HTF: 9.2
B: 14.4
RN, Chapter 18 – 18.3

Decision Tree

R Greiner

Cmput 466 / 551

Learning Decision Trees

- Def'n: Decision Trees
- Algorithm for Learning Decision Trees
  - Entropy, Inductive Bias (Occam's Razor)
- Overfitting
  - Def’n, MDL, χ^2, PostPruning
- Topics:
  - k-ary attribute values
  - Real attribute values
  - Other splitting criteria
  - Attribute Cost
  - Missing Values
  - ...

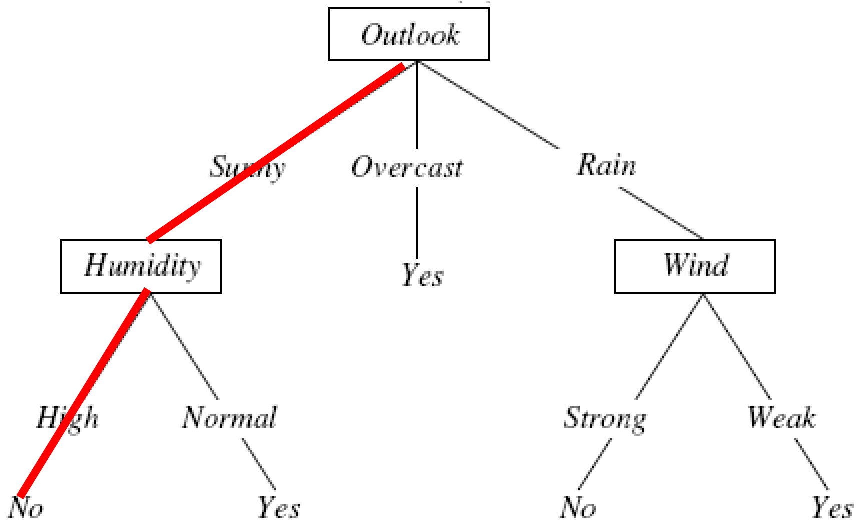
DecisionTree Hypothesis Space

- Internal nodes labeled with some feature x_j
- Arc (from x_j) labeled with results of test x_j
- Leaf nodes specify class $h(x)$

- Instance:

Outlook	= Sunny
Temperature	= Hot
Humidity	= High
Wind	= Strong

  classified as "No"
  - (Temperature, Wind: irrelevant)
- Easy to use in Classification
  - Answer short series of questions...



Decision Trees

Hypothesis space is. . .

- Variable Size: Can represent any boolean function
- Deterministic
- Discrete and Continuous Parameters

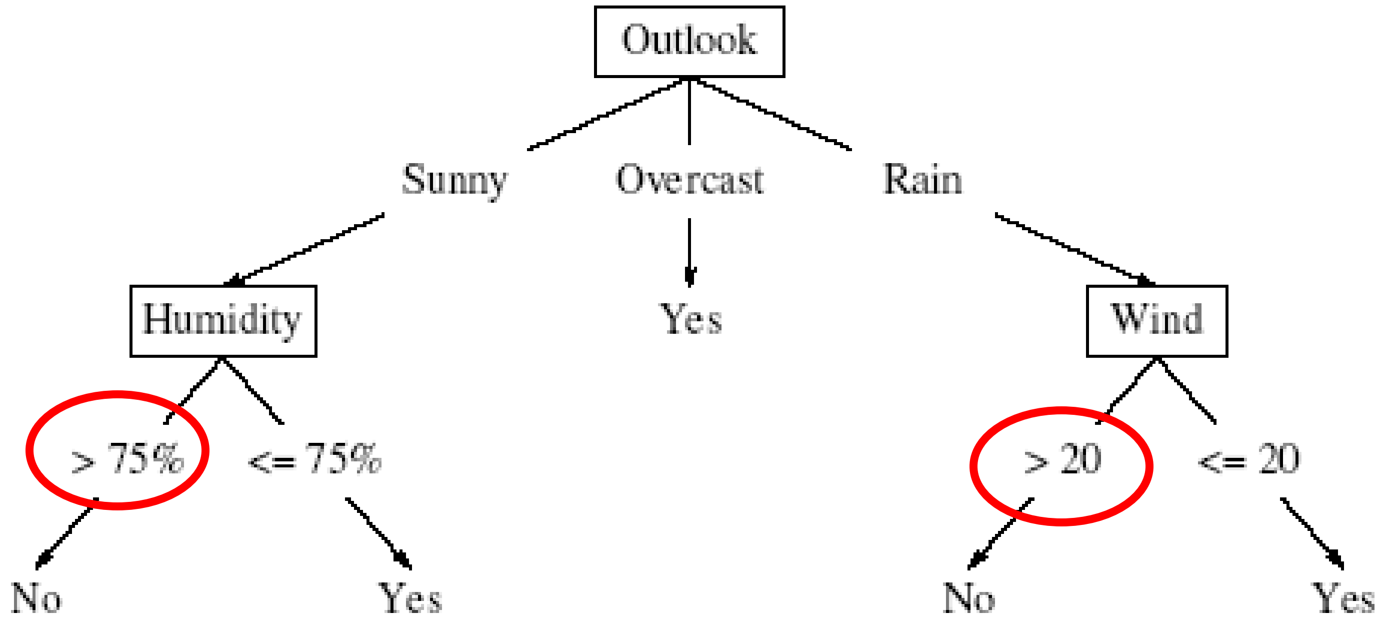
Learning algorithm is. . .

- Constructive Search: Build tree by adding nodes
- Eager
- Batch (although ∃ online algorithms)

Continuous Features

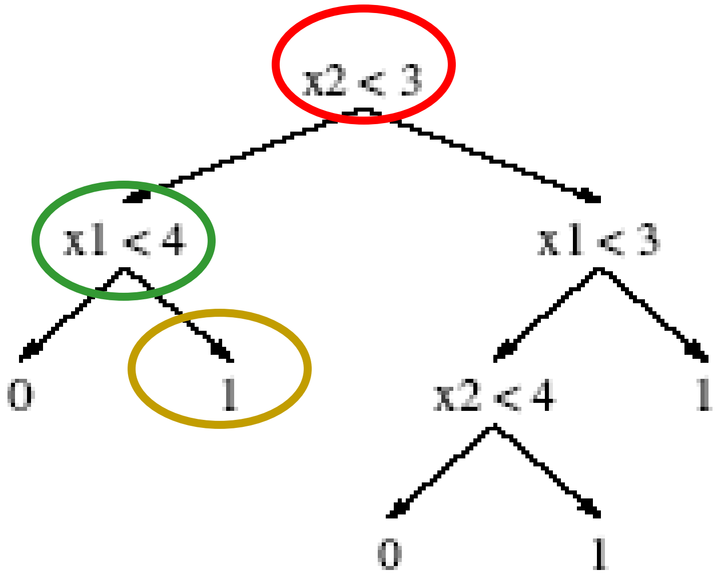
- If feature is continuous:

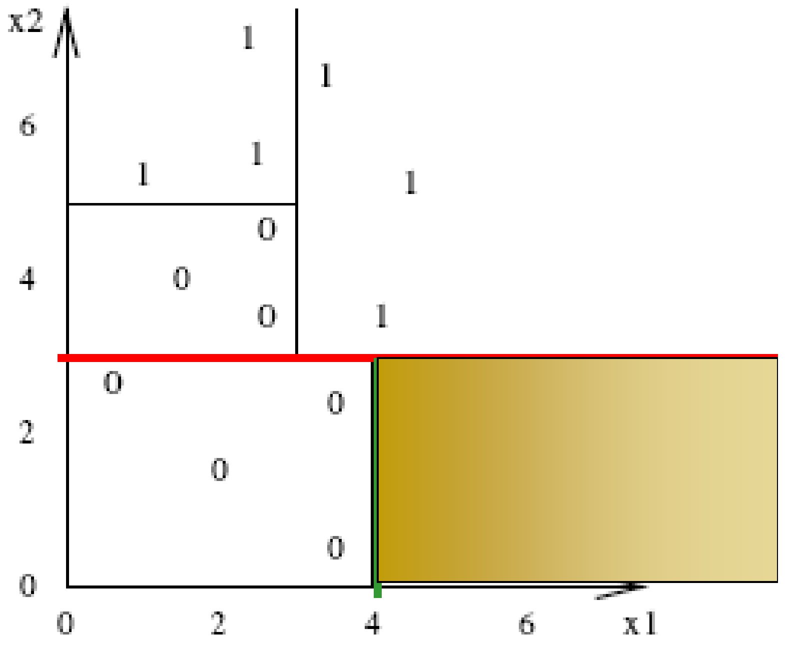
  internal nodes may test value against threshold



DecisionTree Decision Boundaries

- Decision trees divide feature space into axis-parallel rectangles, labeling each rectangle with one class

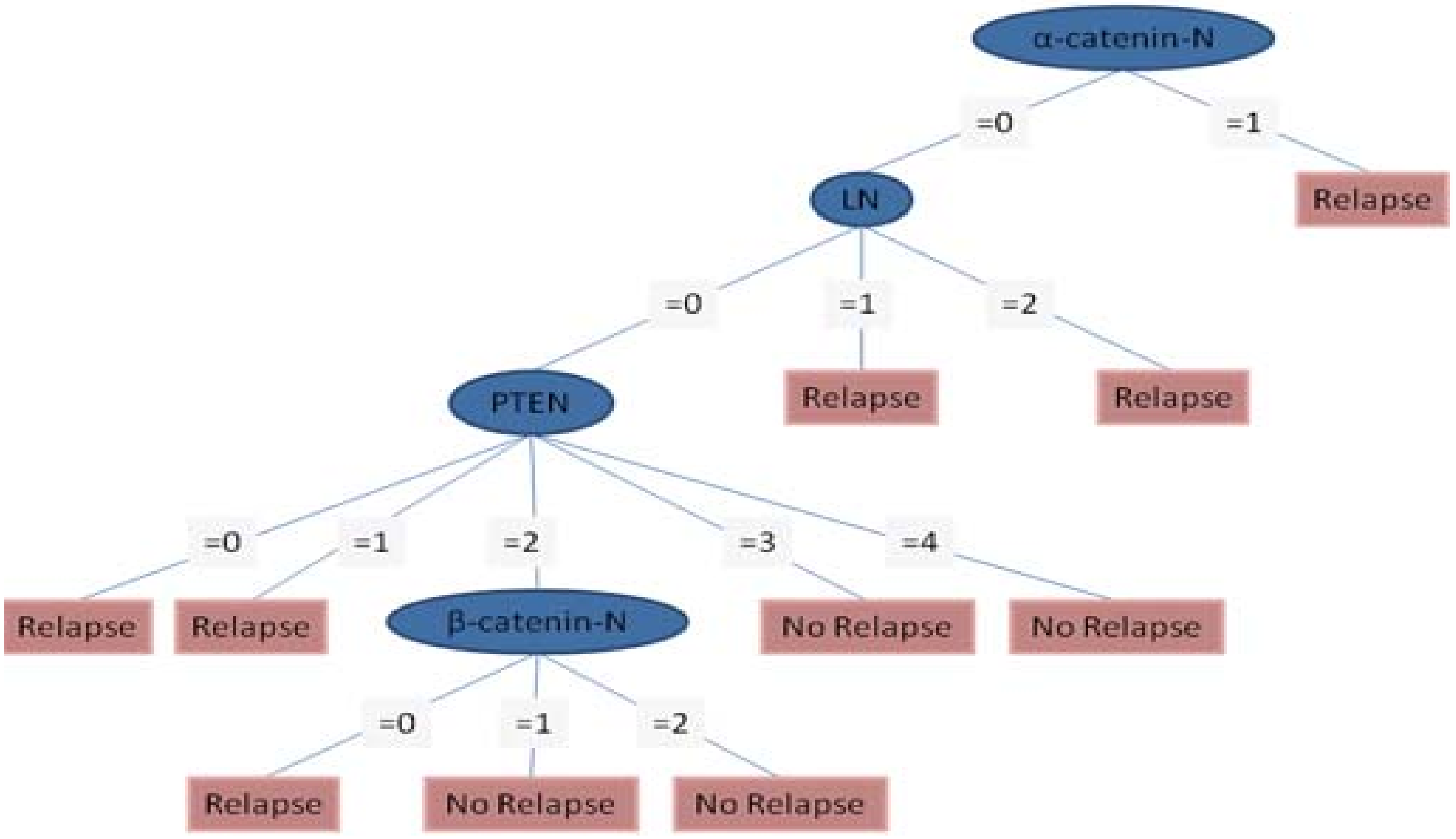




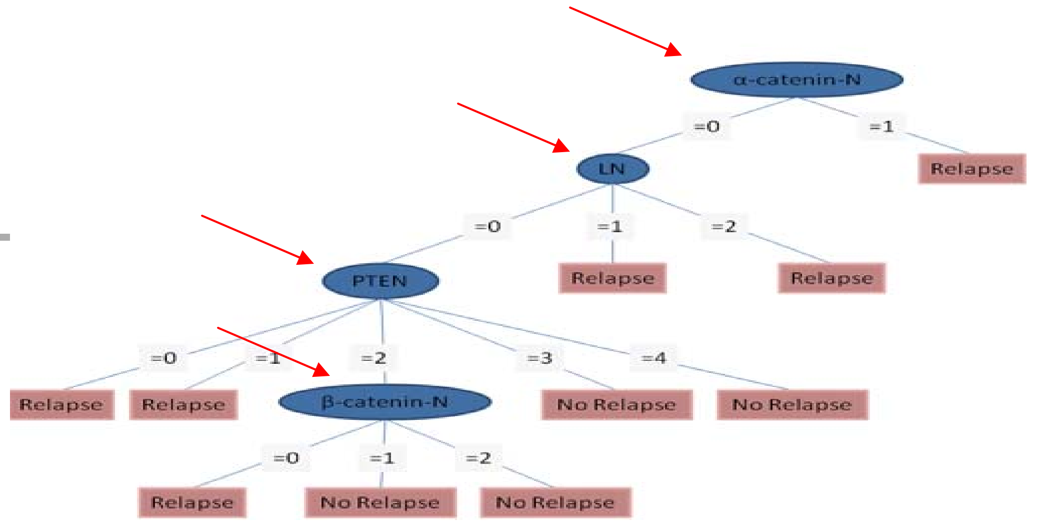
Using Decision Trees

- Instances represented by Attribute-Value pairs
  - "Bar = Yes", "Size = Large", "Type = French", "Temp = 82.6", ...
  - (Boolean, Discrete, Nominal, Continuous)
- Can handle:
  - Arbitrary DNF
  - Disjunctive descriptions
- Our focus:
  - Target function output is discrete
  - (DT also work for continuous outputs [regression])
- Easy to EXPLAIN
- Uses:
  - Credit risk analysis
  - Modeling calendar scheduling preferences
  - Equipment or medical diagnosis

Learned Decision Tree



Meaning



- Concentration of α-catenin in nucleus is very important:
  - If >0, probably relapse
- If =0, then #lymph_nodes is important:
  - If >0, probably relaps
- If =0, then concentration of pten is important:
  - If <2, probably relapse
  - If >2, probably NO relapse
- If =2, then concentration of β-catenin in nucleus is important:
  - If =0, probably relapse
  - If >0, probably NO relapse

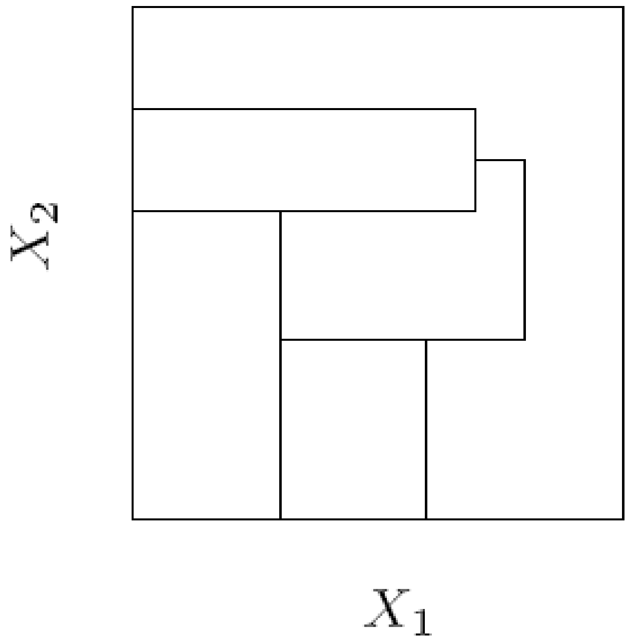
Can Represent Any Boolean Fn

- v, &, ⊕, MofN

  (A v B) & (C v ¬D v E)

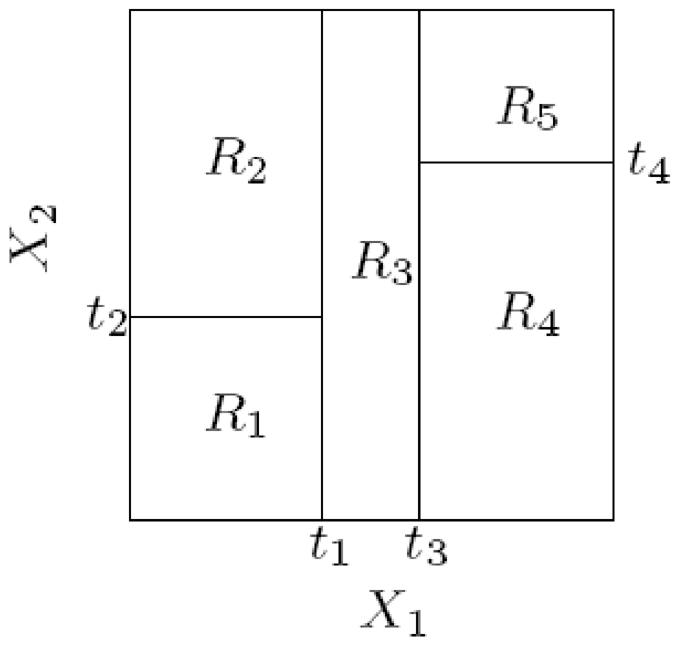
  . . . but may require exponentially many nodes. . .
- **Variable-Size Hypothesis Space**
  - Can "grow" hypothesis space by increasing number of nodes
  - **depth 1** ("decision stump"):
    represent any boolean function of one feature
  - **depth 2**: Any boolean function of two features;
    + some boolean functions involving three features
    (x1 v x2) & (¬ x1 v ¬ x3)
  - ...

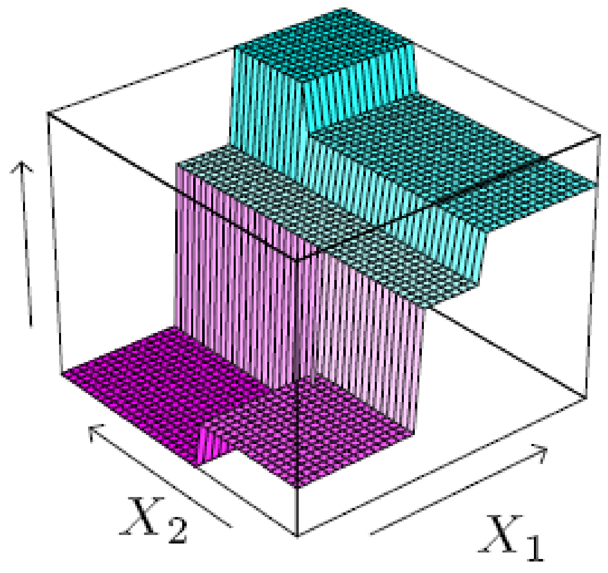
May require >2-ary splits



- Cannot represent using Binary Splits

Regression (Constant) Tree



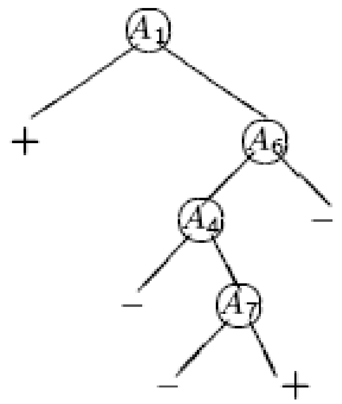


- Represent each region as CONSTANT

Learning Decision Tree

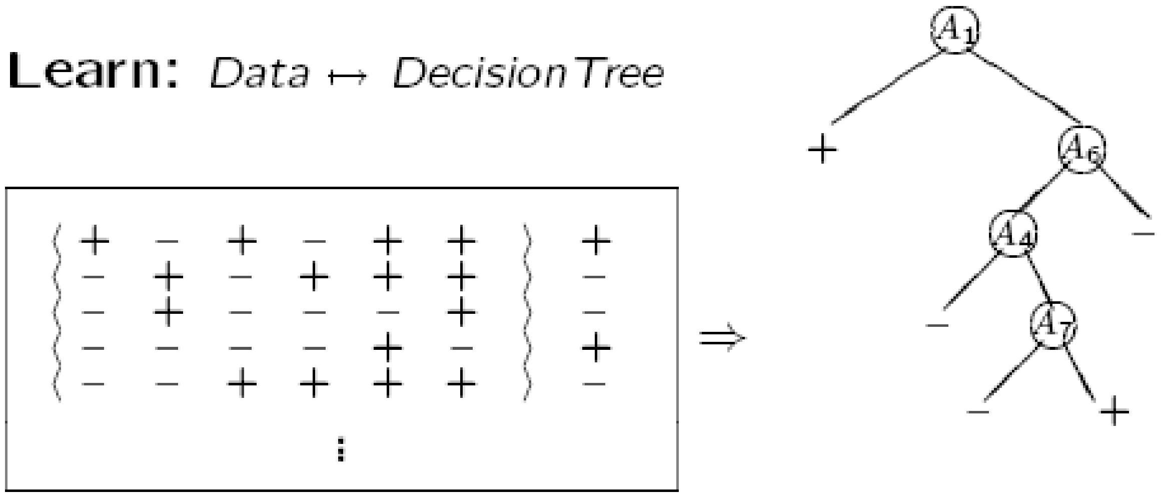
Classify: $DecisionTree \times Instance \mapsto ClassLabel$

Given Decision Tree

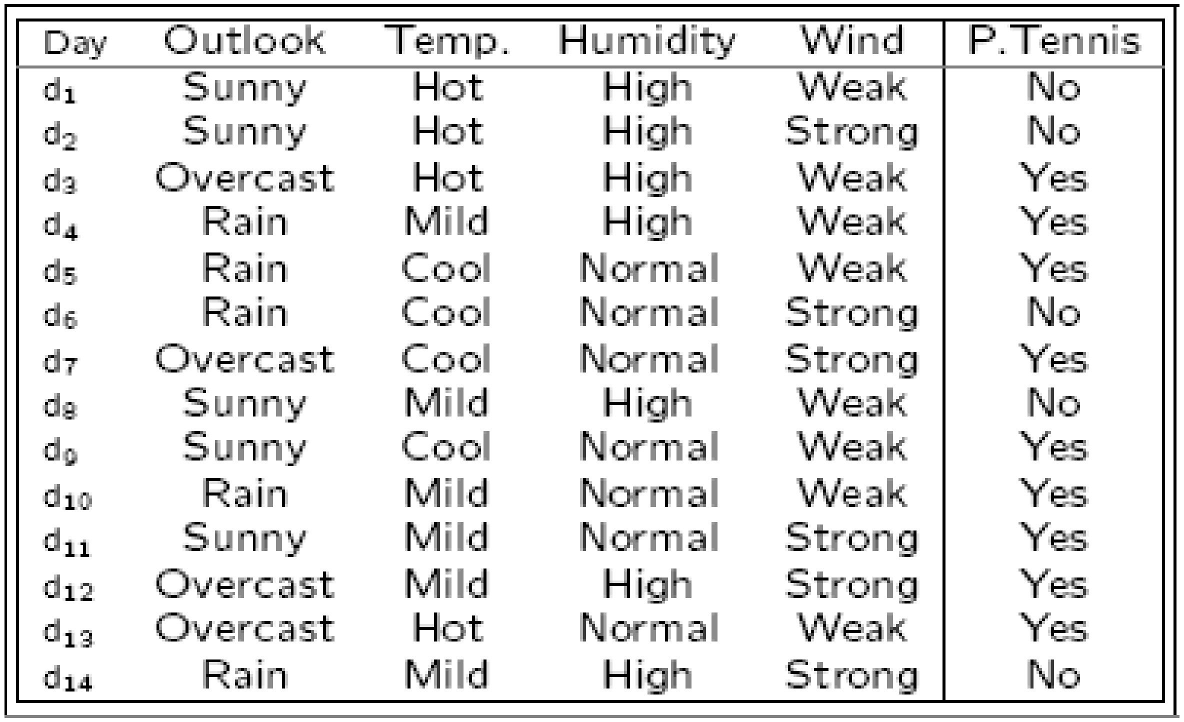


can classify (new) instance $\langle 1,\ 0.2,\ -1,\ false,\ \ldots,\ 1\rangle$ as false

Learn: $Data \mapsto DecisionTree$



Training Examples



Day	Outlook	Temp.	Humidity	Wind	P.Tennis
d_1	Sunny	Hot	High	Weak	No
d_2	Sunny	Hot	High	Strong	No
d_3	Overcast	Hot	High	Weak	Yes
d_4	Rain	Mild	High	Weak	Yes
d_5	Rain	Cool	Normal	Weak	Yes
d_6	Rain	Cool	Normal	Strong	No
d_7	Overcast	Cool	Normal	Strong	Yes
d_8	Sunny	Mild	High	Weak	No
d_9	Sunny	Cool	Normal	Weak	Yes
d_{10}	Rain	Mild	Normal	Weak	Yes
d_{11}	Sunny	Mild	Normal	Strong	Yes
d_{12}	Overcast	Mild	High	Strong	Yes
d_{13}	Overcast	Hot	Normal	Weak	Yes
d_{14}	Rain	Mild	High	Strong	No

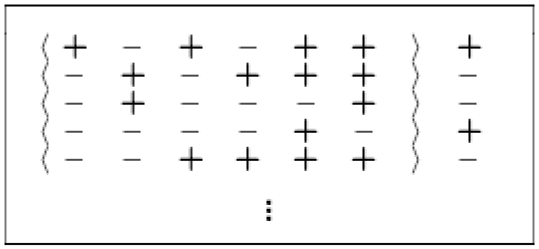
- 4 discrete-valued attributes
- "Yes/No" classification

Want: Decision Tree

DT_{PT} (*Out, Temp, Humid, Wind*) $\in$ { Yes, No }

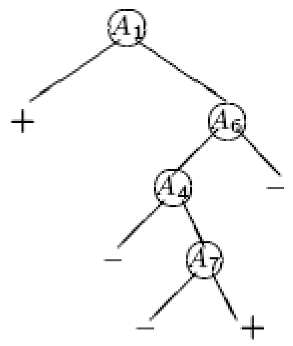
Learning Decision Trees – Easy?

- Learn: Data $\mapsto$ DecisionTree

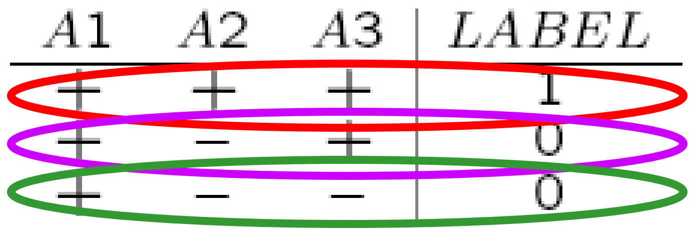


$$\begin{array}{c} \langle + \; - \; + \; - \; + \; + \rangle \; + \\ \langle - \; + \; - \; + \; + \; + \rangle \; - \\ \langle - \; + \; - \; - \; - \; + \rangle \; - \\ \langle - \; - \; - \; - \; + \; - \rangle \; + \\ \langle - \; - \; + \; + \; + \; + \rangle \; - \\ \vdots \end{array}$$

$\Rightarrow$

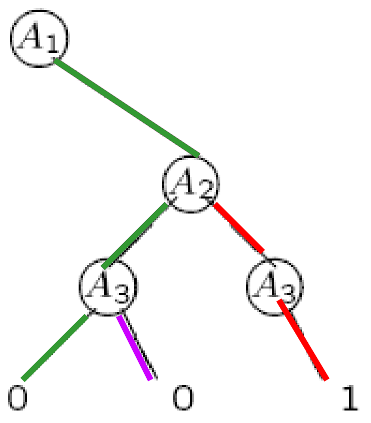


- Option 1: Just store training data



$A1$	$A2$	$A3$	$LABEL$
+	+	+	1
+	−	+	0
+	−	−	0

$\Rightarrow$



- But ...

Learn ?Any? Decision Tree

- Just produce "path" for each example
  - May produce large tree
  - Any generalization? (what of other instances?)

    $\langle -, ++ \rangle$?
  - Noise in data

    $\langle \langle +, -, - \rangle, 0 \rangle$

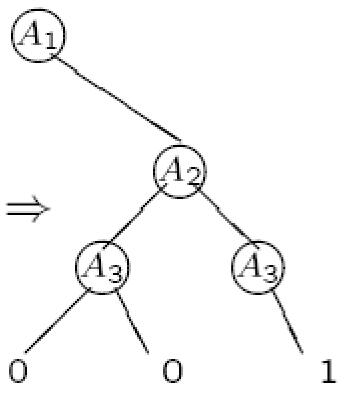
    mis-recorded as

    $\langle \langle +, \mathbf{+}, - \rangle, 0 \rangle$

    $\langle \langle +, -, - \rangle, \mathbf{1} \rangle$

$A1$	$A2$	$A3$	$LABEL$
+	+	+	1
+	−	+	0
+	−	−	0

$\Rightarrow$



- Intuition:

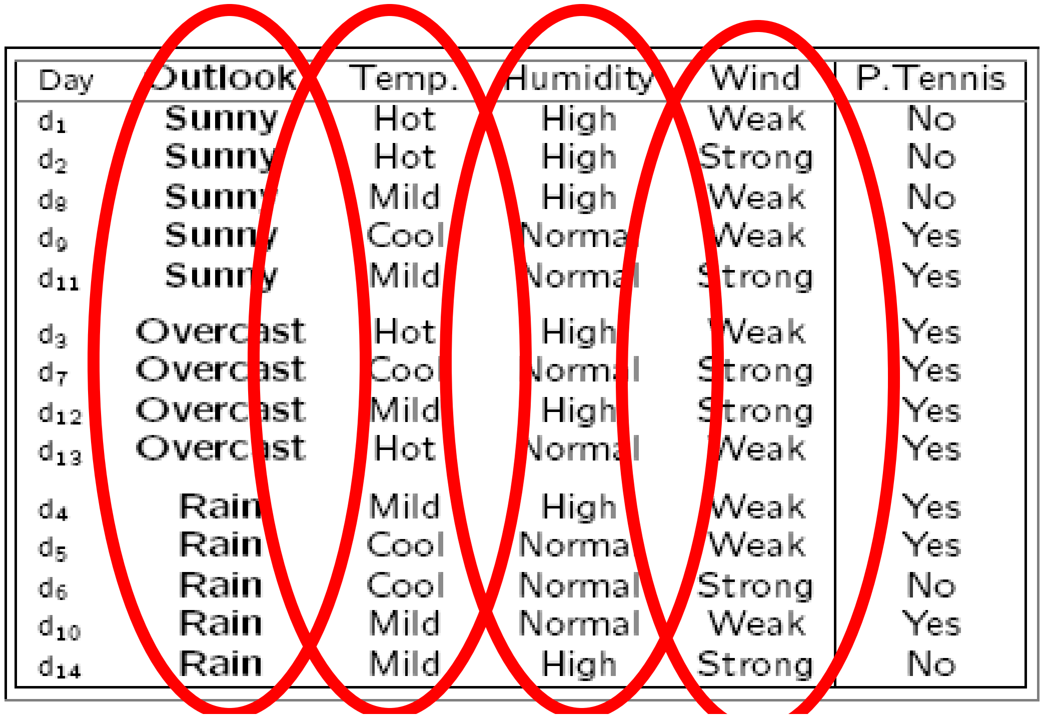
  Want SMALL tree

  … to capture "regularities" in data …

  … easier to understand, faster to execute, …

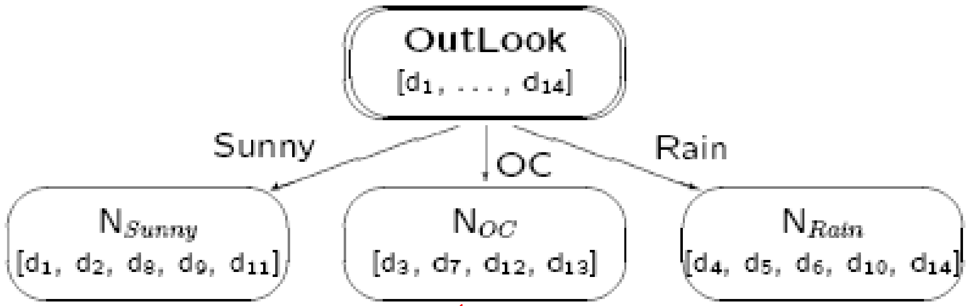
First Split?

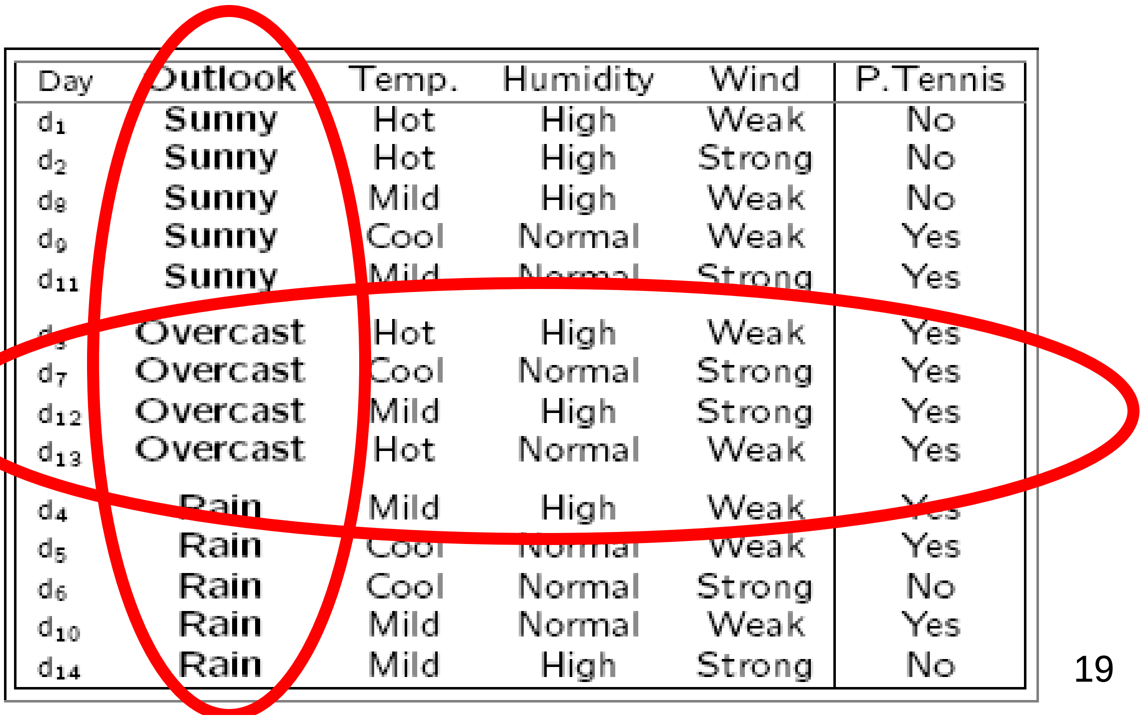
??



Day	Outlook	Temp.	Humidity	Wind	P.Tennis
d_1	Sunny	Hot	High	Weak	No
d_2	Sunny	Hot	High	Strong	No
d_8	Sunny	Mild	High	Weak	No
d_9	Sunny	Cool	Normal	Weak	Yes
d_{11}	Sunny	Mild	Normal	Strong	Yes
d_3	Overcast	Hot	High	Weak	Yes
d_7	Overcast	Cool	Normal	Strong	Yes
d_{12}	Overcast	Mild	High	Strong	Yes
d_{13}	Overcast	Hot	Normal	Weak	Yes
d_4	Rain	Mild	High	Weak	Yes
d_5	Rain	Cool	Normal	Weak	Yes
d_6	Rain	Cool	Normal	Strong	No
d_{10}	Rain	Mild	Normal	Weak	Yes
d_{14}	Rain	Mild	High	Strong	No

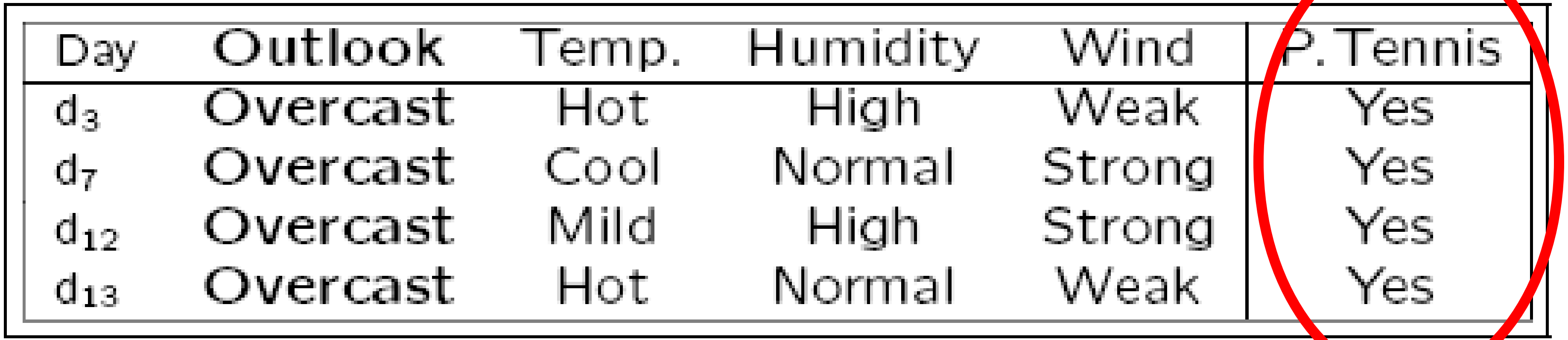
First Split: **Outlook**



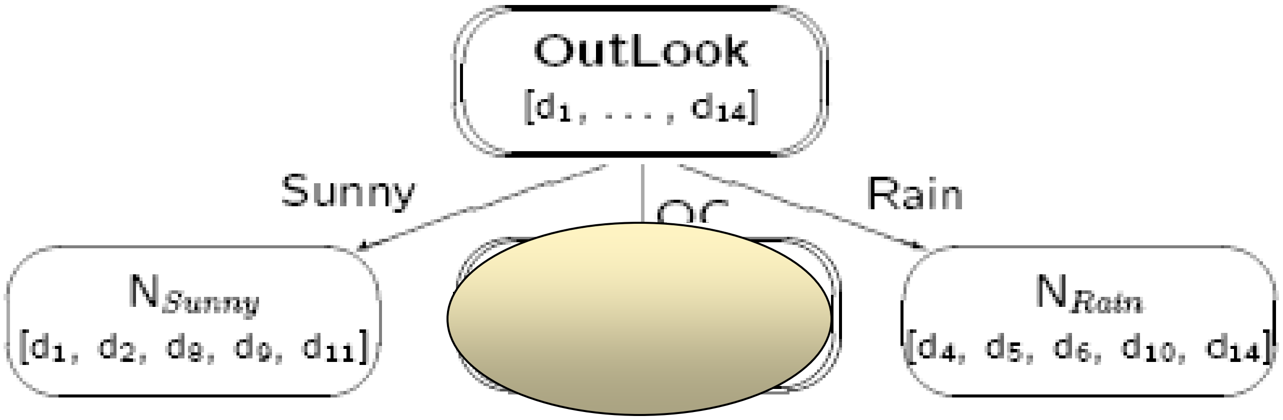


Day	Outlook	Temp.	Humidity	Wind	P.Tennis
d_1	Sunny	Hot	High	Weak	No
d_2	Sunny	Hot	High	Strong	No
d_8	Sunny	Mild	High	Weak	No
d_9	Sunny	Cool	Normal	Weak	Yes
d_{11}	Sunny	Mild	Normal	Strong	Yes
d_3	Overcast	Hot	High	Weak	Yes
d_7	Overcast	Cool	Normal	Strong	Yes
d_{12}	Overcast	Mild	High	Strong	Yes
d_{13}	Overcast	Hot	Normal	Weak	Yes
d_4	Rain	Mild	High	Weak	Yes
d_5	Rain	Cool	Normal	Weak	Yes
d_6	Rain	Cool	Normal	Strong	No
d_{10}	Rain	Mild	Normal	Weak	Yes
d_{14}	Rain	Mild	High	Strong	No

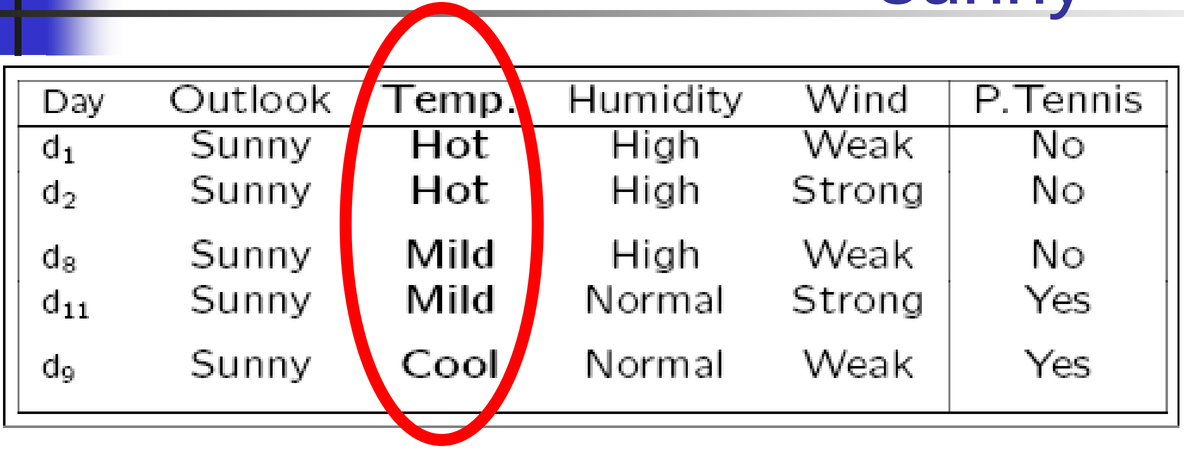
Onto N_{OC} …



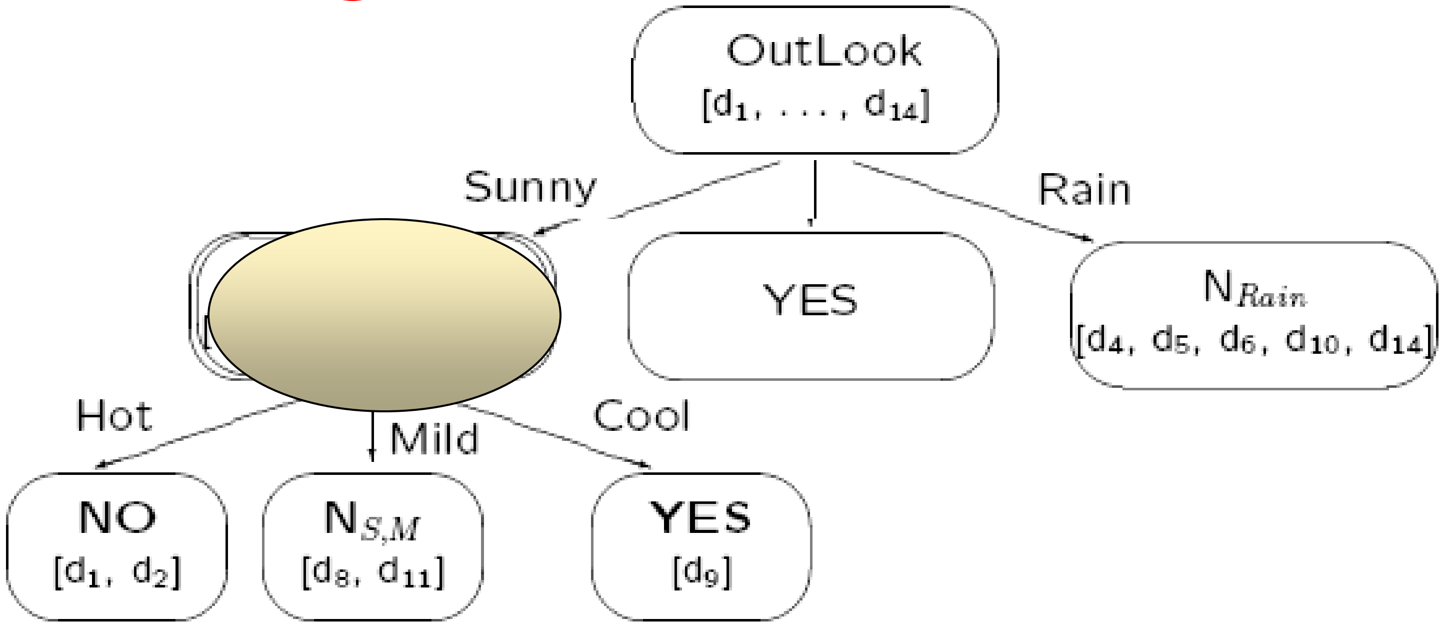
Day	Outlook	Temp.	Humidity	Wind	P.Tennis
d_3	Overcast	Hot	High	Weak	Yes
d_7	Overcast	Cool	Normal	Strong	Yes
d_{12}	Overcast	Mild	High	Strong	Yes
d_{13}	Overcast	Hot	Normal	Weak	Yes


OutLook
[d_1, . . . , d_{14}]
Sunny
OC
Rain
N_{Sunny}
[d_1, d_2, d_8, d_9, d_{11}]
N_{Rain}
[d_4, d_5, d_6, d_{10}, d_{14}]

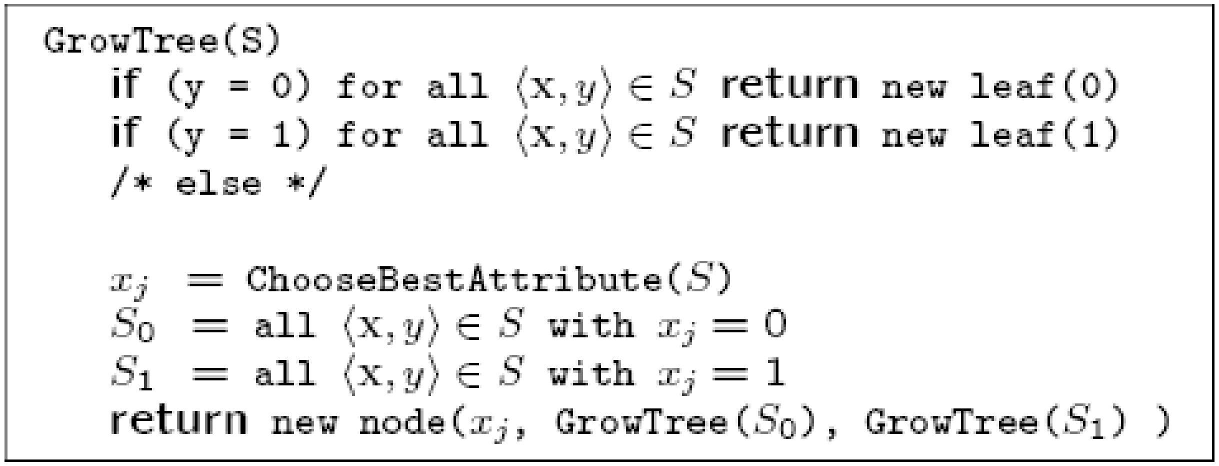
What about N_{Sunny} ?



Day	Outlook	Temp.	Humidity	Wind	P.Tennis
d_1	Sunny	Hot	High	Weak	No
d_2	Sunny	Hot	High	Strong	No
d_8	Sunny	Mild	High	Weak	No
d_{11}	Sunny	Mild	Normal	Strong	Yes
d_9	Sunny	Cool	Normal	Weak	Yes



(Simplified) Algorithm for Learning Decision Tree

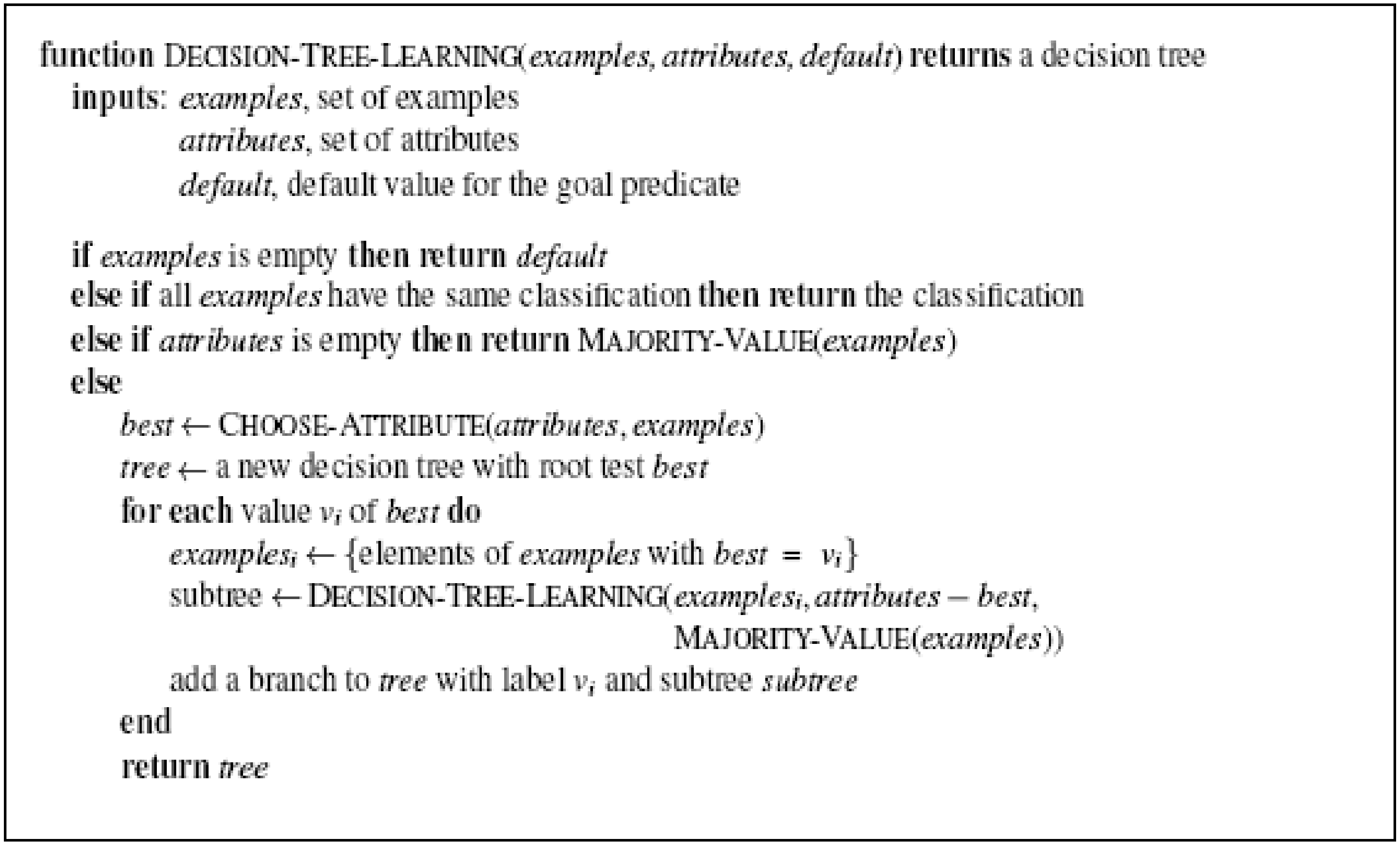


```
GrowTree(S)
    if (y = 0) for all $\langle x, y \rangle \in S$ return new leaf(0)
    if (y = 1) for all $\langle x, y \rangle \in S$ return new leaf(1)
    /* else */

    $x_j$ = ChooseBestAttribute($S$)
    $S_0$ = all $\langle x, y \rangle \in S$ with $x_j = 0$
    $S_1$ = all $\langle x, y \rangle \in S$ with $x_j = 1$
    return new node($x_j$, GrowTree($S_0$), GrowTree($S_1$) )
```

- Many fields independently discovered this learning alg...
- Issues
  - no more attributes
  - > 2 labels
  - continuous values
  - oblique splits
  - pruning

Alg for Learning Decision Trees

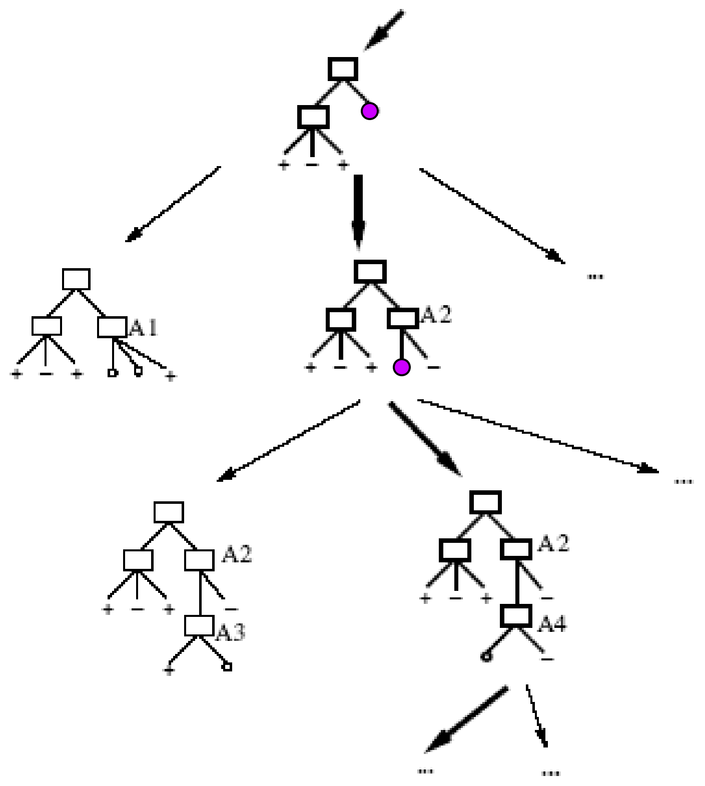


```
function DECISION-TREE-LEARNING(examples, attributes, default) returns a decision tree
  inputs: examples, set of examples
          attributes, set of attributes
          default, default value for the goal predicate

  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MAJORITY-VALUE(examples)
  else
     best ← CHOOSE-ATTRIBUTE(attributes, examples)
     tree ← a new decision tree with root test best
     for each value $v_i$ of best do
        $examples_i$ ← {elements of examples with best = $v_i$}
        subtree ← DECISION-TREE-LEARNING($examples_i$, attributes − best,
                                            MAJORITY-VALUE(examples))
        add a branch to tree with label $v_i$ and subtree subtree
     end
     return tree
```

Search for Good Decision Tree

- Local search
  - expanding one leaf at-a-time
  - no backtracking
- Trivial to find tree that perfectly “fits” training data*
- but… this is NOT necessarily best tree
- Prefer small tree
  - NP-hard to find smallest tree that fits data



* noise-free data

Issues in Design of Decision Tree Learner

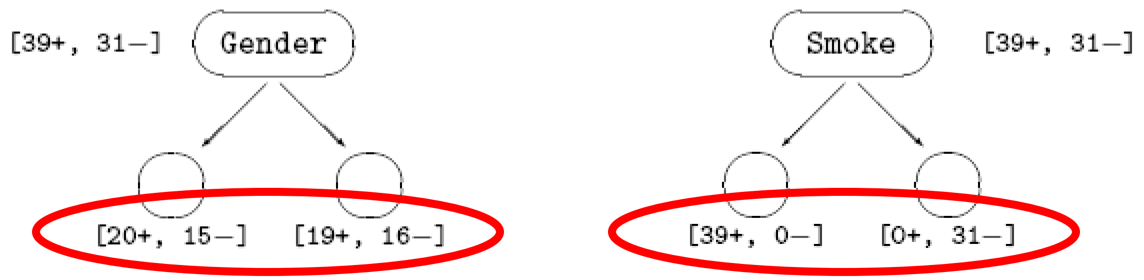
- What attribute to split on?

- Avoid Overfitting
  - When to stop?
  - Should tree by pruned?

- How to evaluate classifier (decision tree) ?
  … learner?

Choosing Best Splitting Test

- How to choose best feature to split?



- After Gender split, still some uncertainty

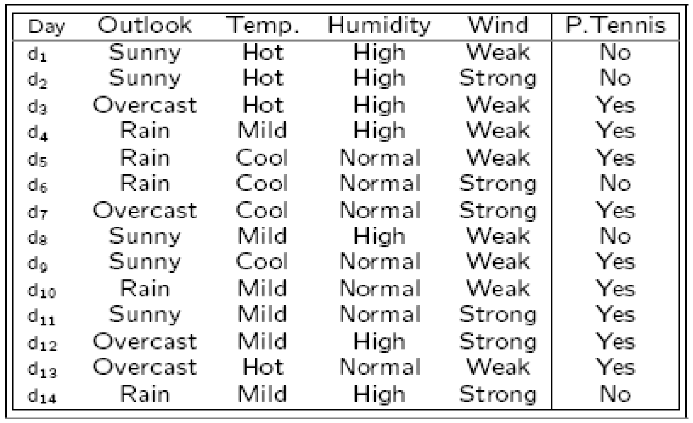
  After Smoke split, no more Uncertainty

  ⇒ NO MORE QUESTIONS!

  (Here, Smoke is a great predictor for Cancer)

- Want a "measure" that prefers

  Smoke over Gender



Day	Outlook	Temp.	Humidity	Wind	P.Tennis
d_1	Sunny	Hot	High	Weak	No
d_2	Sunny	Hot	High	Strong	No
d_3	Overcast	Hot	High	Weak	Yes
d_4	Rain	Mild	High	Weak	Yes
d_5	Rain	Cool	Normal	Weak	Yes
d_6	Rain	Cool	Normal	Strong	No
d_7	Overcast	Cool	Normal	Strong	Yes
d_8	Sunny	Mild	High	Weak	No
d_9	Sunny	Cool	Normal	Weak	Yes
d_{10}	Rain	Mild	Normal	Weak	Yes
d_{11}	Sunny	Mild	Normal	Strong	Yes
d_{12}	Overcast	Mild	High	Strong	Yes
d_{13}	Overcast	Hot	Normal	Weak	Yes
d_{14}	Rain	Mild	High	Strong	No

Statistics ...

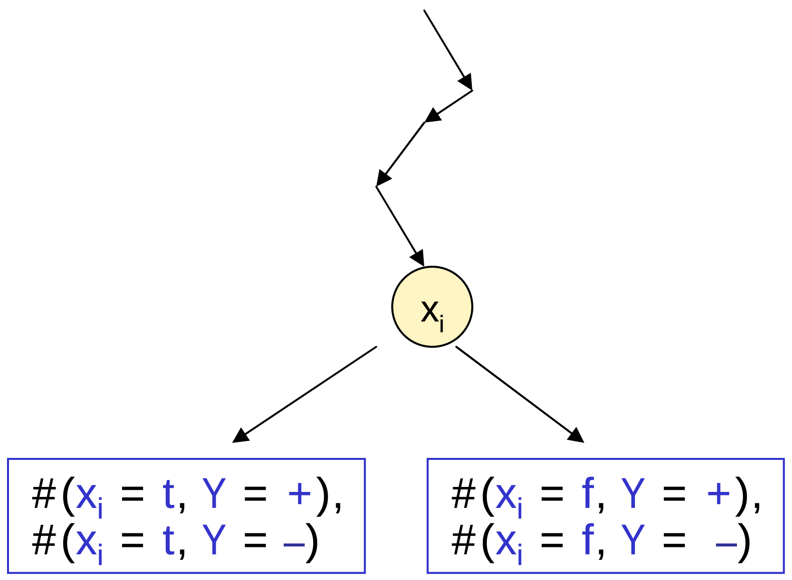
If split on x_i, produce 2 children:

- $\#(x_i = t)$ follow TRUE branch

  $\Rightarrow$ data: [$\#(x_i = t, Y = +)$,
  $\#(x_i = t, Y = -)$]

- $\#(x_i = f)$ follow FALSE branch

  $\Rightarrow$ data: [$\#(x_i = f, Y = +)$,
  $\#(x_i = t, Y = -)$]


x_i
$\#(x_i = t, Y = +)$, $\#(x_i = t, Y = -)$
$\#(x_i = f, Y = +)$, $\#(x_i = f, Y = -)$

Desired Properties

- Score for split M(S, x_i) related to

$$S\begin{pmatrix} \#(x_i = t, Y = +) \\ \#(x_i = t, Y = -) \end{pmatrix} \qquad S\begin{pmatrix} \#(x_i = f, Y = +) \\ \#(x_i = f, Y = -) \end{pmatrix}$$

- Score S(.) should be
  - Score is BEST for [+0, –200]
  - Score is WORST for [+100, –100]
  - Score is "symmetric"

    Same for [+19, –5] and [+5, –19]
  - Deals with any number of values

v_1	7
v_2	19
:	:
v_k	2

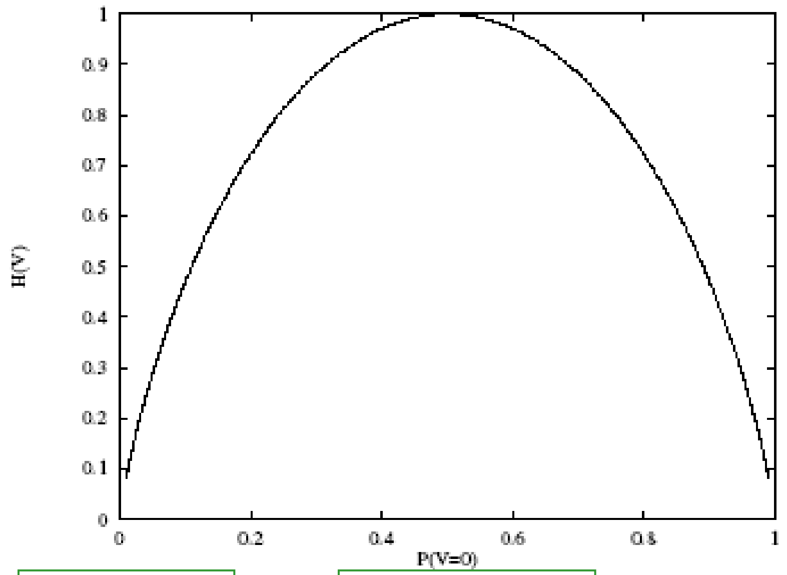
Play 20 Questions

- I'm thinking of integer $\in$ {1, ..., 100 }
- Questions
  - Is it 22?
  - More than 90?
  - More than 50?
- Why?
- Q(r) = # of additional questions wrt set of size r
  - = 22? $1/100 \times Q(1) + 99/100 \times Q(99)$
  - $\geq$ 90? $11/100 \times Q(11) + 89/100 \times Q(89)$
  - $\geq$ 50? $50/100 \times Q(50) + 50/100 \times Q(50)$

Want this to be **small**. . .

Desired Measure: Entropy

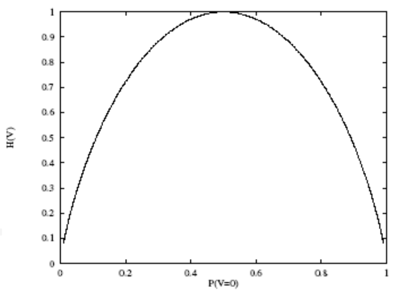
- Entropy of $V = [\ p(V = 1), p(V = 0)\ ]$:
  $H(V) \;=\; -\sum_{v_i} P(\ V = v_i\ ) \log_2 P(\ V = v_i\ )$
  $\equiv$ # of bits needed to obtain full info
  ...average surprise of result of one "trial" of V
- Entropy $\approx$ measure of uncertainty



+200, – 0

+100, – 100

Examples of Entropy



- Fair coin:
  - $H(½, ½) = -½ \log_2(½) - ½ \log_2(½) = 1$ bit
  - ie, need 1 bit to convey the outcome of coin flip)

- Biased coin:

  $H(1/100, 99/100) =$
  $-1/100 \log_2(1/100) - 99/100 \log_2(99/100) = 0.08$ bit

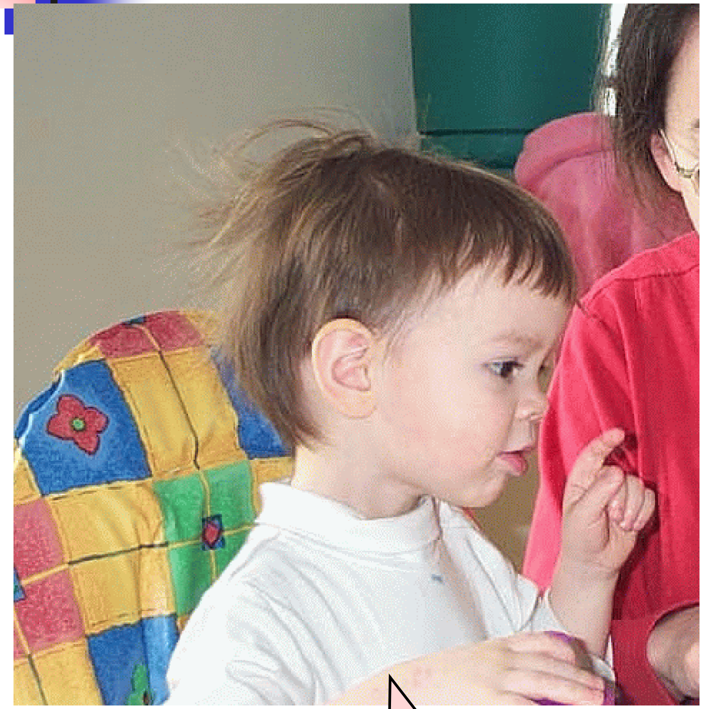
- As P(heads) $\mapsto$ 1, info of actual outcome $\mapsto$ 0

  $H(0, 1) = H(1, 0) = 0$ bits

  ie, no uncertainty left in source

  $(0 \times \log_2(0) = 0)$

Entropy in a Nut-shell



Low Entropy

...the values
(locations of soup)
sampled entirely from within the soup bowl

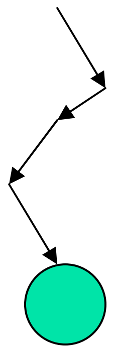
High Entropy

...the values
(locations of soup)
unpredictable... almost uniformly sampled throughout Andrew's dining room

Prefer Low Entropy Leaves

- Use decision tree $h(.)$ to classify (unlabeled) test example x

  ... Follow path down to leaf r
  ... What classification?



- Consider training examples that reached r:
  - If all have same class c $\boxed{+200, -0}$
    $\Rightarrow$ label x as c (ie, entropy is 0)
  - If ½ are $+$; ½ are $-$ $\boxed{+100, -100}$
    $\Rightarrow$ label x as **???** (ie, entropy is 1)
- On reaching leaf r with entropy H_r, uncertainty w/label is H_r

  (ie, need H_r more bits to decide on class)

$\Rightarrow$ prefer leaf with LOW entropy

Entropy of Set of Examples

- Don't have **exact** probabilities...
  ... but training data provides estimates of probabilities:

- Given training set with $\left\{ \begin{array}{l} p \text{ positive} \\ n \text{ negative} \end{array} \right\}$ examples:

$$H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

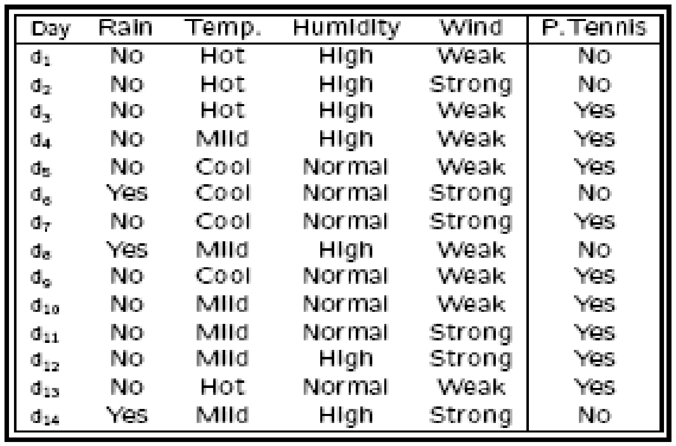
- Eg: wrt 12 instances, *S*:
  $p = n = 6 \quad \Rightarrow \quad H(\tfrac{1}{2}, \tfrac{1}{2}) = 1$ bit
  ... so need 1 bit of info to classify example
  randomly picked from *S*

Remaining Uncertainty

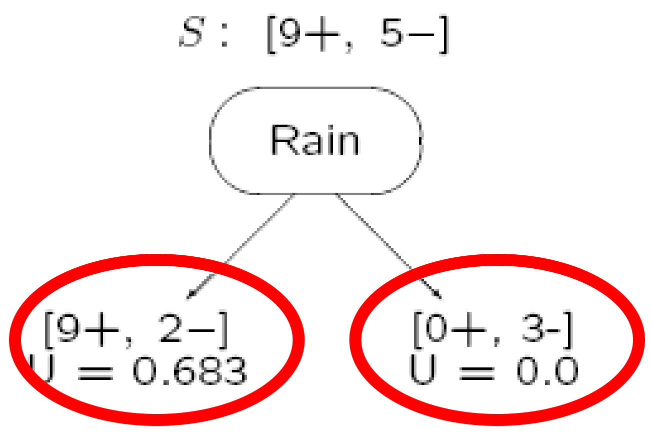
$Uncert(S, A)$ = remaining expected entropy after splitting on A

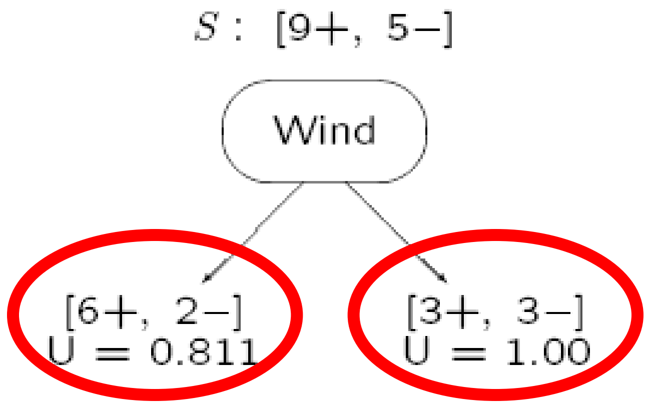
$$\equiv \sum_{v_i \in Values(A)} \frac{|S_{v_i}|}{|S|} \text{Entropy}(S_{v_i})$$

$$\equiv \sum_{i=1}^{v} \frac{p_i^{(A)} + n_i^{(A)}}{p+n} H\left(\frac{p_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}\right)$$



Day	Rain	Temp.	Humidity	Wind	P.Tennis
d_1	No	Hot	High	Weak	No
d_2	No	Hot	High	Strong	No
d_3	No	Hot	High	Weak	Yes
d_4	No	Mild	High	Weak	Yes
d_5	No	Cool	Normal	Weak	Yes
d_6	Yes	Cool	Normal	Strong	No
d_7	No	Cool	Normal	Strong	Yes
d_8	Yes	Mild	High	Weak	No
d_9	No	Cool	Normal	Weak	Yes
d_{10}	No	Mild	Normal	Weak	Yes
d_{11}	No	Mild	Normal	Strong	Yes
d_{12}	No	Mild	High	Strong	Yes
d_{13}	No	Hot	Normal	Weak	Yes
d_{14}	Yes	Mild	High	Strong	No

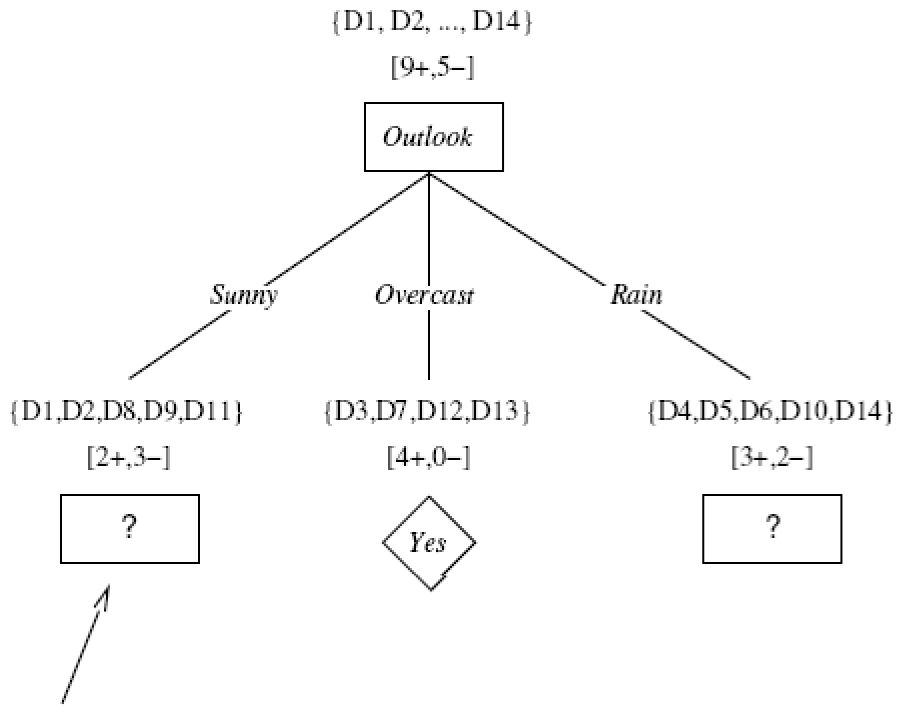




$Uncert(S, Rain)$
$= \frac{11}{14} 0.683 + \frac{3}{14} 0.0$
$= 0.536$

$Uncert(S, Wind)$
$= \frac{8}{14} 0.811 + \frac{6}{14} 1.00$
$= 0.892$

… as tree is built …



Day	Outlook	Temp.	Humidity	Wind	P.Tennis
d_1	Sunny	Hot	High	Weak	No
d_2	Sunny	Hot	High	Strong	No
d_3	Overcast	Hot	High	Weak	Yes
d_4	Rain	Mild	High	Weak	Yes
d_5	Rain	Cool	Normal	Weak	Yes
d_6	Rain	Cool	Normal	Strong	No
d_7	Overcast	Cool	Normal	Strong	Yes
d_8	Sunny	Mild	High	Weak	No
d_9	Sunny	Cool	Normal	Weak	Yes
d_{10}	Rain	Mild	Normal	Weak	Yes
d_{11}	Sunny	Mild	Normal	Strong	Yes
d_{12}	Overcast	Mild	High	Strong	Yes
d_{13}	Overcast	Hot	Normal	Weak	Yes
d_{14}	Rain	Mild	High	Strong	No

Which attribute should be tested here?

$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Rem(S_{sunny}, Humidity) = (3/5)\ 0.0 - (2/5)\ 0.0 = .000$

$Rem(S_{sunny}, Temperature) = (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .400$

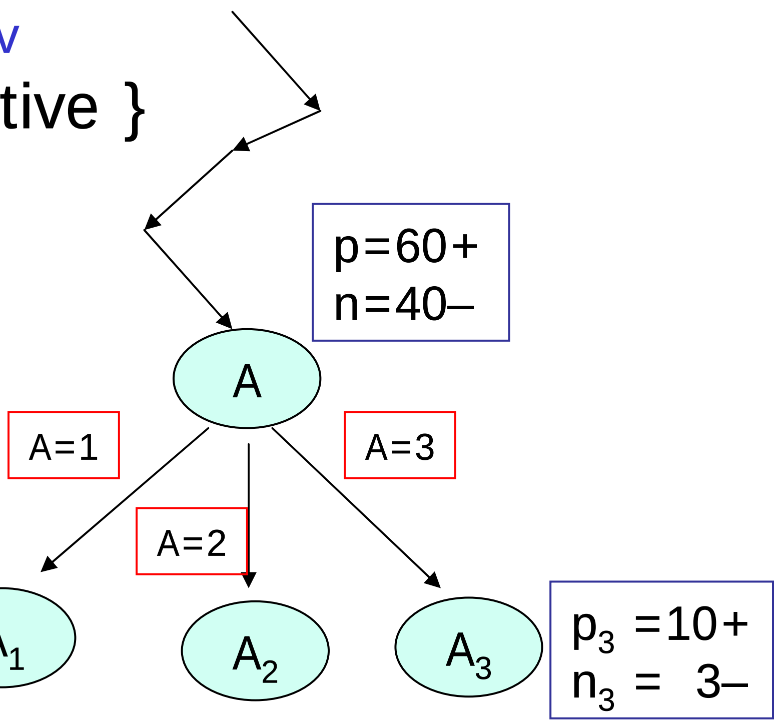
$Rem(S_{sunny}, Wind) = (2/5)\ 1.0 - (3/5)\ .918 = .951$

Entropy wrt Feature

- Assume [p,n] reach node
- Feature A splits into $A_1, \ldots, A_v$
  - A_i has { $p_i^{(A)}$ positive, $n_i^{(A)}$ negative }
- Entropy of each is ...

$$H\left(\frac{p_i^{(A)}}{p_i^{(A)}+n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)}+n_i^{(A)}}\right)$$

So for A_2:
H(28/40, 12/40)



$$Uncert(A) = \sum_{i=1}^{v} \frac{p_i^{(A)}+n_i^{(A)}}{p+n} H\left(\frac{p_i^{(A)}}{p_i^{(A)}+n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)}+n_i^{(A)}}\right)$$

28+
12–

Minimize Remaining Uncertainty

- Greedy: Split on attribute that leaves **least entropy** wrt class
  ... over training examples that reach there
- Assume A divides training set E into E_1, ..., E_v
- E_i has { $p_i^{(A)}$ positive, $n_i^{(A)}$ negative } examples
- Entropy of each E_i is $H\left(\frac{p_i^{(A)}}{p_i^{(A)}+n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)}+n_i^{(A)}}\right)$
- Uncert(A) = expected information content
  $\Rightarrow$ weighted contribution of each E_i

$$\boxed{Uncert(A) = \sum_{i=1}^{v} \frac{p_i^{(A)}+n_i^{(A)}}{p+n} H\left(\frac{p_i^{(A)}}{p_i^{(A)}+n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)}+n_i^{(A)}}\right)}$$

- Often worded as ***Information Gain***

$$Gain(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - Uncert(A)$$

Notes on Decision Tree Learner

- Hypothesis space is complete!

  ⇒ contains target function...

- No back tracking
  - Local minima...

- Statistically-based search choices
  - Robust to noisy data...

- Inductive bias: ≈ “prefer shortest tree”

Inductive Bias in C4.5

- **H** = DecisionTreeClassifiers

  $\approx$ power set of instances X

  $\Rightarrow$ Unbiased?
- Not really…
  - Preference for short trees,

    [trees w/ high info gain attributes near root]
  - Here: Bias is **preference** for some hypotheses, rather than restriction of hypothesis space H
  - **Occam's razor:**

    Prefer shortest hypothesis that fits data

Occam's Razor

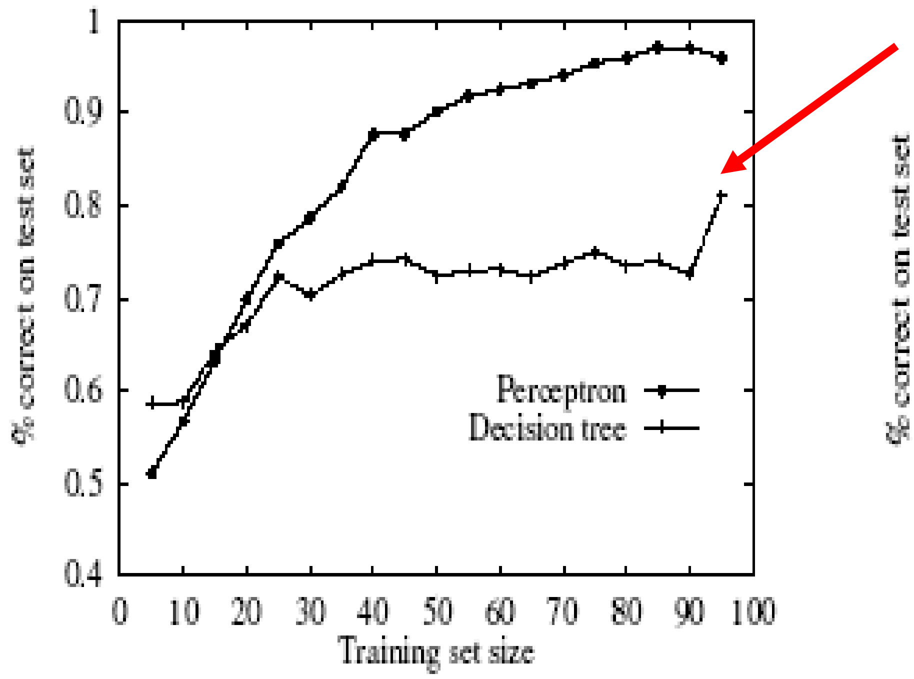
- Q: Why prefer short hypotheses?
- Argument in favor:
  - Fewer short hyps. than long hyps.

  $\Rightarrow$ a short hyp that fits data *unlikely* to be coincidence

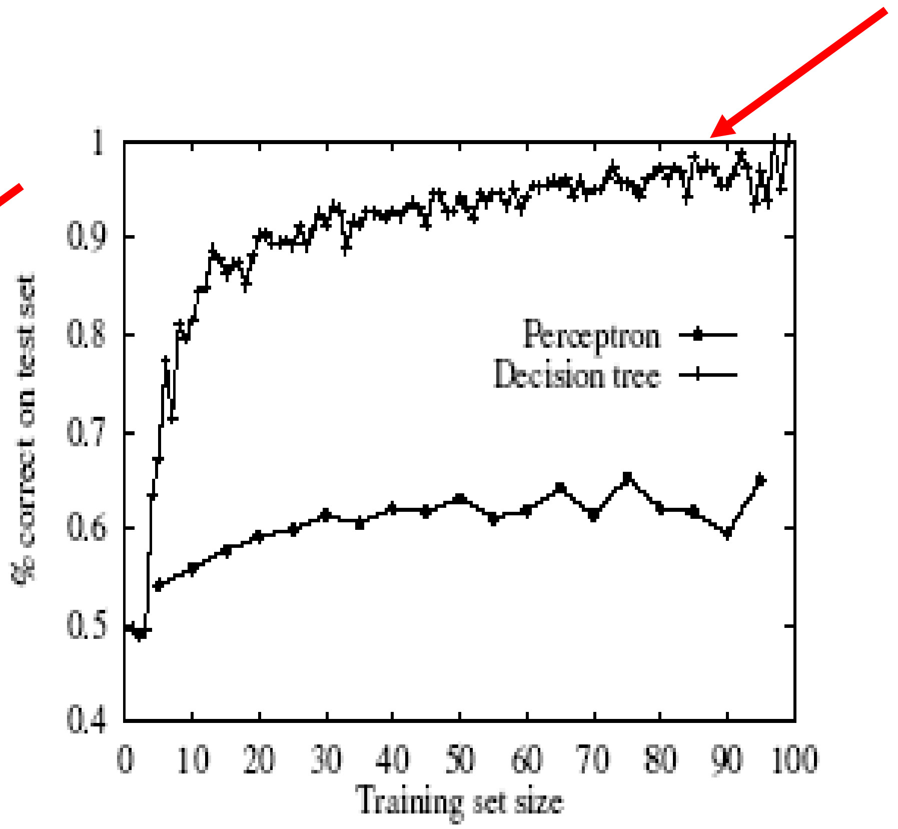
  $\Rightarrow$ a long hyp that fits data *might be* coincidence
- Argument opposed:
  - $\exists$ many ways to define small sets of hyps

    Eg, all trees with prime number of nodes
      whose attributes all begin with “Z"
  - What's so special about small sets based on size of hypothesis??

Perceptron vs Decision Tree



Majority Function (11 inputs)



WillWait Predicate

Learning Decision Trees

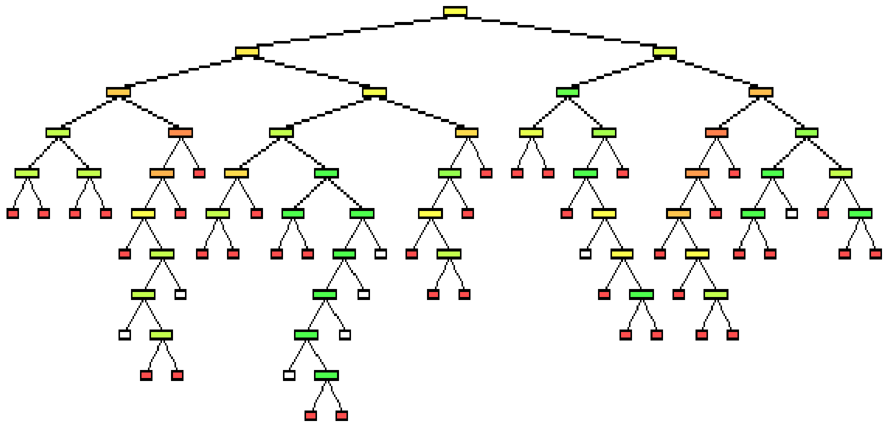
- Defn: Decision Tree
- Algorithm for Learning Decision Trees
- Overfitting
  - Def'n
  - MDL, χ^2
  - PostPruning
- Topics:

Example of Overfitting

- 25% have butterfly-itis
- ½ of patients have $F_1 = 1$
  - Eg: "odd birthday"
- ½ of patients have $F_2 = 1$
  - Eg: "even SSN"
- ... for 10 features
- Decision Tree results
  - over 1000 patients (using these silly features) ...

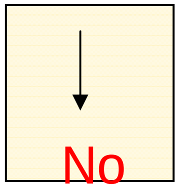
Decision Tree Results

- Standard decision tree learner:



- Error Rate:
  - Train data: 0%
  - New data: 37%

- Optimal decision tree:



- Error Rate:
  - Train data: 25%
  - New data: 25%

Overfitting

- Often "meaningless regularity" in data due to coincidences in the noise

  $\Rightarrow$ bad generalization behavior

  "Overfitting"
- Consider error in hypothesis h over …
  - training data S: $err_S(h)$
  - entire distribution D of data: $err_{D,f}(h)$
- Hypothesis $h \in H$ **overfits** training data if

  $\exists$ alternative hypothesis $h' \in H$ s.t.

  $err_S(h) < err_S(h')$

  but

  $err_{D,f}(h) > err_{D,f}(h')$

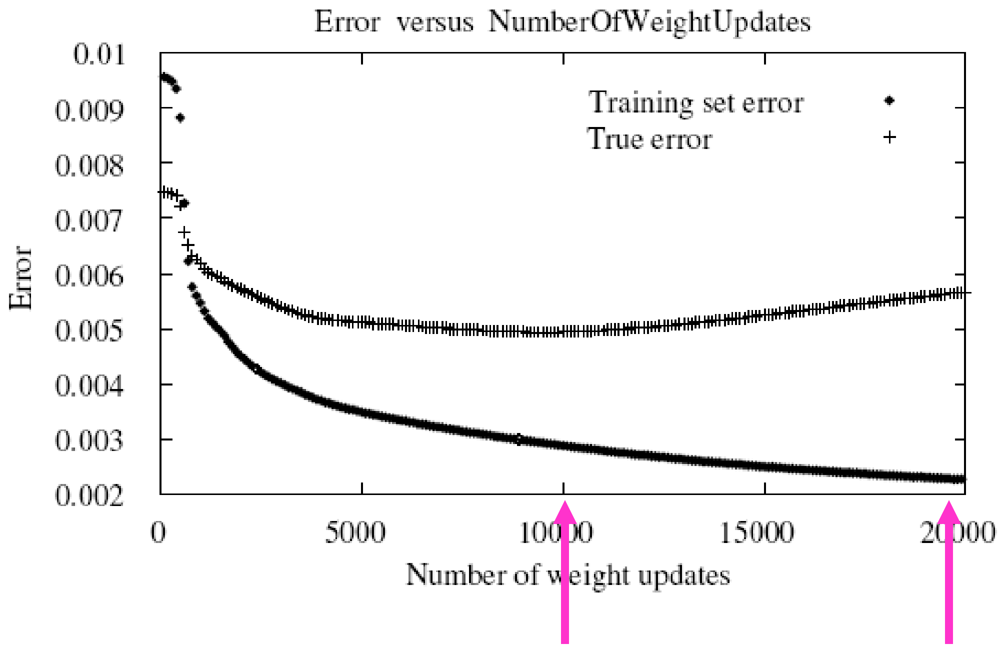
Fit-to-Data ≠ Generalization

- h_k = hyp after k updates

  $err_S(h_{20000}) < err_S(h_{10000})$

  but

  $err_{D,f}(h_{20000}) > err_{D,f}(h_{10000})$



- "Overfitting"

  Best "fit-to-data" will often find meaningless regularity in data (coincidences in the noise)

  ⇒ bad generalization behavior

Example of Overfitting

- Spse 10 binary attributes (uniform),

  but class is random: $\begin{cases} \text{N w/prob } p = 0.75 \\ \text{Y w/prob } 1-p = 0.25 \end{cases}$

- C4.5 builds nonsensical tree w/ 119 nodes!
  ⇒ Should be SINGLE NODE!
- Error rate (hold-out): 35%
  ⇒ Should be 25% (just say "No")
- Why? Tree assigns leaf "N" w/prob 1–p, "Y" w/prob p
  - Tree sends instances to arbitrary leaves
  - Mistake if
    - Y-instance reaches N-leaf: p x (1–p)
    - N-instance reaches Y-leaf: (1–p) x p
  - Total prob of mistake = 2 x p x (1–p) = 0.375

- Overfitting happens for EVERY learner … not just DecTree !!

How to Avoid Overfitting (Decision Trees)

- When to act
  - Use more stringent STOPPING criterion while growing tree

    . . . only allow statistically significant splits …
  - Grow full tree, then post-prune
- To evaluate tree, measure performance over …
  - training data
  - separate validation data set
- How to represent classifier?
  - as Decision Tree
  - as Rule Set

Avoid Overfitting #1

(StopEARLY, Training-Data, DecTree)

- Add more stringent STOPPING criterion while growing tree
  - At leaf n_r (w/ instances S_r)
    spse optimal proposed split is based on attribute A

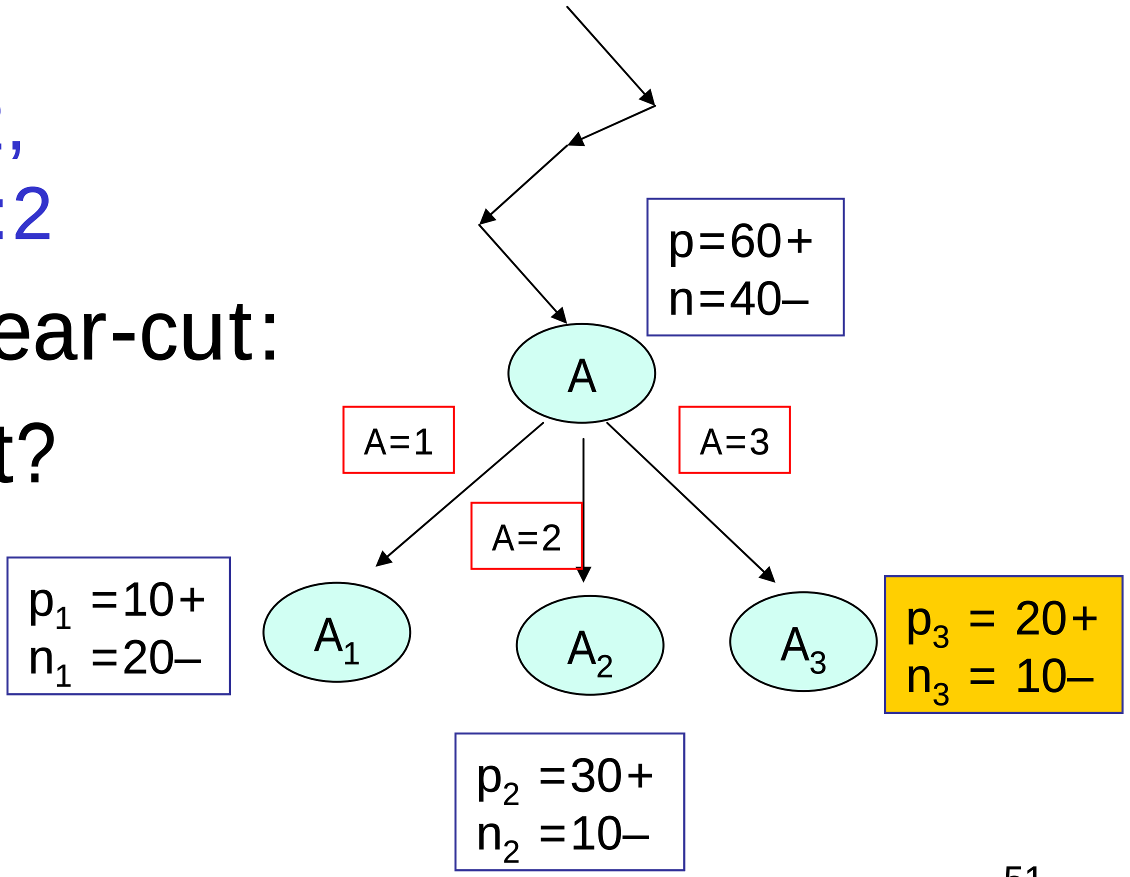
A. Use χ^2 **test**, on data S_r
  - Apply statistical test to compare
    - T_0: leaf at r (majority label) vs
    - T_A: split using A
  - Is error of T_A statistically better than T_0?

B. **MDL**: minimize

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

Test for Significance

- Spse A is irrelevant
  - $[p_i, n_i] \propto [p, n]$
  - So if $[p,n] = 3:2$, then $[p_i, n_i] = 3:2$
- Not always so clear-cut:
- Is this significant?
- Or this??



χ^2 Test for Significance

- Null hypothesis H_0:

  Attribute A is irrelevant in context of r

  Ie, distr of class labels at node $n_r \equiv$ distr after splitting on A
- Observe some difference between these distr's.

  What is prob (under H_0) of observing this difference, given m = $|S_r|$ iid samples?

- Defn: $\begin{Bmatrix} p \\ n \end{Bmatrix}$ of S_r are $\begin{Bmatrix} \text{positive} \\ \text{negative} \end{Bmatrix}$

  After splitting on A, get *k* subsets

  wrt *A = i* : p_i positives, n_i negatives
- If H_0 (A irrelevant), would have
  - $p_i^{\sim} = p \times (p_i+n_i)/p+n$ positives
  - $n_i^{\sim} = n \times (p_i+n_i)/p+n$ negatives

χ^2 Test – con't

- If H_0 (A irrelevant), would have

$$\widehat{p}_i = p \times \frac{p_i+n_i}{p+n} \text{ positives}$$
$$\widehat{n}_i = n \times \frac{p_i+n_i}{p+n} \text{ negatives}$$

- Measure of deviation:

$$D = \sum_{i=1}^{k} \frac{(p_i - \widehat{p}_i)^2}{\widehat{p}_i} + \frac{(n_i - \widehat{n}_i)^2}{\widehat{n}_i}$$

$\sum$ (exp – obs)^2 / exp

Under H_0, $D \sim \chi^2_{(k-1)(2-1)}$

(k = size of A's domain; 2 = # classes)

Q? Is D value within tolerences?

Don't add iff D < T$_{\alpha,d}$

$T_{\alpha,d}$ is threshold:

α = prob of making mistake

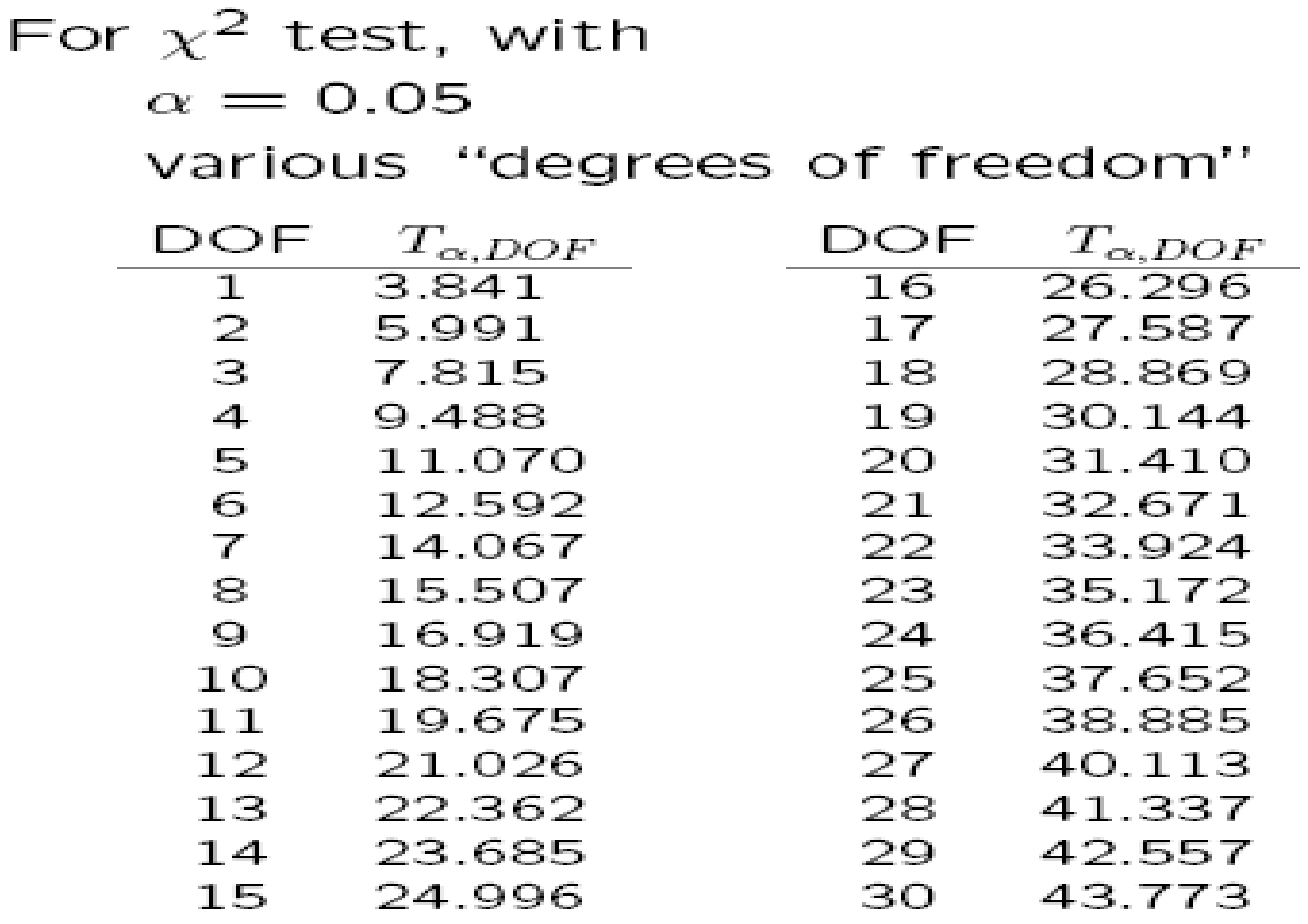
(rejecting null hyp, when A is irrelevant)

d = degree of freedom

- Obvious extension when $>$ 2 classes

Results: Empirically, not reliable. . .

χ^2 Table



For χ^2 test, with

$\alpha = 0.05$

various "degrees of freedom"

DOF	$T_{\alpha,DOF}$	DOF	$T_{\alpha,DOF}$
1	3.841	16	26.296
2	5.991	17	27.587
3	7.815	18	28.869
4	9.488	19	30.144
5	11.070	20	31.410
6	12.592	21	32.671
7	14.067	22	33.924
8	15.507	23	35.172
9	16.919	24	36.415
10	18.307	25	37.652
11	19.675	26	38.885
12	21.026	27	40.113
13	22.362	28	41.337
14	23.685	29	42.557
15	24.996	30	43.773

Minimum Description Length

A wants to transmit to B classification function $c()$

- simplified to:
  - A and B agree on instances $\langle x_1, \ldots, x_M \rangle$
  - What should A send, to allow B to determine M bits:

    $\langle c(x_1), \ldots, c(x_M) \rangle$
- Option#1: A can send M “bits”
- Option#2: A sends “perfect” decision tree d

  s.t. $c(x_i) = d(x_i)$ for each x_i
- Option#3: A sends "imperfect" decision tree d'

  + set of indices of K exceptions $B = \{ x_{i1}, \ldots, x_{iK} \}$

$$c(x_i) = \begin{cases} \neg d(x_i) & \text{if } x_i \in B \\ d(x_i) & \text{otherwise} \end{cases}$$

- So… Increase tree-size

  IF (significant) reduction in #exceptions

Avoid overfitting#2: PostPruning

- Grow tree, to “purity”, then PRUNE it back!

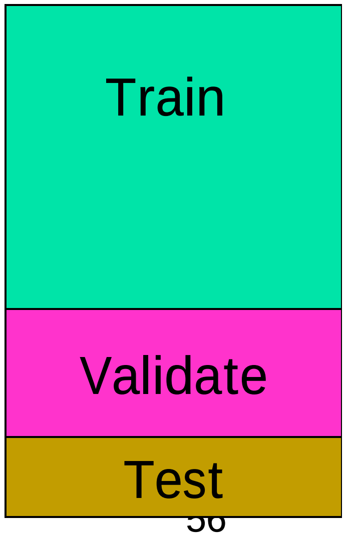
```
Build “complete” decision tree $h$, from train
For each penultimate node: $n_i$
  Let $h_i$ be tree formed by “collapsing” subtree under $n_i$, into single node
  If $h_i$ better than $h$
    Reset $h \leftarrow h_i$, . . .
```

- How to decide if h_i better than h?

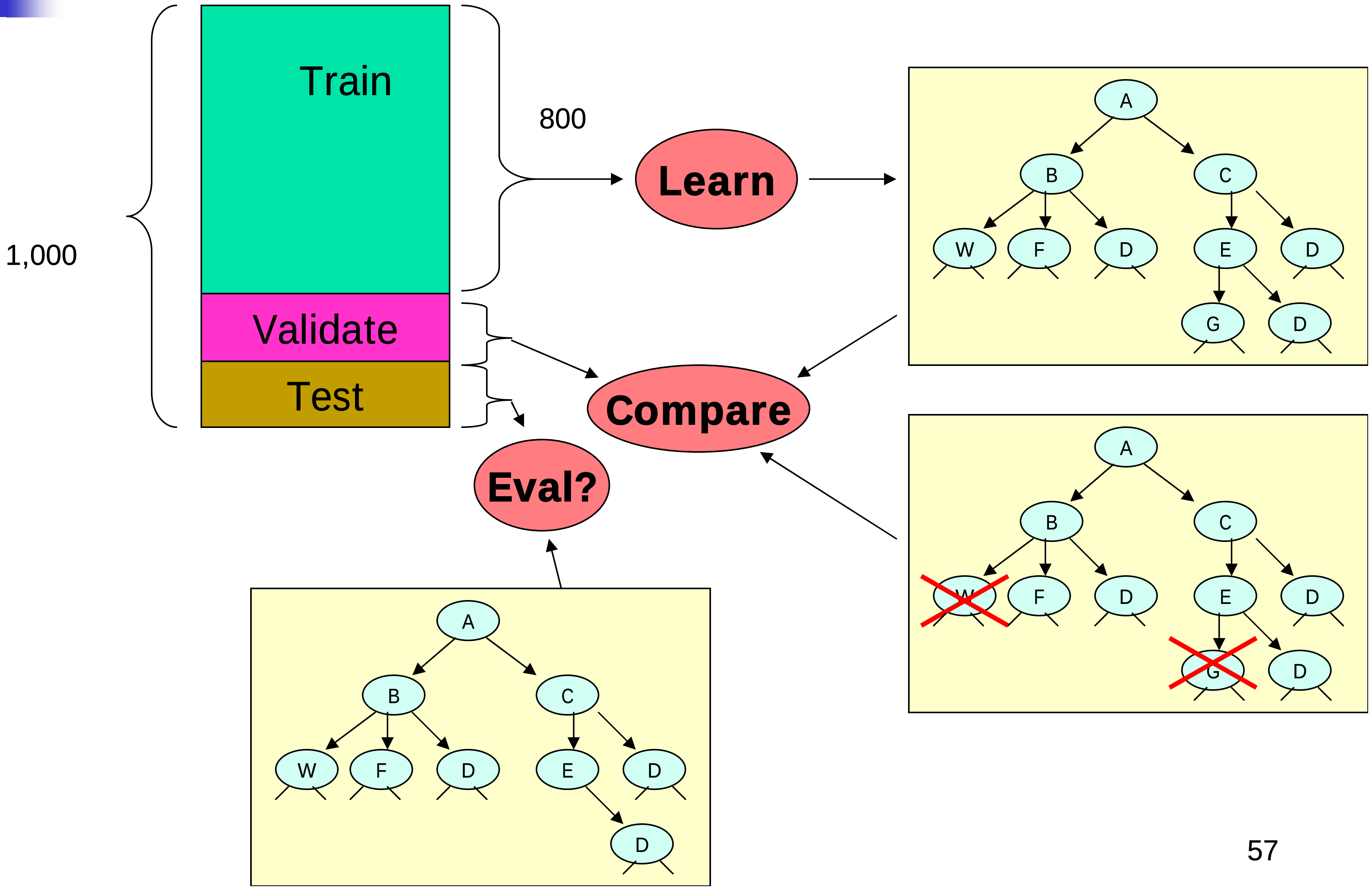
1. Test on Hold-Out data?
   - 3 sets: training, *VALIDATION*, testing

   Problematic if small total # of samples
2. Pessimistic Pruning

   . . . re-use training samples . . .

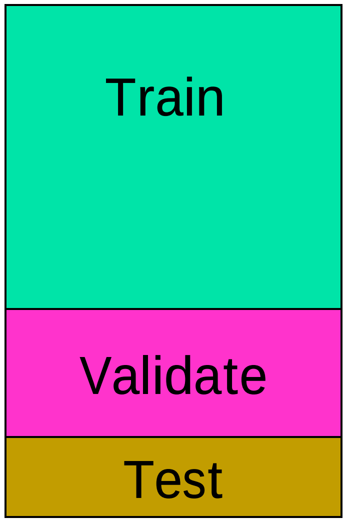


Using Validation Set



Avoid Overfitting#2.1 “Reduced-Error Pruning"

- Split data into *training* and *validation* set



Alg: Do until further pruning is harmful:

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

- Produces small version of accurate subtree
- What if data is limited?

Avoid Overfitting#2.2 "Pessimistic Pruning"

- Assume *N* training samples reach leaf r; ... which makes E mistakes
  so (resubstitution) error is E/N

- For confidence level (eg, 1-sided $\alpha = 0.25$),
  can estimate upper bound on # of errors:
  Number Of Errors = $N \times [E/N \pm EB_\alpha(N,E)]$

- Let $U_\alpha(N,E) = E/N + EB_\alpha(N, E)$

  $EB_\alpha(E;N)$ based on binomial distribution

  ~ Normal distribution: $z_\alpha \times \sqrt{p(1-p)/N}$
  $p \approx E/N$

- Laplacian correction... to avoid "divide by 0" problems:

  Use $p = (E+1)/(N+2)$ not E/N

- For $\alpha = 0.25$, use $z_{0.25} = 1.53$
  (recall 1-sided)

Pessimistic Pruning (example)

$U_\alpha(N,E) = E/N + EB_\alpha(N, E)$

- Eg, spse A has 3 values: $\{ v_1\ v_2\ v_3 \}$
- If split on A, get
  - A= v_1 – return Y (6 cases, 0 errors)
  - A= v_2 – return Y (9 cases, 0 errors)
  - A= v_3 – return N (1 cases, 0 errors)

  So 0 errors if split on A

W/prob α,

$$\text{For } A = \left\{ \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix} \right\}, \text{ error rate is under } \left\{ \begin{matrix} U_\alpha(6,0) \\ U_\alpha(9,0) \\ U_\alpha(1,0) \end{matrix} \right\}$$

- For α=0.25:

$$\begin{aligned} \#\text{errors} &\le 6 \times U_{0.25}(6,0) + 9 \times U_{0.25}(9,0) + 1 \times U_{0.25}(1,0) \\ &= 6 \times 0.206 + 9 \times 0.143 + 1 \times 0.75 = 3.273 \end{aligned}$$

- If replace A-subtree w/ simple “Y”-leaf: (16 cases, 1 error)

$$\#\text{errors} \le 16 \times U_{0.25}(16,1) = 16 \times 0.1827 = 2.923$$

- As $2.923 < 3.273$, prune A-subtree to single “Y” leaf

  Then recur – going up to higher node

Pessimistic Pruning: Notes

- Results: Pruned trees tend to be
  - more accurate
  - smaller
  - easier to understand than original tree
- Notes:
  - Goal: to remove irrelevant attributes
  - Seems inefficient to grow subtree, only to remove it
  - This is VERY ad hoc, and WRONG statistically
    but works SO WELL in practice it seems essential
  - Resubstitution error goes UP; but generalization error, down...
  - Could replace n_i with single node,
    or with most-frequently used branch

Avoid Overfitting #3 Using Rule Post-Pruning

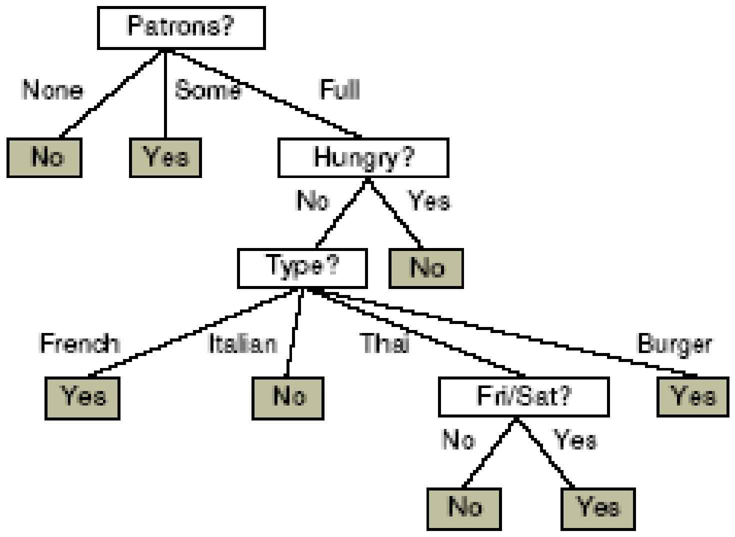
1. Grow decision tree.
   Fit data as well as possible.
   Allow overfitting.
2. Convert tree to equivalent set of rules:
   - One rule for each path from root to leaf.
3. Prune each rule independently of others.
   - ie, delete preconditions that improve its accuracy
4. Sort final rules into desired sequence for use depending on accuracy.

---

5. Use ordered sequence for classification.

Converting Trees to Rules

- Every decision tree corresponds to set of rules:
  - IF (Patrons = None)
    THEN WillWait = No
  - IF (Patrons = Full)
      & (Hungry = No)
      &(Type = French)
    THEN WillWait = Yes
  - ...



- Why? (Small) RuleSet MORE expressive
  small DecTree ≈ small RuleSet
  (DecTree is subclass of ORTHOGONAL DNF)

Learning Decision Trees

- Def'n: Decision Trees
- Algorithm for Learning Decision Trees
- Overfitting
- Topics:
  - k-ary attribute values
  - Real attribute values
  - Other splitting criteria
  - Attribute Cost
  - Missing Values
  - ...

Attributes with Many Values

- Problem: *Gain* prefers attribute with many values.
  Entropy ≈ ln(k) . . .
  Eg, imagine using
  Date = Jun 3 1996
  Name = Russ
- One approach: use *GainRatio* instead

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

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

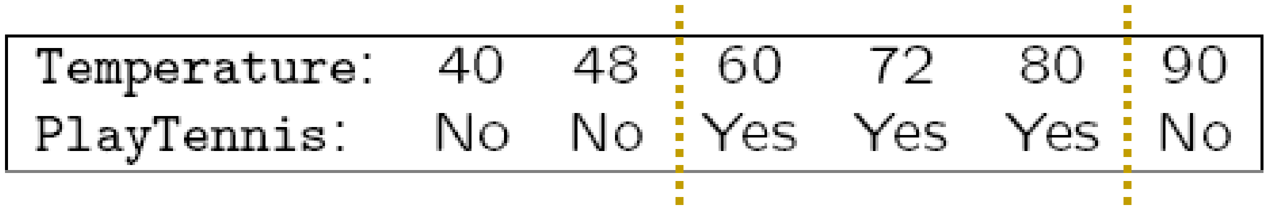
where S_i is subset of S for which A has value v_i

- Issues:
  - Construct a multiway split?
  - Test one value versus all of the others?
  - Group values into two disjoint subsets?

Continuous Valued Attributes

Create a discrete attribute to test continuous

- *Temperature* = 82.5
- (Temperature > 72.3) ∈ { t, f }



Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

- Note: need only consider splits between "class boundaries"

  Eg, between 48 / 60; 80 / 90

Finding Split for Real-Valued Features

- Best threshold θ_j for attribute j
  - Sort training instance by $x_{i,j}$
  - For each $\theta \in \Re$, where

		$< \theta$	$\geq \theta$
labeled	A	$n_{A,\ell}(\theta)$	$n_{A,r}(\theta)$
	B	$n_{B,\ell}(\theta)$	$n_{B,r}(\theta)$
	(either)	$n_\ell(\theta)$	$n_r(\theta)$

y_i	A	A	B	A	B	B	A	B	B
$x_{i,j}$	0.2	0.4	0.7	1.1	1.3	1.7	1.9	2.4	2.9

$n_{A,\ell} = 3$	$n_{A,r} = 1$
$n_{B,\ell} = 1$	$n_{B,r} = 4$

Mutual information $= 0.2294$

  - Sequentially set θ to value of $\frac{x_{i,j}+x_{i,j+1}}{2}$
    Compute mutual information

$$\frac{n_\ell(\theta)}{n_\ell(\theta)+n_r(\theta)}\left[\frac{n_{A,\ell}(\theta)}{n_{A,\ell}(\theta)+n_{B,\ell}(\theta)} \ln \frac{n_{A,\ell}(\theta)}{n_{A,\ell}(\theta)+n_{B,\ell}(\theta)} + \frac{n_{B,\ell}(\theta)}{n_{A,\ell}(\theta)+n_{B,\ell}(\theta)} \ln \frac{n_{B,\ell}(\theta)}{n_{A,\ell}(\theta)+n_{B,\ell}(\theta)}\right]$$
$$+$$
$$\frac{n_r(\theta)}{n_\ell(\theta)+n_r(\theta)}\left[\frac{n_{A,r}(\theta)}{n_{A,r}(\theta)+n_{B,r}(\theta)} \ln \frac{n_{A,r}(\theta)}{n_{A,r}(\theta)+n_{B,r}(\theta)} + \frac{n_{B,r}(\theta)}{n_{A,r}(\theta)+n_{B,r}(\theta)} \ln \frac{n_{B,r}(\theta)}{n_{A,r}(\theta)+n_{B,r}(\theta)}\right]$$

  - Just need to consider j s.t. $y_j \neq y_{j+1}$!

Desired Properties

Repeat!

- Score for split M(D, x_i) related to

$$S\left(\begin{array}{c}\#(x_i=t,Y=+)\\ \#(x_i=t,Y=-)\end{array}\right) \qquad S\left(\begin{array}{c}\#(x_i=f,Y=+)\\ \#(x_i=f,Y=-)\end{array}\right)$$

- Score S(.) should be
  - Score is BEST for [+0, –200]
  - Score is WORST for [+100, – 100]
  - Score is "symmetric"

    Same for [+19, – 5] and [+5, –19]
  - Deals with any number of values

v_1	7
v_2	19
:	:
v_k	2

Other Splitting Criteria

- Why use *Gain* as splitting criterion?

Want: Large "use me" value if split is

$\langle$ 85, 0, 0, ..., 0 $\rangle$

Small "avoid me" value if split is

$\langle$ 5, 5, 5, ..., 5 $\rangle$

- True of *Gain*, *GainRatio*... also for. . .
- Statistical tests: χ^2

For each attr A, compute deviation:

$$D(A) = \sum_{i=1}^{k} \frac{(p_i^{(A)} - \hat{p}_i^{(A)})^2}{\hat{p}_i^{(A)}} + \frac{(n_i^{(A)} - \hat{n}_i^{(A)})^2}{\hat{n}_i^{(A)}}$$

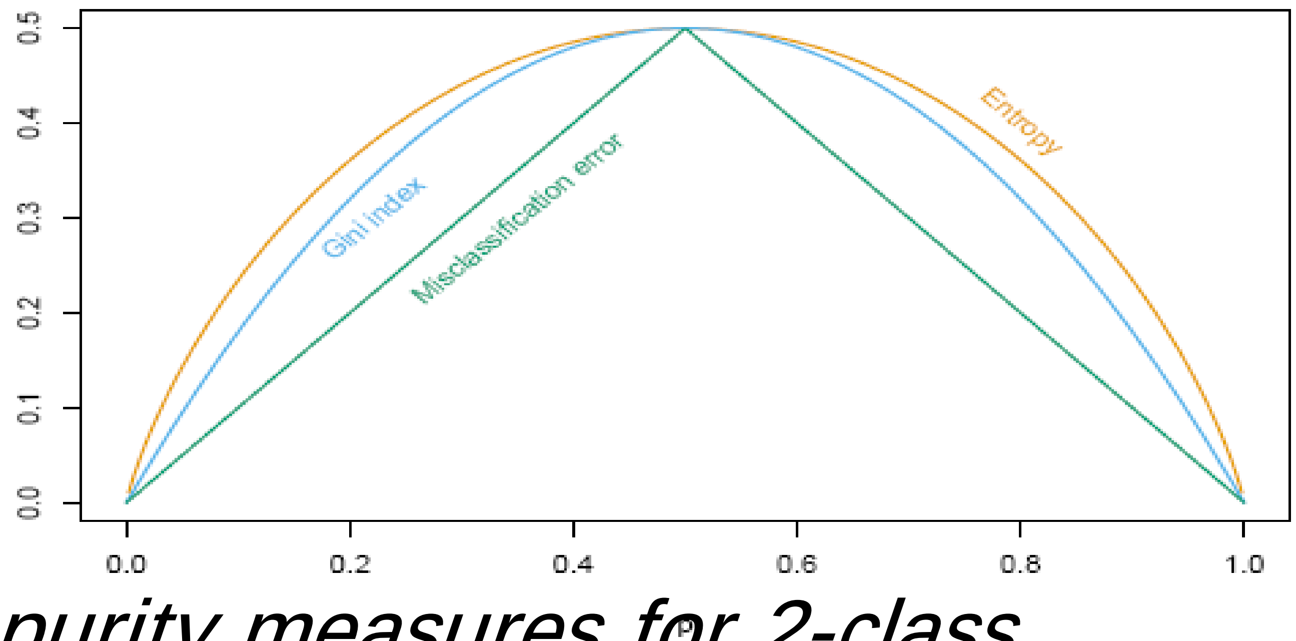
- GINI index: $\text{GINI}(A) = \sum_i \sum_{j \neq i} p_i \, p_j = 1 - \sum_i p_i^2$

- Others: “Marshall Correction”

“G” statistic

Probabilities (rather than statistic)

Node Impurity Measures



- *Node impurity measures for 2-class classification*
  - *function of the proportion p in class* 2.
  - *Scaled coss-entropy has been scaled to pass through* (0.5, 0.5).

HTF 2009

Example of χ^2

- Attributes T_1, T_2 class c

Data:

$T_1 =$	$c =$ Yes	No	
Y	0	3	3
N	5	2	7
	5	5	10

$T_2 =$	$c =$ Yes	No	
a	3	2	5
b	1	2	3
c	1	1	2
	5	5	10

χ^2: For T_1:

$$\frac{(0-1.5)^2}{1.5} + \frac{(3-1.5)^2}{1.5} + \frac{(2-3.5)^2}{3.5} + \frac{(5-3.5)^2}{3.5} = 4.29$$

For T_2:

$$\frac{(3.0-2.5)^2}{2.5} + \frac{(2-2.5)^2}{2.5} + \frac{(1-1.5)^2}{1.5} + \frac{(2-1.5)^2}{1.5} + \frac{(1-1)^2}{1} + \frac{(1-1)^2}{1}$$
$$= 0.533$$

- As 4.29 (T_1) > 0.533 (T_2), ... use T_1
  (less likely to be irrelevant)

Example of GINI

- Attributes T_1, T_2 class c

Data:

$T_1 =$	$c =$ Yes	No	
Y	0	3	3
N	5	2	7
	5	5	10

$T_2 =$	$c =$ Yes	No	
a	3	2	5
b	1	2	3
c	1	1	2
	5	5	10

Scores:

$$GINI(T_1) = \left[1-\left(\tfrac{5}{10}\right)^2-\left(\tfrac{5}{10}\right)^2\right] - \left[\tfrac{3}{10}\left(1-\left(\tfrac{0}{3}\right)^2-\left(\tfrac{3}{3}\right)^2\right)+\tfrac{7}{10}\left(1-\left(\tfrac{5}{7}\right)^2-\left(\tfrac{2}{7}\right)^2\right)\right]$$

$$= \tfrac{1}{10}\left(\left(\tfrac{0^2}{3}+\tfrac{3^2}{3}+\tfrac{5^2}{7}+\tfrac{2^2}{7}\right)-\left(\tfrac{5^2}{10}+\tfrac{5^2}{10}\right)\right)$$

$$= \boxed{0.21429}$$

$$GINI(T_2) = \tfrac{1}{10}\left(\left(\tfrac{3^2}{5}+\tfrac{2^2}{5}+\tfrac{1^2}{3}+\tfrac{2^2}{3}+\tfrac{1^2}{2}+\tfrac{1^2}{2}\right)-\left(\tfrac{5^2}{10}+\tfrac{5^2}{10}\right)\right)$$

$$= \boxed{0.026667}$$

- As $GINI(T_1) > GINI(T_2)$, split on T_1
- As "$\left(\tfrac{5^2}{10}+\tfrac{5^2}{10}\right)$" is subtracted from both, can just use first term... and ignore $\tfrac{1}{10}$ multiplier

Cost-Sensitive Classification ... Learning

- So far, only considered **ACCURACY**

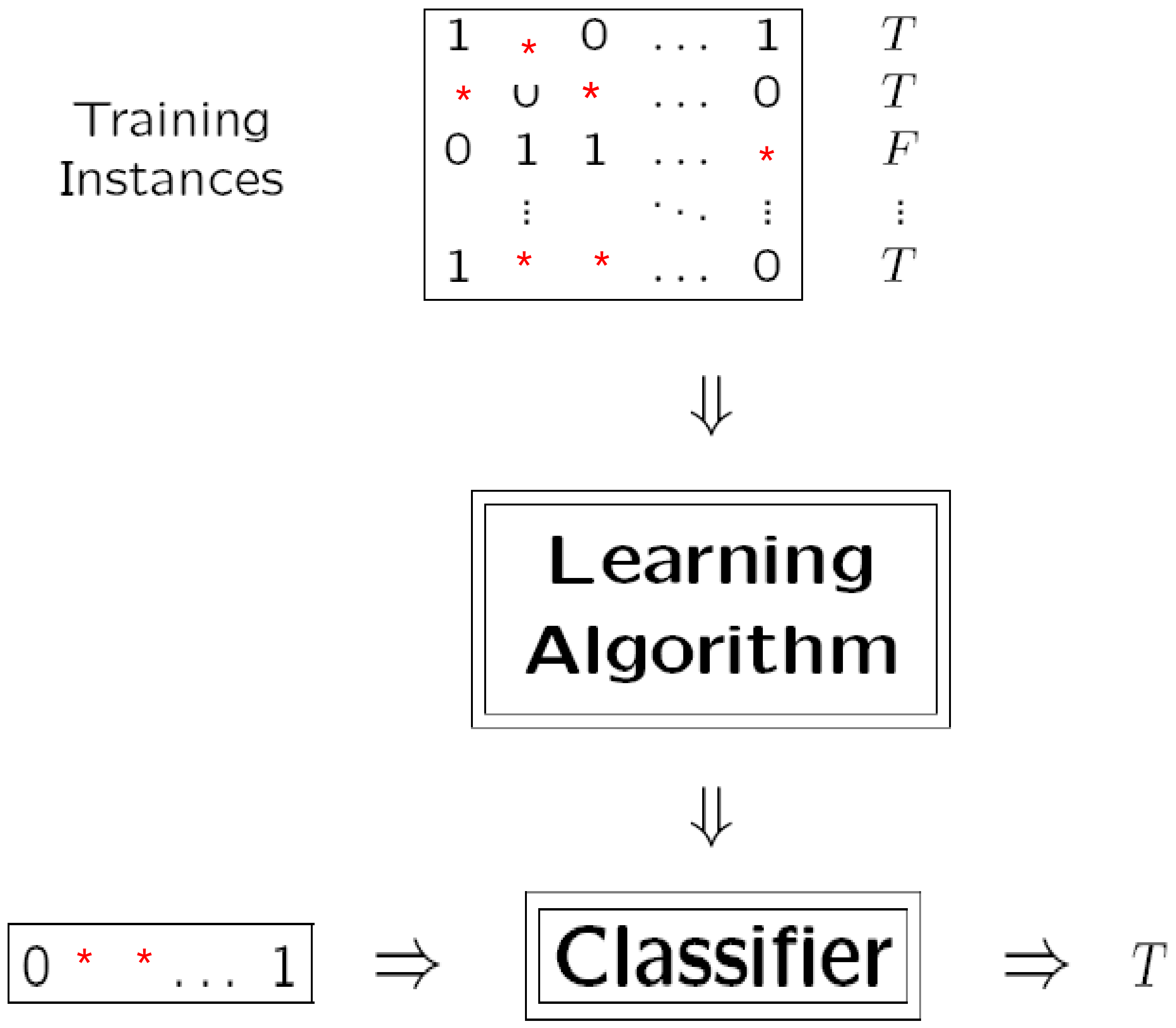
  In gen'l, may want to consider **COST** as well
  - medical diagnosis: BloodTest costs \$150
  - robotics: Width_from_1ft costs 23 sec
- Learn a consistent tree with low expected cost?

  . . . perhaps replace *InfoGain(S,A)* by
  - $\text{Gain}^2(S,A)$ / $\text{Cost}(A)$ [Tan/Schlimmer'90]
  - $[\ 2^{\text{Gain}(S,A)} - 1] \ / \ [\text{Cost}(A)+1]^w$

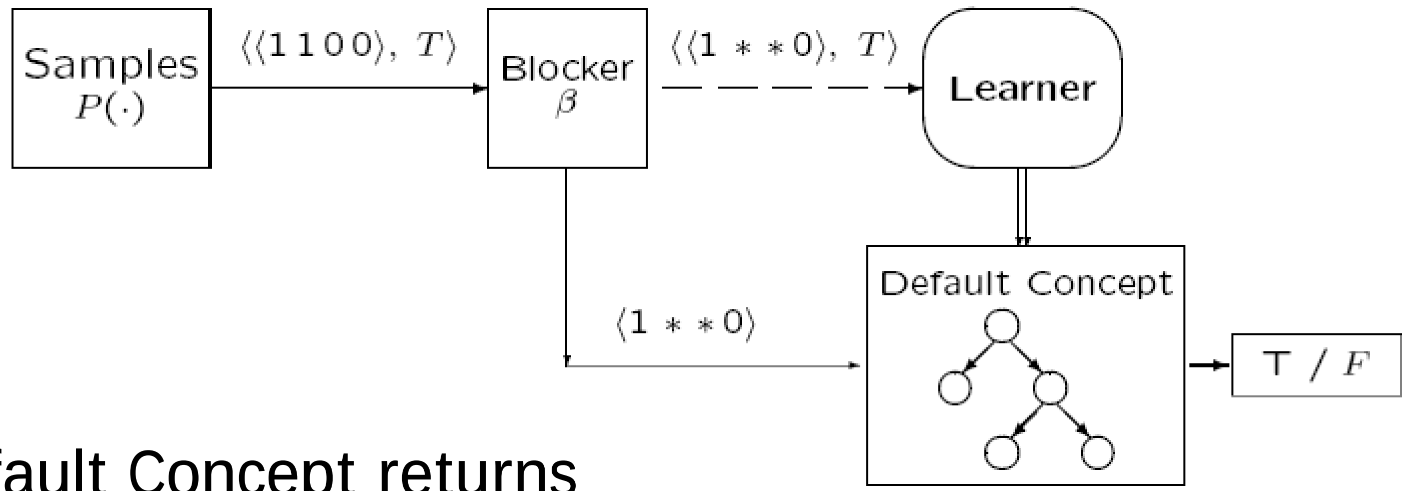
    where $w \in [0, 1]$ determines importance of cost [Nunez'88]
- General utility (arb rep'n)

  $E[\ \sum_i \text{cost}(A_i) + \text{Misclass penalty}\ ]$ [Greiner/Grove/Roth'96]

Dealing with Missing Information



Formal Model



- Default Concept returns
  Categorical Label {T, F } even when given partial instance
  - . . . even ⟨ *,*, ..., * ⟩ !
- Blocker β : { 0, 1}^{n} $\rightarrow_{\text{Stoch}}$ {0, 1, * }
- N.b., β
  - does NOT map 0 to 1
  - does NOT change class label
  - may reveal different attributes on different instances
    (on same instance, different times)

Unknown Attribute Values

- Q: What if some examples are **incomplete**
  . . . missing values of some attributes?

When learning:

A1: Throw out all incomplete examples?
  ... May throw out too many. . .

A2: Fill in most common value ("imputation")
  - May miss correlations with other values
  - If impute wrt attributes: may require high order statistics

A3: Follow all paths, w/ appropriate weights
  - Huge computational cost if missing MANY values

When classifying

- Similar ideas . . .
- ISSUE: Why are values missing?
  - Transmission Noise
  - "Bald men wear hats"
  - "You don't care"

  See [Schuurmans/Greiner'94]

Handling Missing Values: Proportional Distribution

- Associate weight w_i with example $\langle x_i, y_i \rangle$
  At root, each example has weight 1.0
- Modify mutual information computations:
  use weights instead of counts
- When considering test on attribute j,
  only consider examples that include x_{ij}
- When splitting examples on attribute j:
  - p_L = prob. non-missing example sent left
    p_R = prob. non-missing example sent right
  - For each example $\langle x_i, y_i \rangle$ missing attribute j:
    send it to both children;
    - to left w/ $w_i := w_i \times p_L$
    - to right w/ $w_i := w_i \times p_R$
  - To classify example missing attribute j:
    - Send it down left subtree; $P(\tilde{y}_L \mid x)$ = resulting prediction
    - Send it down left subtree; $P(\tilde{y}_R \mid x)$ = resulting prediction
    - Return $p_L \times P(\tilde{y}_L \mid x) + p_R \times P(\tilde{y}_R \mid x)$

Handling Missing Values: Surrogate Splits

- **Choose** attribute j and splitting threshold θ_j using all examples that include j

$$u_i = \begin{cases} \text{L if } \langle x_i, y_i \rangle \text{ sent to LEFT subtree} \\ \text{R if } \langle x_i, y_i \rangle \text{ sent to RIGHT subtree} \end{cases}$$

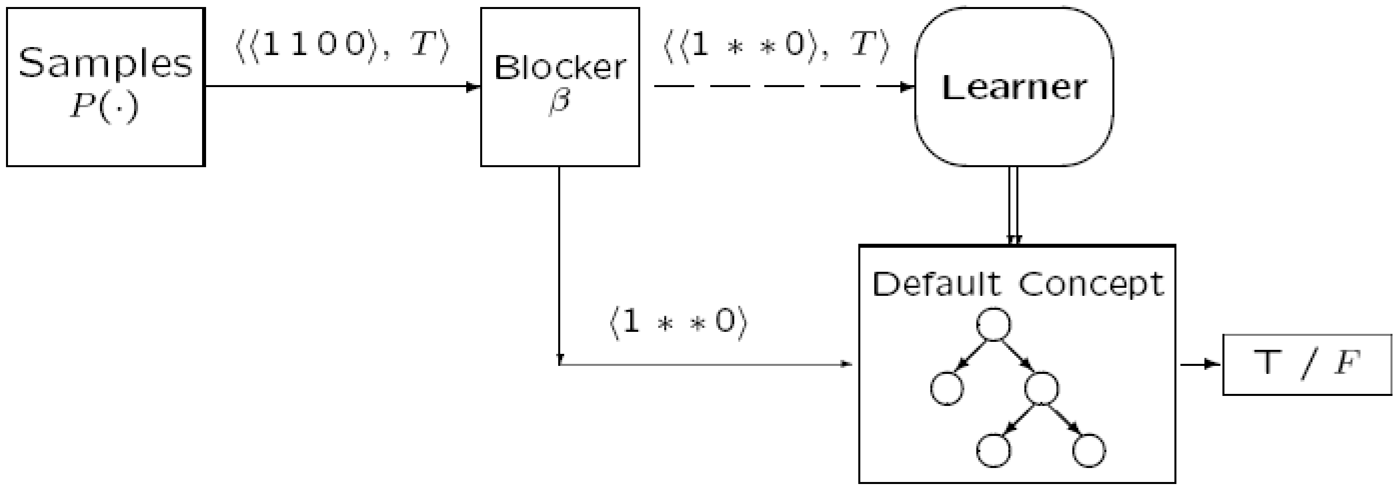
- For each other attribute q, find splitting threshold θ_q that best predicts u_i

  Sort q by predictive power

  Called "surrogate splits"
- Sort via "surrogate splits"

  To handle $\langle x_i, y_i \rangle$ where $x_{ij} = *$:
  - go thru surrogate splits q until finding one NOT missing
  - Use q, θ_q to decide which child gets x_i

Questions



1. How to represent default concept?
2. When is best default concept learnable?
3. If so, how many samples are required?
4. Is it better to learn from ...
   - Complete Samples, or
   - Incomplete Samples?

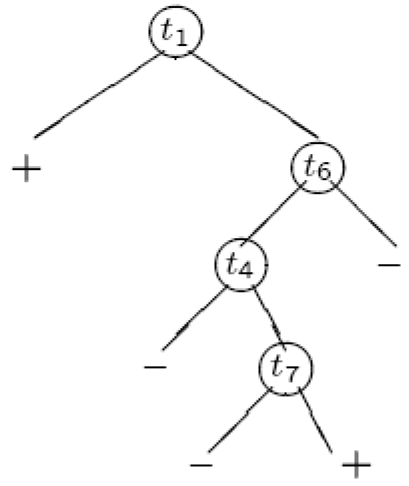
Learning Task

Input:

P#1:	⟨	+				⟩	+
P#2:	⟨	−	−	+	+	⟩	−
P#3:	⟨	−		−		⟩	−
P#4:	⟨	−	−	+	−	⟩	+
P#5:	⟨	−	+	+		⟩	−

⋮

Output: (small) decision tree consistent with data



Learning Decision Trees … with “You Don't Care” Omissions

- No known algorithm for PAC-learning gen'l Decision Trees

  given all attribute values
- … but Decision Trees are TRIVIAL to learn,

  if superfluous values are omitted:

```
Algorithm GrowDT
    Collect "enough" labeled (blocked) instances
    Let **root** = never-blocked instance xi
    Split instances by $x_i = 1$ vs $x_i = 0$,
      and recur (until purity)
```

Motivation

- Most learning systems work best when
  - few attribute values are missing
  - missing values randomly distributed
- but. . . [Porter,Bareiss,Holte'90]
  - many datasets missing > ½ values!
  - not randomly missing but . . .

    "[missing] when they are known to be irrelevant for classication or redundant with features already present in the case description"

  ⇒ Our Situation!!
- Why Learn? . . . when experts
  - not available, or
  - unable to articulate classification process

Decision Tree Evaluation

Criterion	LMS	Logistic	LDA	DecTree
Mixed data	No	No	No	Yes
Missing values	No	No	Yes	Yes
Outliers	No	Yes	No	Yes
Monotone transformations	No	No	No	Yes
Scalability	Yes	Yes	Yes	Yes
Irrelevant inputs	No	No	No	Somewhat
Linear combinations	Yes	Yes	Yes	No
Interpretable	Yes	Yes	Yes	Yes
Predictive power	Yes	Yes	Yes	No

Comments on Decision Trees

- "Decision Stumps" (1-level DT)
  seem to work surprisingly well
- Efficient algorithms for learning optimal “depth-k decision trees”
  ... even if continuous variables
- Oblique Decision Trees

  Not just "$x_3 > 5$", but "$x_4 + x_8 > 91$"
- Use of prior knowledge
  - Incremental Learners ("Theory Revision")
  - "Relevance" info
- Software Systems:
  - C5.0 (from ID3, C4.5) [Quinlan'93]
  - CART
  - ...
- Applications:
  - Gasoil $\approx$ 2500 rules
  - designing gas-oil separation for offshore oil platforms
  - Learning to fly Cessna plane

What we haven't discussed...

- Real-valued *outputs* – Regression Trees
- Bayesian Decision Trees
  - a different approach to preventing overfitting
- How to choose MaxPchance automatically
- Boosting: a simple way to improve accuracy

What you should know

- Information gain:
  - What is it?   Why use it?
- Recursive algorithm for building an unpruned decision tree
- Why pruning can reduce test set error
- How to exploit real-valued inputs
- Computational complexity
  - straightforward, cheap
- Coping with Missing Data
- Alternatives to Information Gain for splitting nodes

For more information

- Two nice books
  - *Classification and Regression Trees*. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Wadsworth, Belmont, CA, 1984.
  - *C4.5: Programs for Machine Learning* (Morgan Kaufmann Series in Machine Learning) by J. Ross Quinlan

Both started ≈ 1983, in Bay Area...done independently -- CS vs Stat

- Dozens of nice papers, including
  - *Learning Classification Trees*, Wray Buntine, Statistics and Computation (1992), Vol 2, pages 63-73
  - *On the Boosting Ability of Top-Down Decision Tree Learning Algorithms*. Kearns and Mansour,, STOC: ACM Symposium on Theory of Computing, 1996“
- Dozens of software implementations available on the web for free and commercially for prices ranging between $50 - $300,000

Conclusions

- Classification: predict a categorical output from categorical and/or real inputs
- Decision trees are the single most popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- Need to avoid overfitting