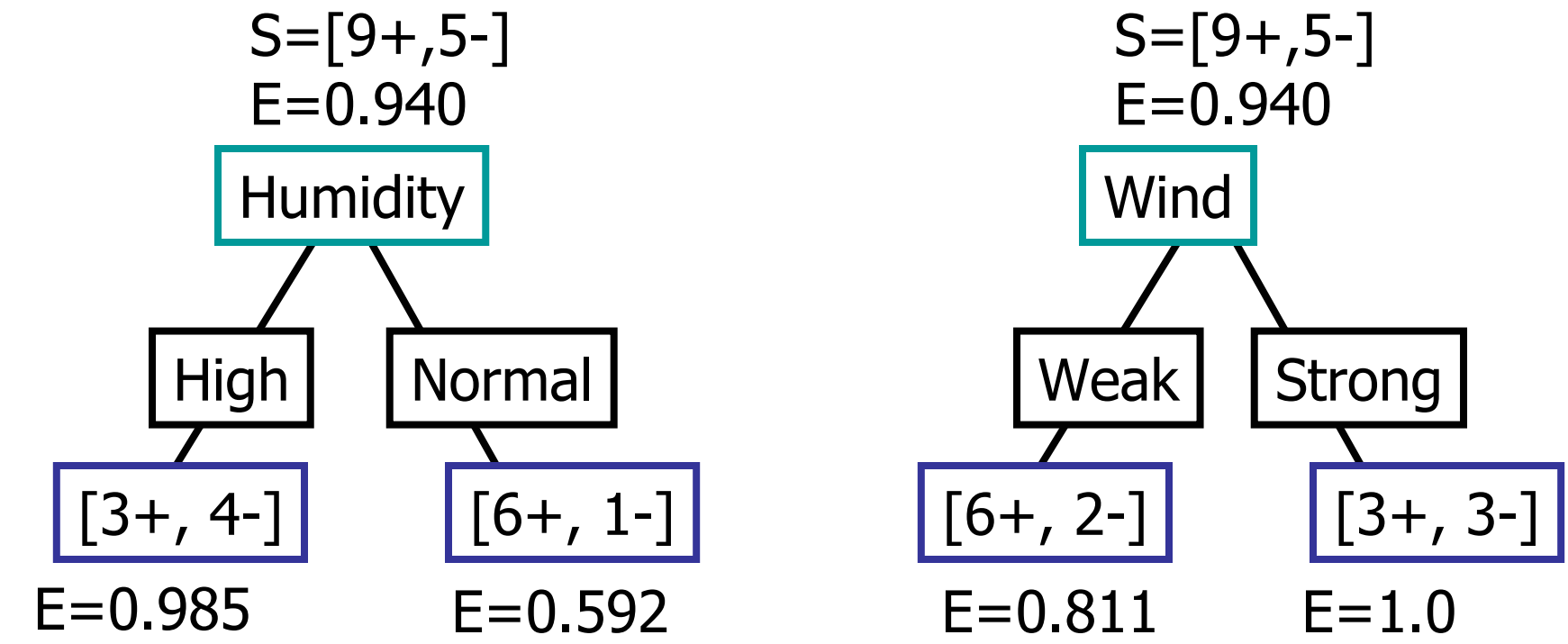




Ejemplo: Datos de Entrenamiento

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

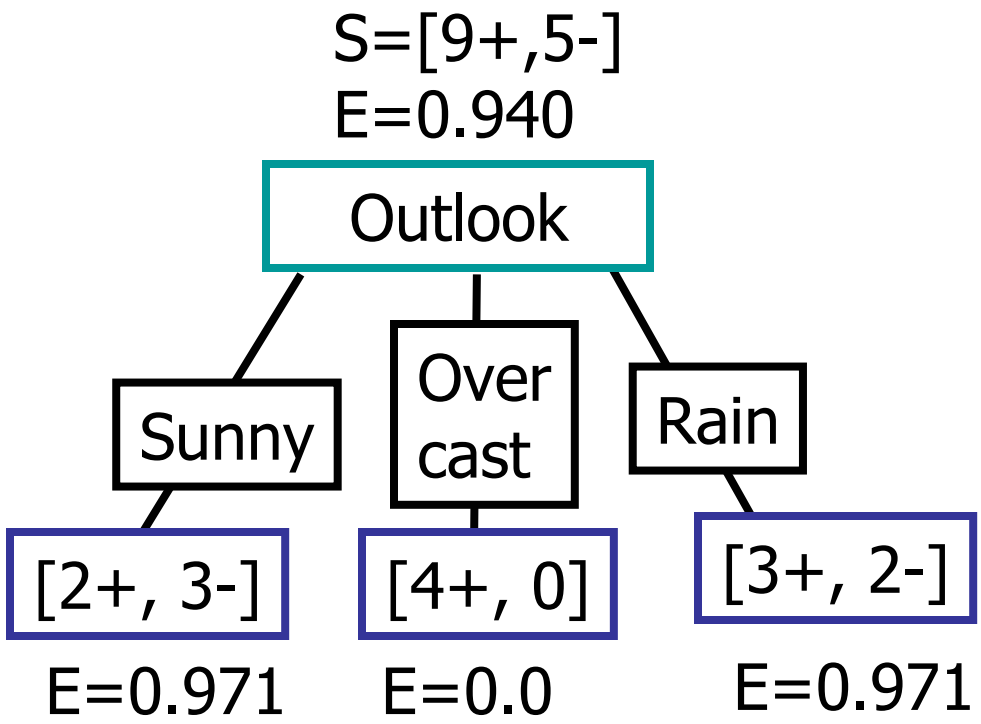
Ejemplo: Elejir Próximo Atributo



Gain(S, Humidity)
= 0.940-(7/14)*0.985
– (7/14)*0.592
= 0.151

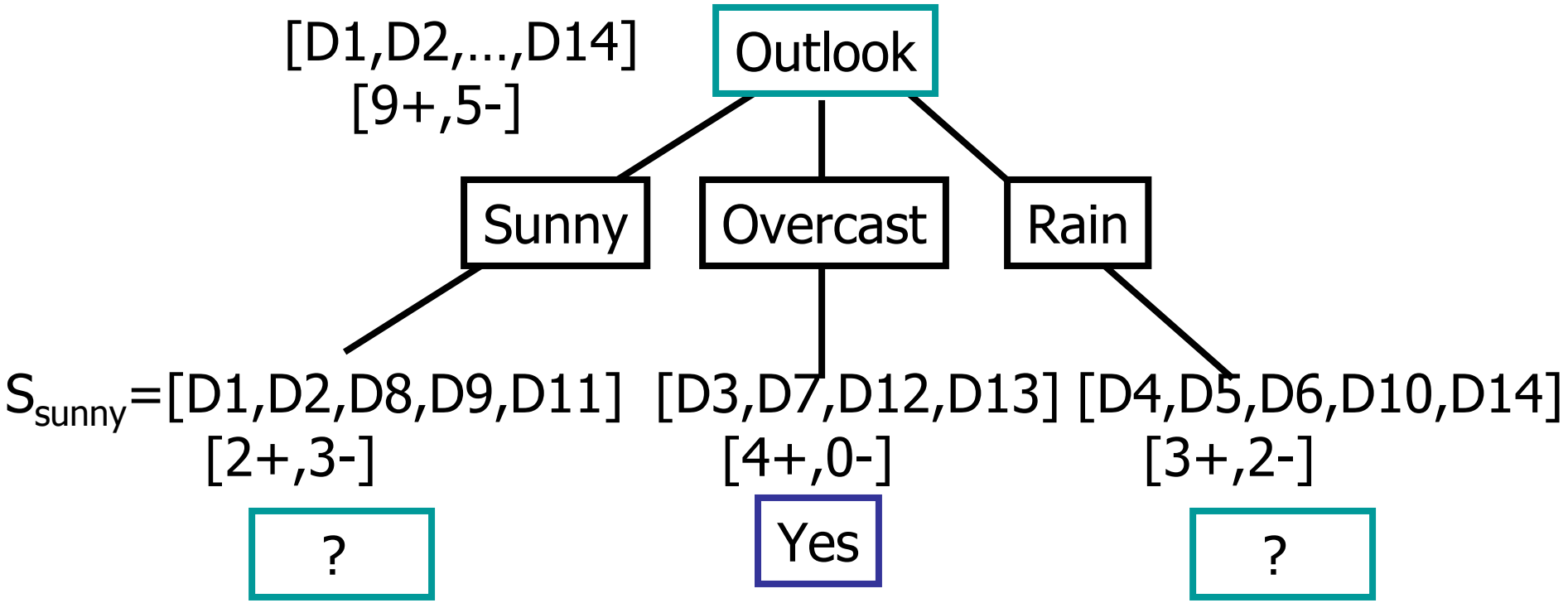
Gain(S, Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
= 0.048

Ejemplo: Elejir Próximo Atributo



Gain(S, Outlook)
= 0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
= 0.247

Ejemplo: Algoritmo ID3

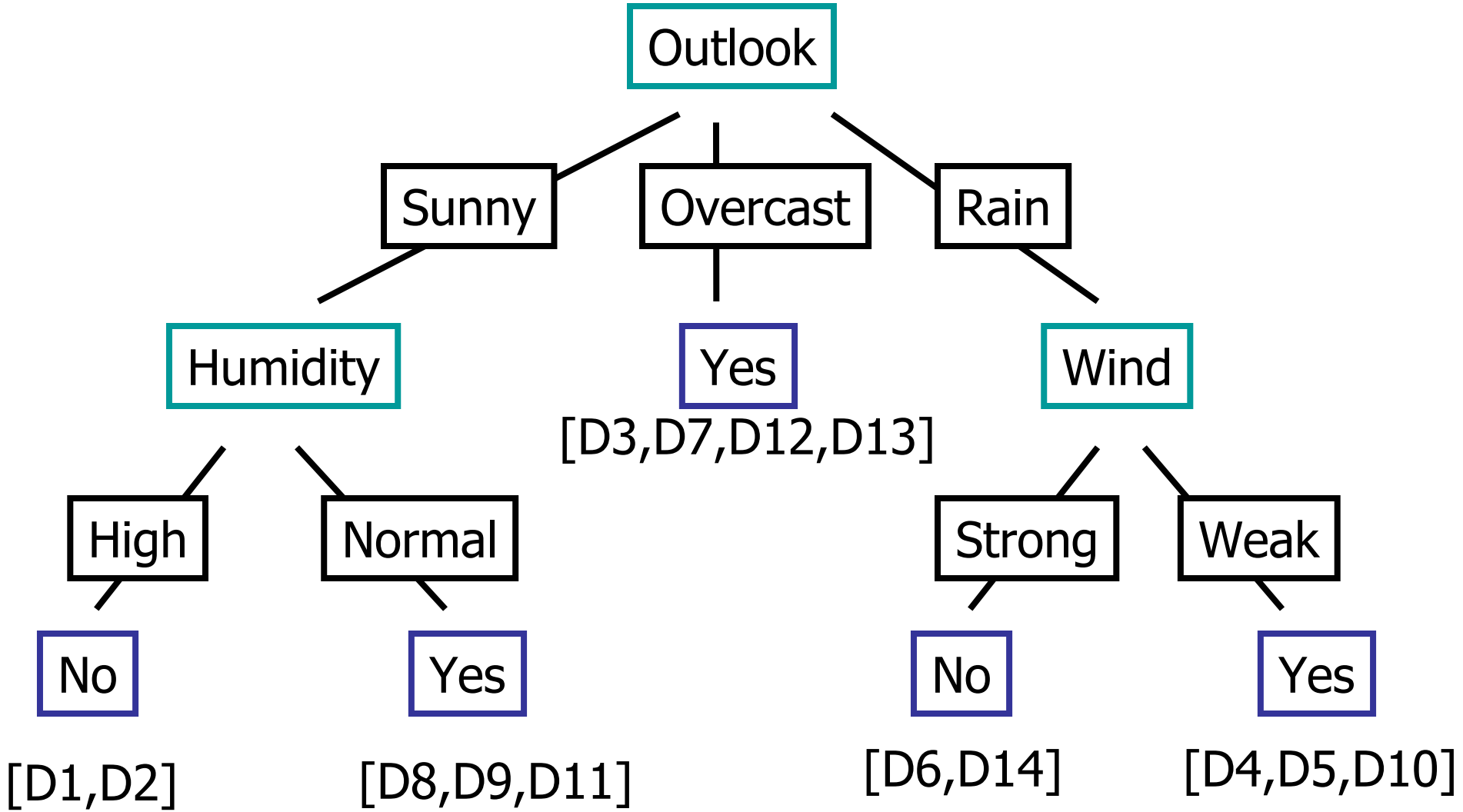


Gain(S_{sunny} , Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970

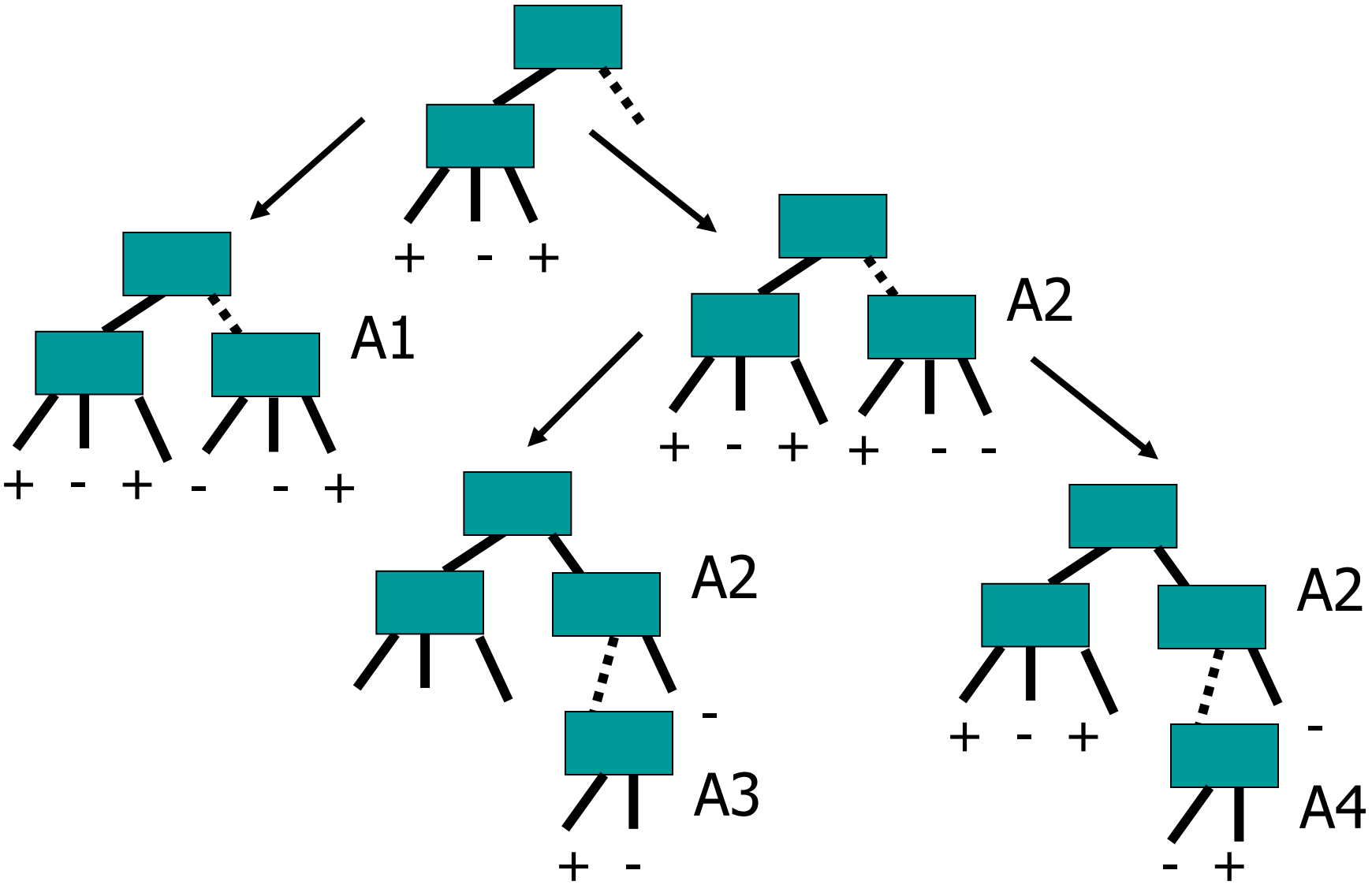
Gain(S_{sunny} , Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570

Gain(S_{sunny} , Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

Ejemplo: Algoritmo ID3



Búsqueda en Espacio de Hipótesis (H) de ID3



Búsqueda en Espacio de Hipótesis (H) de ID3

- Búsqueda en espacio de hipótesis es completa!
  - La función objetiva seguramente esta ahí…
- Output es una hipótesis
- No hay vuelta atrás (backtracking) en los atributos seleccionados (búsqueda codiciosa)
  - Mínima local (divisiones suboptimas)
- Selecciones de búsqueda estadística
  - Robusto a ruidos en datos
- Preferencia de inducción (search bias)
  - Prefiere árboles cortos que largos
  - Pone atributos con alta ganancia de información cerca de la raíz

Búsqueda en Espacio de Hipótesis de ID3

- Desarrolla *aprendizaje en grupos (batch learning)* que procesa todas las instancia de entrenamiento juntos en vez de *aprendizaje incremental (incremental learning)* que actualiza una hipótesis después de cada ejemplo.
- Garantizado de encontrar un árbol consistente con cualquier conjunto de entrenamiento libre de conflictos (i.e. vectores idénticos de características siempre asignados la misma clase), pero no necesariamente el árbol mas simple.
- Encuentra una hipótesis discreta.

Bias en Inducción de Árbol de Decisión

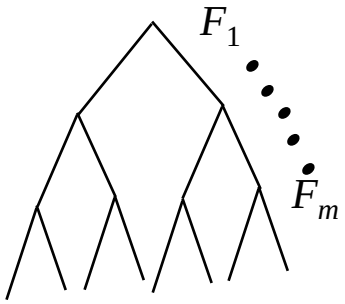
- Ganancia de información da una preferencia (bias) para árboles con profundidad mínima.
- El bias es una preferencia por algunas hipótesis (un bias de búsqueda) y no una restricción de hipótesis en el espacio H.

Ejemplo: Weka

```
data> java weka.classifiers.trees.J48 -t contact-lenses.arff
J48 pruned tree
------------------
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|   astigmatism = no: soft (6.0/1.0)
|   astigmatism = yes
|   |   spectacle-prescrip = myope: hard (3.0)
|   |   spectacle-prescrip = hypermetrope: none (3.0/1.0)

Number of Leaves  :     4
Size of the tree :      7

Time taken to build model: 0.03 seconds
Time taken to test model on training data: 0 seconds

=== Error on training data ===

Correctly Classified Instances          22               91.6667 %
Incorrectly Classified Instances        2                8.3333 %
Kappa statistic                          0.8447
Mean absolute error                      0.0833
Root mean squared error                  0.2041
Relative absolute error                 22.6257 %
Root relative squared error             48.1223 %
Total Number of Instances               24

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  0  3  1 |  b = hard
  1  0 14 |  c = none


=== Stratified cross-validation ===

Correctly Classified Instances          20               83.3333 %
Incorrectly Classified Instances        4                16.6667 %
Kappa statistic                          0.71
Mean absolute error                      0.15
Root mean squared error                  0.3249
Relative absolute error                 39.7059 %
Root relative squared error             74.3898 %
Total Number of Instances               24

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  0  3  1 |  b = hard
  1  2 12 |  c = none
```

Complejidad Computacional

- Peor caso construye un árbol completo en el cual cada camino en el árbol prueba cada característica. Asuma n ejemplos y m características.



Máximo de *n* ejemplos esparcidos en todos los nodos de los *m* niveles

- En cada nivel, *i*, en el árbol, se debe examinar las m-i características que quedan para cada instancia en ese nivel para calcular las ganancias de información.

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

- En la practica un árbol es raramente completo (numero de hojas es $\leq n$) y la complejidad es lineal en m y n.

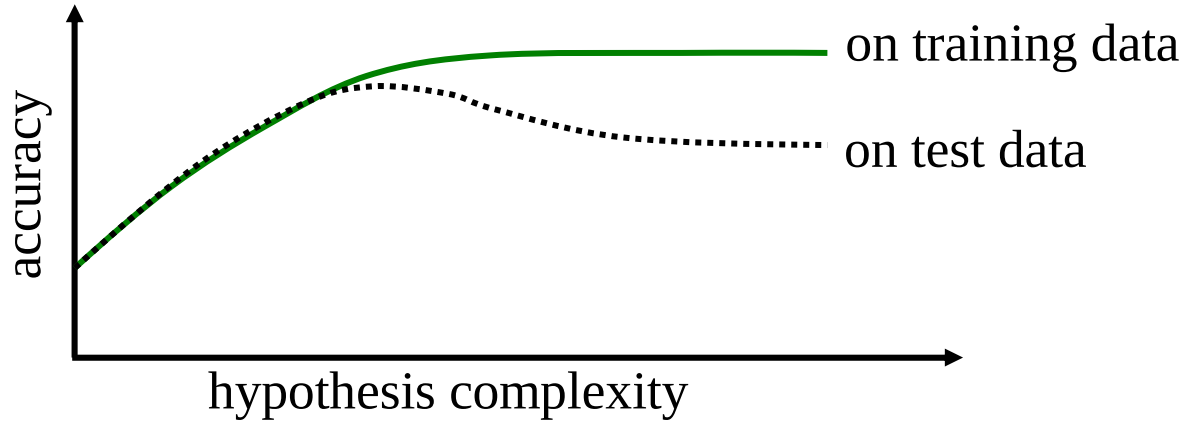
Árboles de Decisión

Contenidos

- Árboles de Decisión
- Sobreajuste
- Recorte (Pruning)

Sobreajuste (Overfitting)

- Crear un árbol que clasifique todos los datos de entrenamiento perfectamente puede no llevarnos a un árbol con la mejor generalización a datos desconocidos.
  - Puede haber error en los datos de entrenamiento que el árbol esta erróneamente ajustando.
  - El algoritmo puede tomar malas decisiones cerca de las ramas basado en pocos datos que no reflejan tendencias confiables.
- Una hipótesis, *h*, se dice que sobreajusta (overfits) los datos de entrenamiento si es que existe otra hipótesis, *h´*, la cual tiene más error que *h* en datos de entrenamiento pero menos error en datos independientes de prueba.

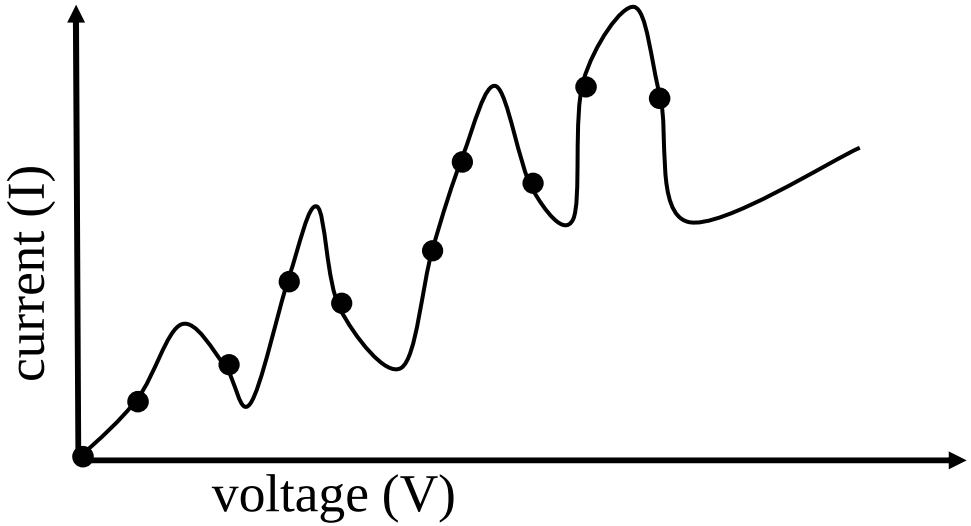


Ejemplo: Sobreajuste

Probando la Ley de Ohm: V = IR (I = (1/R)V)

Experimentalmente se miden 10 puntos

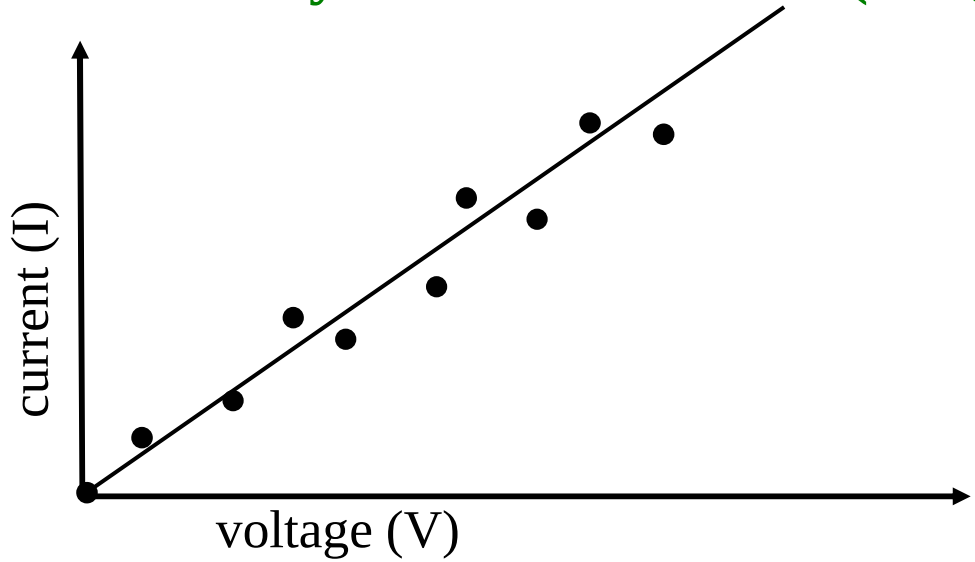
Ajustar una curva a los datos resultantes



Perfectamente se ajustan a los datos de entrenamiento con un polinomio de 9no grado (se pueden ajustar *n* puntos exactamente con un polinomio de grado *n*-1)

Ejemplo: Sobreajuste

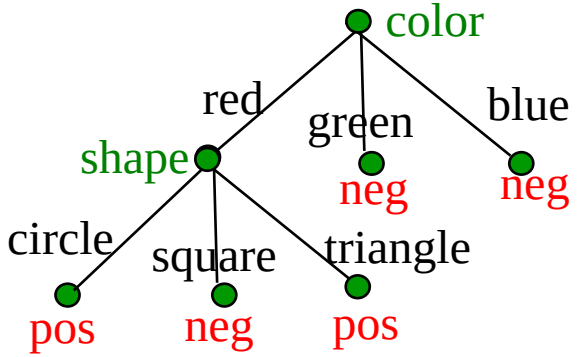
Probando la Ley de Ohm: V = IR (I = (1/R)V)



Se mejora la generalización con una función lineal que se ajusta a los datos de entrenamiento con menor exactitud.

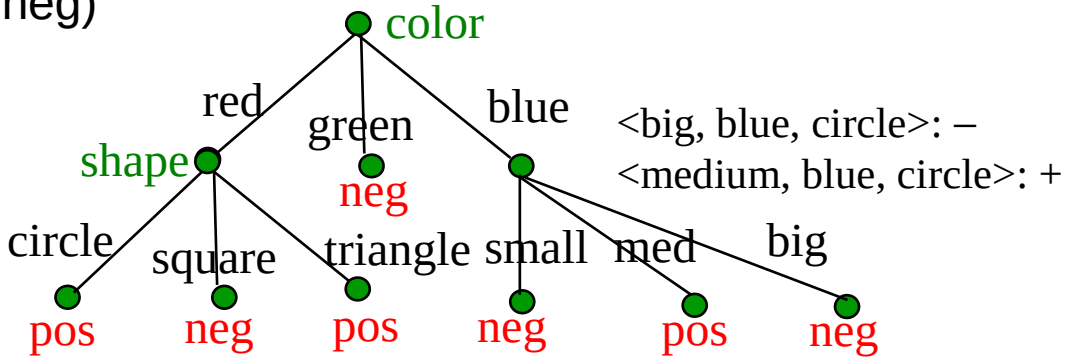
Sobreajuste de Ruido en Árboles de Decisión

- Ruido de características o categoría pueden causar sobreajuste:
  - Agregar instancia (dato) ruidosa: <medium, blue, circle>: pos (pero realmente neg)



Sobreajuste de Ruido en Árboles de Decisión

- Ruido de características o categoría pueden causar sobreajuste:
- Agregar instancia (dato) ruidosa: <medium, blue, circle>: pos (pero realmente neg)



- Ruido también puede causar que diferentes instancias del mismo vector de característica tenga diferentes clases. Es imposible ajustar estos datos y se debe etiquetar la hoja con la clase de la mayoría.
  - <big, red, circle>: neg (but really pos)
- Ejemplos conflictivos pueden surgir si es que las características son incompletas e inadecuadas.

Árboles de Decisión

Contenidos

- Árboles de Decisión
- Sobreajuste
- Recorte (Pruning)

Métodos de Prevención de Sobreajuste (Recorte o Pruning)

- Dos ideas básicas para árboles de decisión
  - Prepruning: Parar de crecer el árbol en algún punto durante construcción top-down cuando no hay suficientes datos para toma de decisiones confiables.
  - Postpruning: Crecer el árbol completo, entonces eliminar subarboles que no tengan suficiente evidencia.
- Etiquetar hoja que resulta de un recorte con la clase de la mayoría de los datos que quedan o con la distribución de probabilidades de la clase.
- Métodos para elegir subarboles a ser recortados:
  - Validacion-cruzada: Reservar algunos datos de entrenamiento (***validation set***, ***tuning set***) para evaluar utilidad de subarboles.
  - Test estadístico: Usar un test estadístico en los datos de entrenamiento para determinar si alguna regularidad observada se puede eliminar por ser simplemente aleatoria.
  - Minimum description length (MDL): Determinar si la complejidad adicional de la hipótesis es menos compleja que explícitamente recordar excepciones resultantes del recorte.

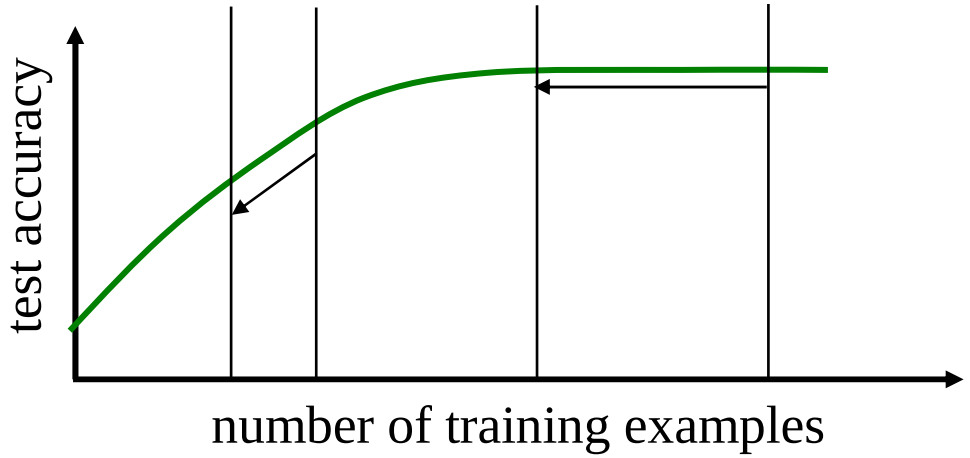
Reduced Error Pruning

- Un método basado en post-pruning y validación cruzada

Partition training data in “grow” and “validation” sets.
Build a complete tree from the “grow” data.
Until accuracy on validation set decreases do:
    For each non-leaf node, n, in the tree do:
        Temporarily prune the subtree below n and replace it with a
            leaf labeled with the current majority class at that node.
        Measure and record the accuracy of the pruned tree on the validation set.
    Permanently prune the node that results in the greatest increase in accuracy on
        the validation set.

Problemas con Reduced Error Pruning

- El problema con este método es que potencialmente “desperdicia” datos de entrenamiento en el conjunto de validacion.
- Severidad de este problema depende de adonde estamos en la curva de aprendizaje:



Validación cruzada sin Perder Datos de Entrenamiento

- Se modifica el algoritmo de crecimiento para crecer al ancho primero y se para de crecer después de llegar a cierta complejidad especifica de un árbol.
- Requiere varias pruebas de diferentes recortes usando divisiones aleatorias de conjuntos de crecimiento y validación
- Se determina la complejidad del árbol y se calcula un promedio de complejidad C
- Se comienza crecer el árbol de todos los datos de entrenamiento pero se detiene cuando la complejidad llega a C.

Otras ideas en Árboles de Decisión

- Mejores criterios de división
  - Ganancia de información prefiere características con muchos valores.
- Características continuas
- Predicción de funciones de valores reales (árboles de regresión)
- Características con costos
- Costos de misclasificación
- Aprendizaje incremental
  - ID4
  - ID5
- Grandes bases de datos que no caben en memoria
- Aplicaciones híbridas (i.e. robótica, reconocimiento patrones, …)

Árboles de Decisión

Referencias:

[1] Mitchel, T., Machine Learning , McGraw Hill, 1997

[2] Karray, F., De Silva, C., Soft Computing and Intelligent Systems Design, Addison Wesley, 2004

[3] Mooney, R., Machine Learning Slides, University of Texas

[4] Hoffman, F., Machine Learning Slides, Stockholm University