

CI5438. Inteligencia Artificial II  
Clase 5: Aprendizaje en Árboles de Decisión  
(Continuación)  
Cap 18.3: RN  
Cap 3: Mitchell

Ivette C. Martínez

Universidad Simón Bolívar

7 de octubre de 2009

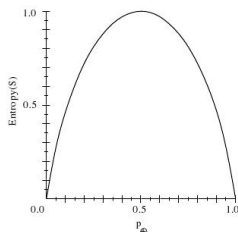
# Algoritmo de Inducción de Árboles de Decisión

[Mitchell]

Ciclo principal:

- 1  $A \leftarrow$  El mejor atributo para el próximo *nodo*
- 2 Asignar  $A$  como el atributo de decisión para el *nodo*
- 3 Para cada valor de  $A$ , crear un nuevo *nodo* descendiente.
- 4 Ordenar los ejemplos en las nodos hojas
- 5 Si los ejemplos de entrenamiento están perfectamente clasificados entonces STOP, si no iterar sobre los nodos hojas.

# Entropía



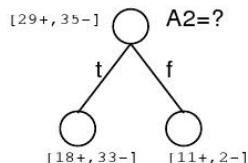
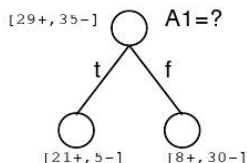
- La entropía mide la impureza de  $S$
- $Entropia(S)$  = El número esperado de bits necesarios para codificar una clase ( $\oplus$  o  $\ominus$ ) de un elemento de  $S$  seleccionado aleatoriamente (bajo la codificación óptima de menor longitud).

$$Entropy(S) = -p_{\oplus} \log_2(p_{\oplus}) - p_{\ominus} \log_2(p_{\ominus})$$

# Ganancia de Información

$Gain(S, A) =$  Es la reducción esperado de la entropía de  $S$  debido al ordenamiento  $A$ .

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

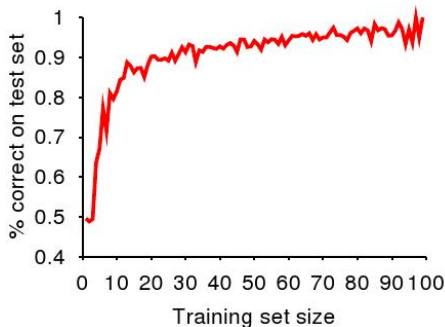


# Desempeño del algoritmo de Aprendizaje

Cómo sabemos si  $h \approx f$ ?

- 1 Usar los teoremas de la teoría de aprendizaje estadístico/computacional
- 2 Probar  $h$  en un nuevo conjunto de ejemplos de prueba.

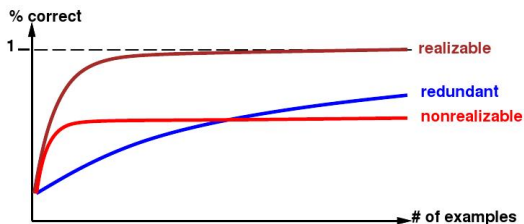
Curva de aprendizaje = % de aciertos en el conjunto de prueba como función del tamaño del conjunto de entrenamiento



# Desempeño del algoritmo de Aprendizaje

Las curvas de aprendizaje dependen de

- realizable vs. no-realizable
  - Lo no-realizable puede deberse a atributos faltantes o a una clase de hipótesis restringida
- Expsividad Redundante (i.e., atributos irrelevantes)



# Desempeño del algoritmo de Aprendizaje

- Un algoritmo de aprendizaje es bueno si produce hipótesis que predicen adecuadamente la clasificación de ejemplos desconocidos.
- Una metodología para verificar la calidad de la predicción:
  - 1 Coleccionar un gran conjunto de ejemplos
  - 2 Separarlo en dos conjuntos disjuntos: el conjunto de entrenamiento y el conjunto de prueba
  - 3 Usar en algoritmo en es conjunto de entrenamiento para generar la hipótesis  $h$
  - 4 Medir el porcentaje de los ejemplos del conjunto de pruebas que son clasificados correctamente por  $h$
  - 5 Repetir los pasos 1 al 4 variando el tamaño del conjunto de entrenamiento y seleccionando aleatoriamente los elementos para cada tamaño

# Overfitting

Consideremos el error de la hipótesis  $h$  sobre:

- Los datos de entrenamiento:  $error_{train}(h)$
- La distribución completa  $\mathcal{D}$  de los datos:  $error_{\mathcal{D}}(h)$

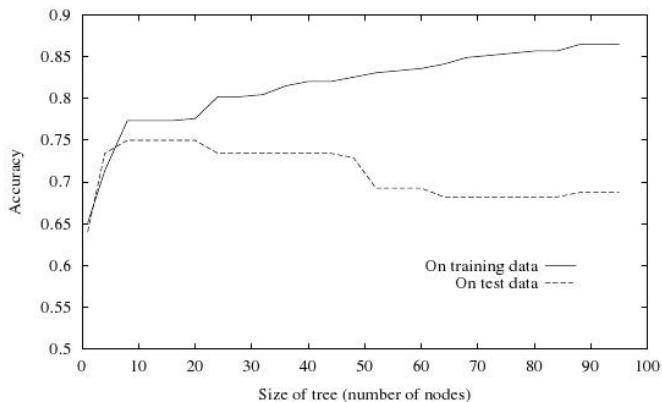
Se dice que la hipótesis  $h \in H$  *overfits* los datos de entrenamiento si existe una hipótesis alternativa  $h' \in H$  tal que:

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

# Overfitting



# Cómo evitar el overfitting?

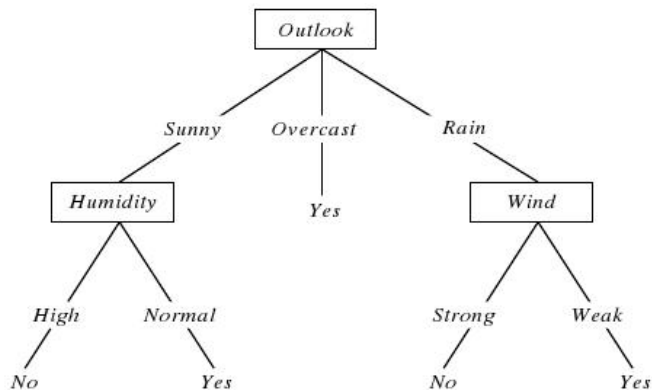
Cómo podemos evitar el overfitting?

- Para de crecer cuando la separación de los datos no es estadísticamente significativa .
- Crecer el árbol completo, luego podarlo

Cómo seleccionar el mejor árbol

- Medir en performance sobre los datos de entrenamiento
- Medir el performance sobre un conjunto de validación diferente (*test set*).
- Minimizar:  
 $size(tree) + size(misclassifications(tree))$

# Convirtiendo un árbol en reglas



# Convirtiendo un árbol en reglas

IF  $(Outlook = Sunny) \wedge (Humidity = High)$   
THEN  $PlayTennis = No$

IF  $(Outlook = Sunny) \wedge (Humidity = Normal)$   
THEN  $PlayTennis = Yes$

...

# Atributos con valores continuos

Crear un atributo discreto para realizar las pruebas sobre continuos.

- $Temperatura = 82,5$
- $(Temperatura > 72,3) = t, f$

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

# Atributos con muchos valores

Problema:

- Si un atributo tiene muchos valores, *Gain* lo seleccionará
- Ejm: Date = Jun\_3\_1996.

Una solución es usar el *GainRatio*

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

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

Ejemplo:

- En diagnóstico Médico, *BloodTest* tiene un costo de \$150

Cómo aprender un árbol consistente con un costo relativamente bajo?

Una solución: reemplazar *Gain* por:

- Tan and Schlimmer (1990)

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

- Nunez (1988)

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

donde  $w \in [0, 1]$  determina la importancia del costo

Qué pasa si algunos ejemplos no tienen valores para  $A$ ?

Usar estos ejemplos de todas maneras, ordenándolos con el árbol

- Si el nodo  $n$  prueba  $A$ , asignarle el valor más común de  $A$  entre los ejemplos en el nodo  $n$
- Asignarle el valor más común de  $A$  entre otros ejemplos con el mismo valor objetivo
- Asigne una probabilidad  $p_i$  a cada posible valor  $v_i$  de  $A$ 
  - Asignarle una fracción  $p_i$  del ejemplo a cada descendiente en el árbol

Clasificar los nuevos ejemplos de la misma manera