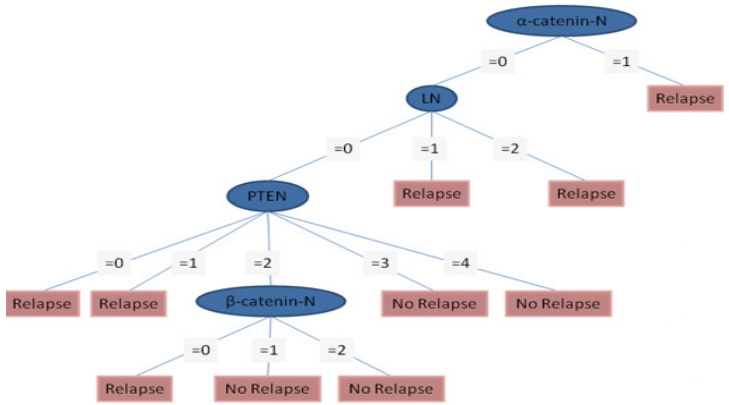
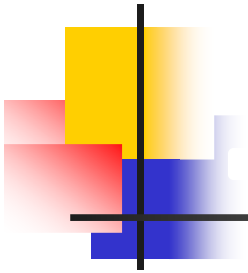
Cmput 466 / 551



Decision Trees

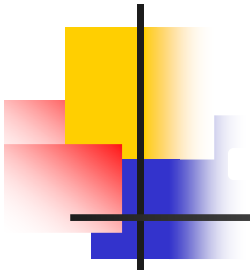
Covering: HTF Ch 9 (esp 9.2) + Induction of Decision Trees (Quinlan)

R Greiner
Dept of Computing Science
University of Alberta



Learning Decision Trees

- Def'n: Decision Trees
- Algorithm for Learning Decision Trees
- Overfitting
  - Def'n, χ^2, MDL, PostPruning
- Topics:
  - k-ary attribute values
  - Real attribute values
  - Other splitting criteria
  - Attribute Cost
  - Missing Values
  - ...



Lessons

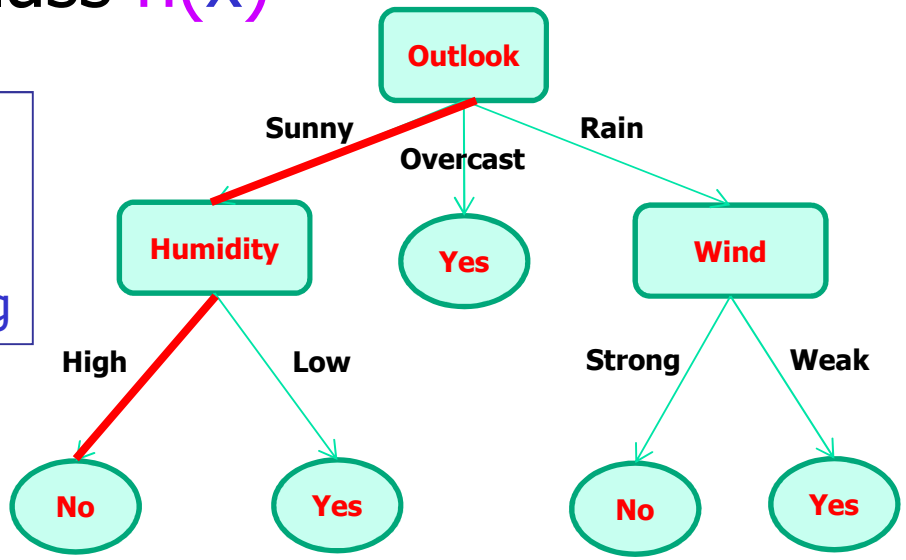
- Understanding Decision Tree
  - Def'n, Alg, Variants, …
- Overview other topics:
  - Entropy
  - (Another) Example of Overfitting
  - χ^2 test
  - Minimal Description Language
  - Missing data …
  - Importance of understanding data

DecisionTree Hypothesis Space

- Each internal nodes labeled with some feature x_j
- Each arc (from x_j) labeled with results of test x_j
- Each leaf node specifies class $h(x)$

- Instance:

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



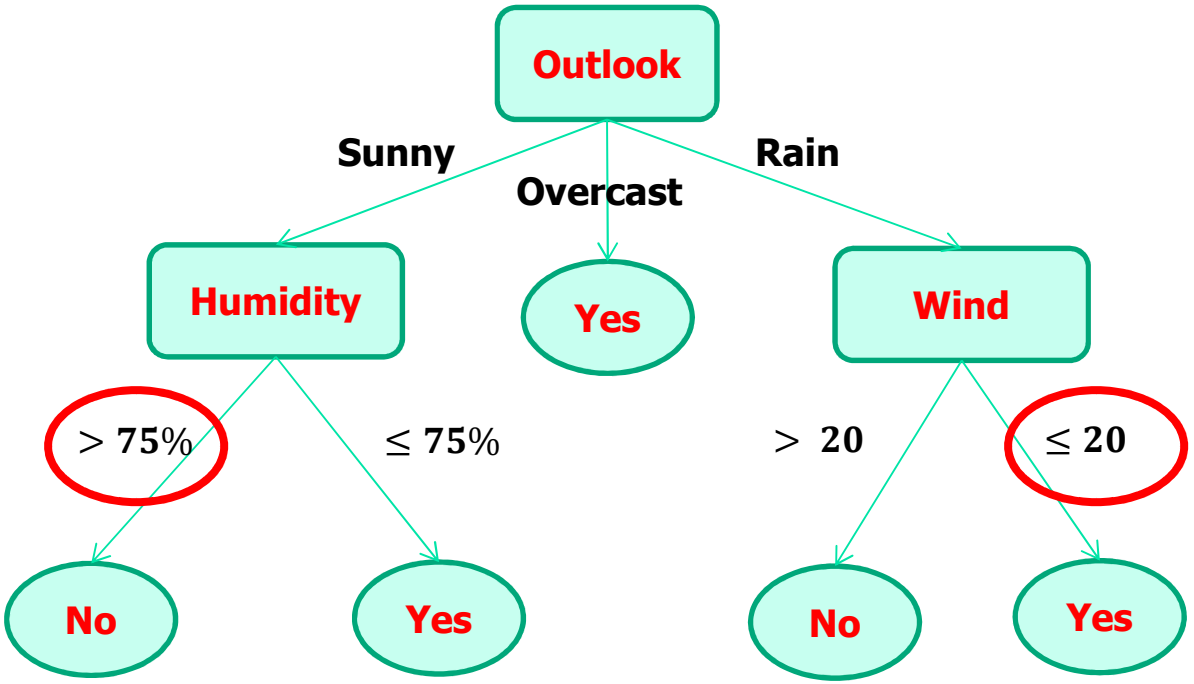
classified as "No"
  - (Temperature, Wind: irrelevant)

- Easy to use in Classification
  - Just need to answer short series of questions…

Continuous Features

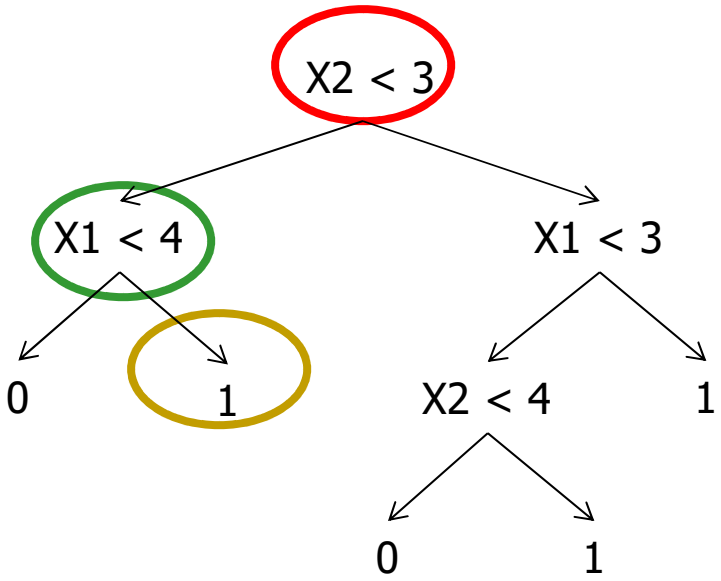
- If feature is continuous:

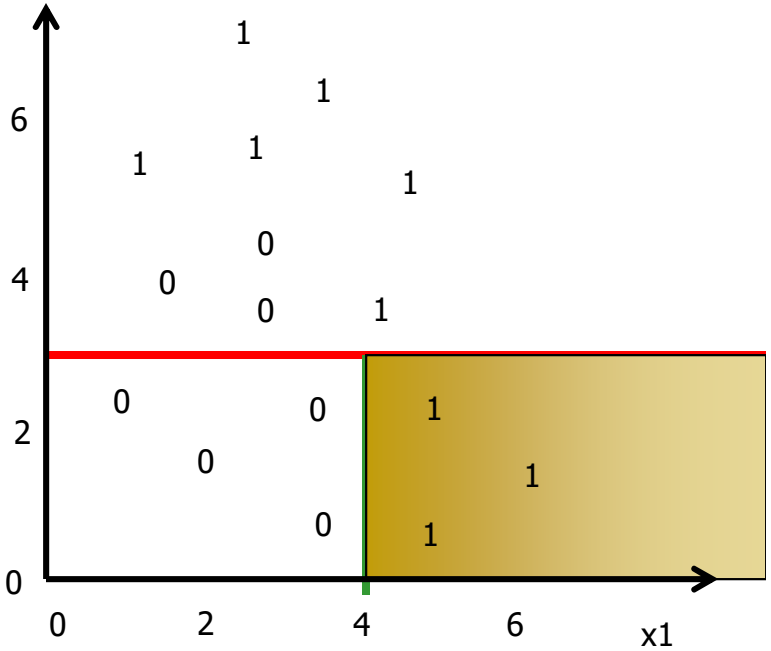
  internal nodes may test value against threshold



DecisionTree Decision Boundaries

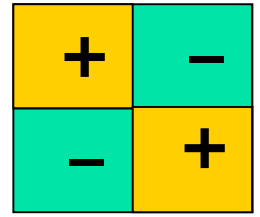
- Decision trees divide feature space into axis-parallel rectangles, labeling each rectangle with one class

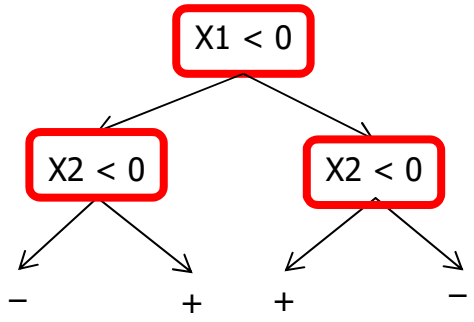




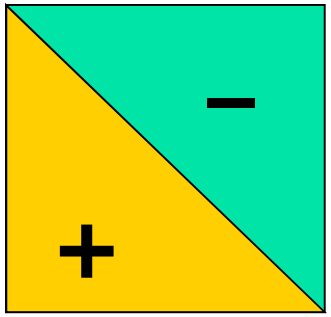
Properties

- Instances represented by Attribute-Value pairs
  - "Bar = Yes", "Size = Large", "Type = French", "Temp = 82.6", ...
  - (Boolean, Discrete, Nominal, Continuous)
- Some concepts are easy to represent: using axis-parallel splits





- Others are difficult :



Can Represent Any Boolean Fn

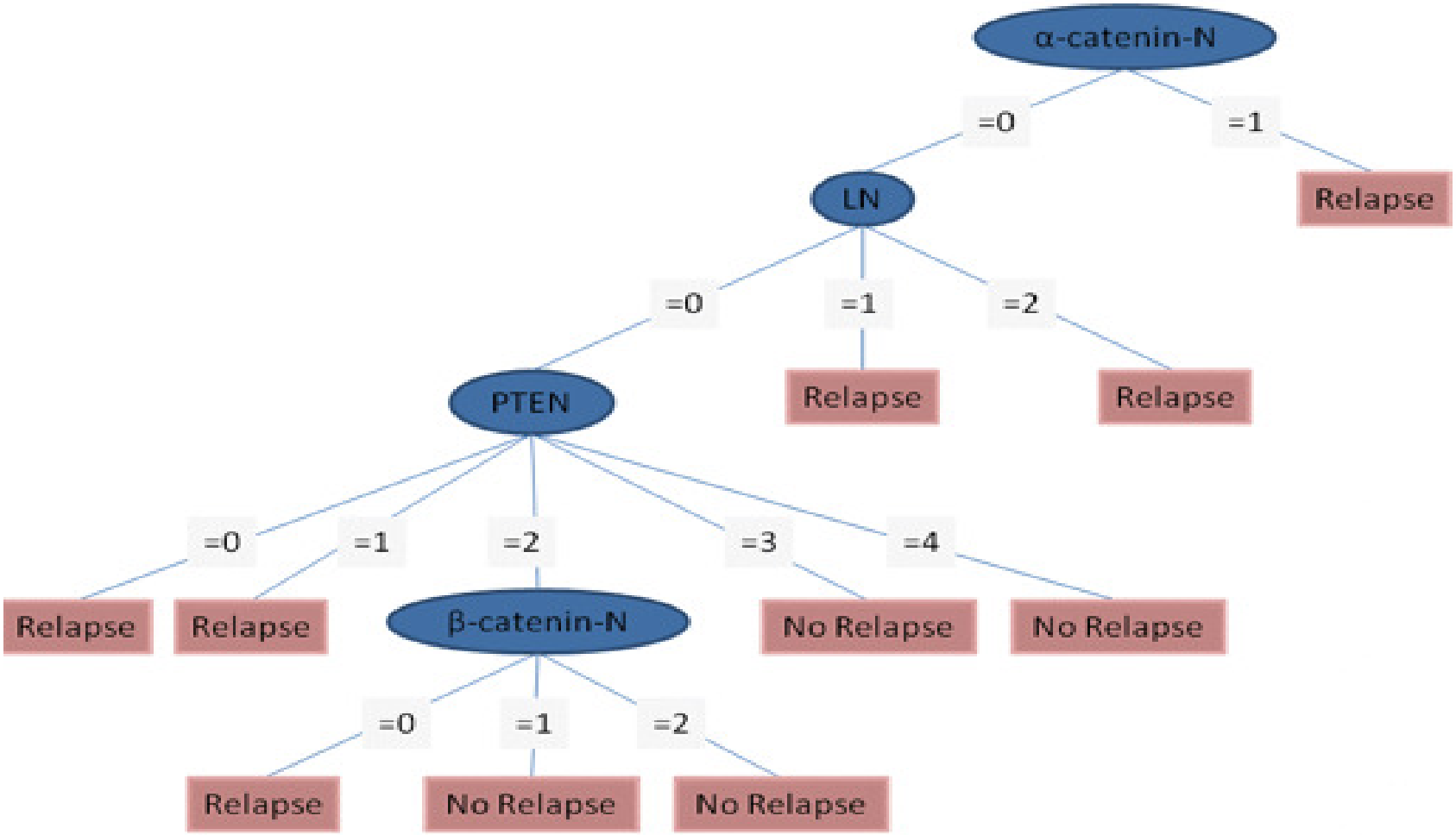
- ∨, &, ⊕, MofN
  (A ∨ B) & (C ∨ ¬D ∨ E)
- Arbitrary boolean functions (DNF)
  - … but may require exponentially many nodes …
  - "disjunctive concepts"

- **Variable-Size Hypothesis Space**
  - Can "grow" hypothesis space by increasing number of nodes
  - **depth 1** ("decision stump"):
    represent any boolean function of one feature

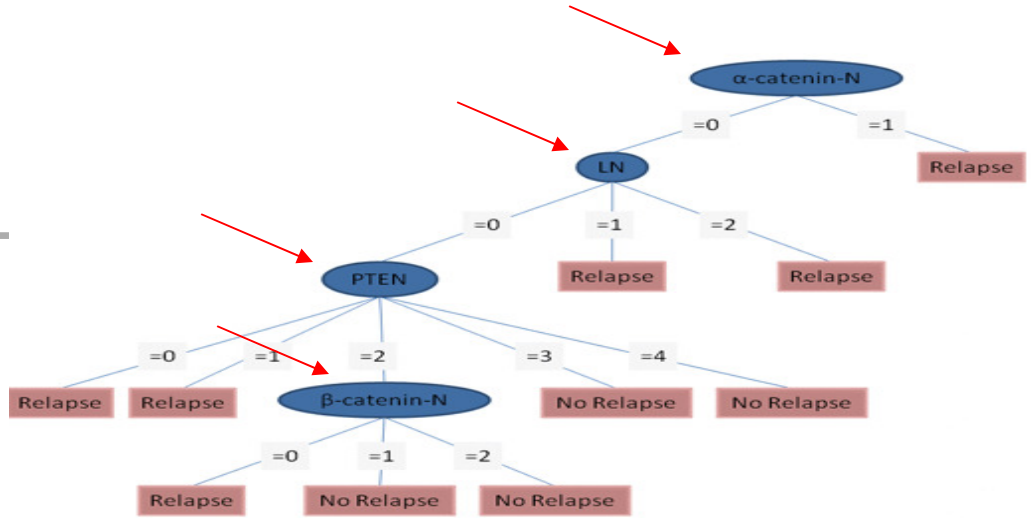
  - **depth 2**: Any boolean function of two features;
    $(x_1 \vee x_2) \,\&\, (\neg x_1 \vee \neg x_3)$

  - …

Learned Decision Tree

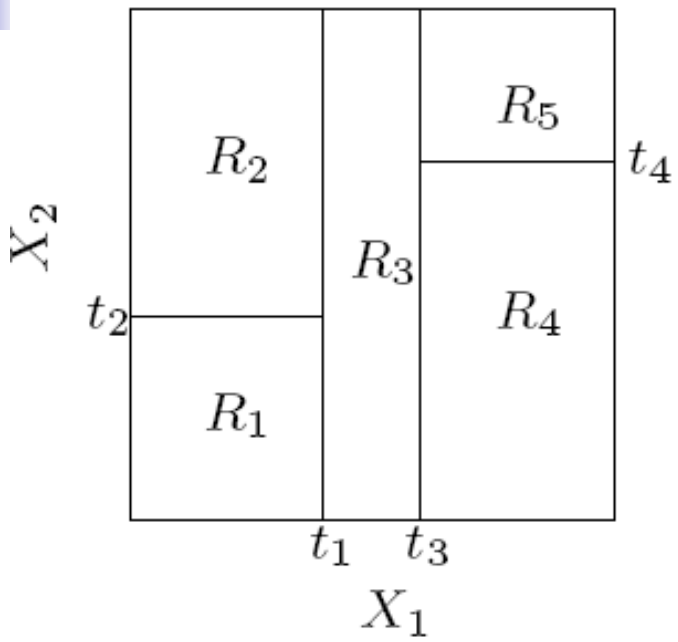


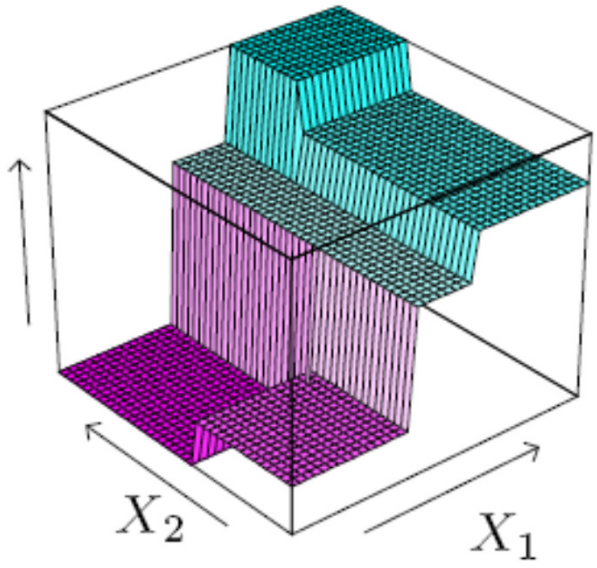
Meaning



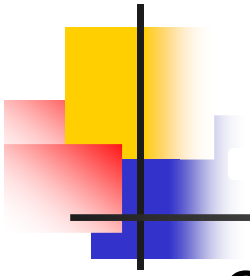
- Concentration of
  α-catenin in nucleus is very important:
  - If >0, probably relapse
- If =0, then #lymph_nodes is important:
  - If >0, probably relapse
- If =0, then concentration of pten is important:
  - If <2, probably relapse
  - If >2, probably NO relapse
- If =2, then concentration of β-catenin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

Regression (Constant) Tree





- Represent each region as CONSTANT
- Alternatively: could have LINEAR FUNCTION at each leaf node
  - … over OTHER variables
  - … or NON-linear …



Using Decision Trees

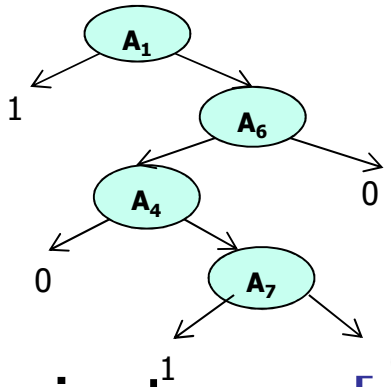
- Our focus:
  - Label is discrete
- Advantages of Decision Trees?
  - Fast to learn
  - Fast(er) to evaluate
  - Uses only subset of features
    - … relevant ones ?
  - Intuitive… easy to EXPLAIN … kinda
- Disadvantages …
  - Typically NOT most accurate
  - Some natural concepts are difficult
- Uses:
  - Credit risk analysis
  - Modeling calendar scheduling preferences
  - Equipment or medical diagnosis
  - …

Learning Decision Trees

- Def'n: Decision Trees
- Algorithm for Learning Decision Trees
  - Greedy
  - Splitting Criterion: Entropy
  - Inductive Bias (Occam's Razor)
- Overfitting
- Topics

Learning / Using Decision Tree

- Classify: DecisionTree × Instance → ClassLabel
  - Given

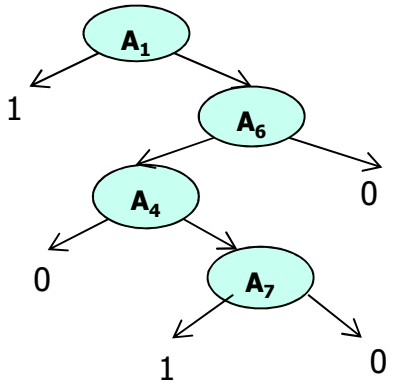


  - classify instance $[A_1 = -,\ A_4 = +, A_6 = +\ ]$ as "0"

- Learn: Data → DecisionTree

A_1	A_2	A_3	..	A_7	A_8	Label
+	−	+	..	+	+	1
−	+	−	..	+	+	0
−	+	−	..	-	+	0
−	−	−	..	+	−	1
:	:	:		:	:	:
−	−	+	..	+	−	0

⇒



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

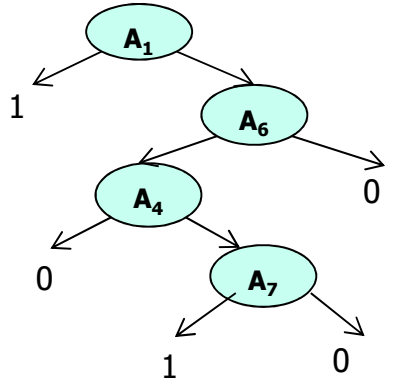
DT_{PT} (Out, Temp, Humid, Wind) $\in$ { Yes, No }

Learning Decision Trees – Easy?

- Learn: Data $\mapsto$ DecisionTree

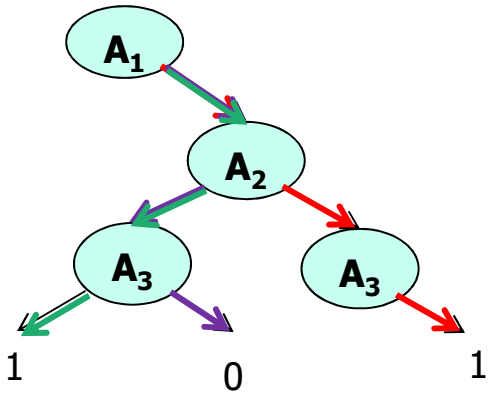
A_1	A_2	A_3	..	A_7	A_8	Label
+	–	+	..	+	+	1
–	+	–	..	+	+	0
–	+	–	..	-	+	0
–	–	–	..	+	–	1
:	:	:		:	:	:
–	–	+	..	+	–	0

$\Rightarrow$



- Option 1: Just store training data

A_1	A_2	A_3	Label
+	+	+	1
+	–	+	0
+	–	–	1



- But ...

Learn ?Any? Decision Tree

- Just produce "path" for each example
  - May produce large tree
  - Any generalization? (what of other instances?)
    [–, +, +] ?
  - Noise in data

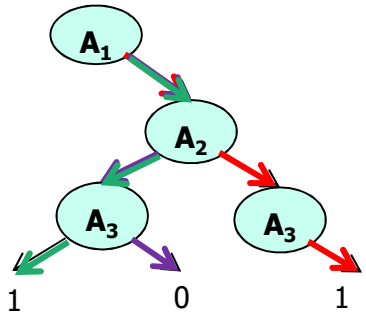
    [[+, –, –], 1]

    mis-recorded as

    [[+, +, –], 1]

    [[+, –, –], **0**]

A_1	A_2	A_3	Label
+	+	+	1
+	–	+	0
+	–	–	1



- Intuition:

  Want SMALL tree

  ... to capture "regularities" in data ...

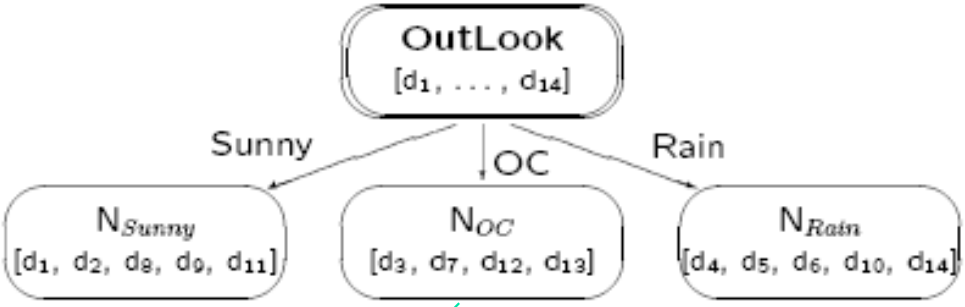
  ... easier to understand, faster to execute, ...

Grow a tree… 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

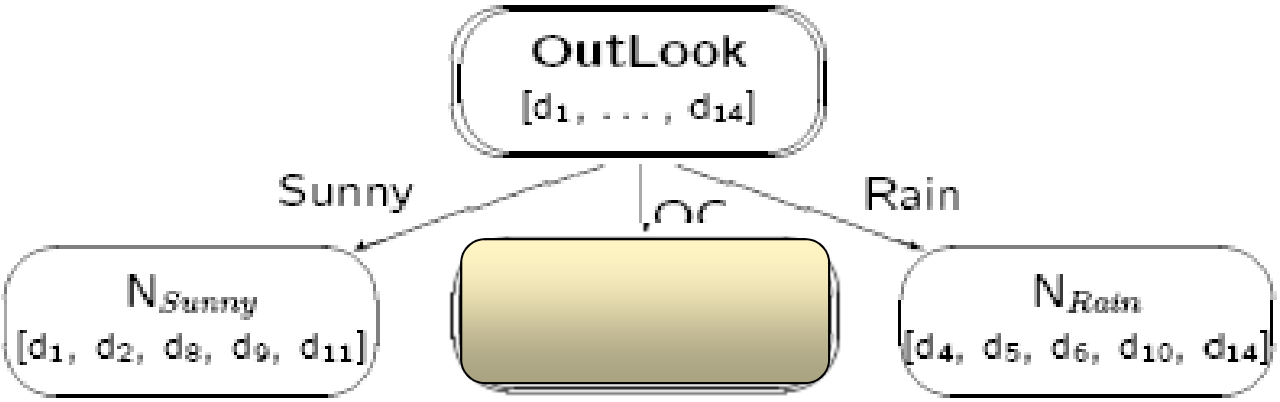
First Split: **Outlook**



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

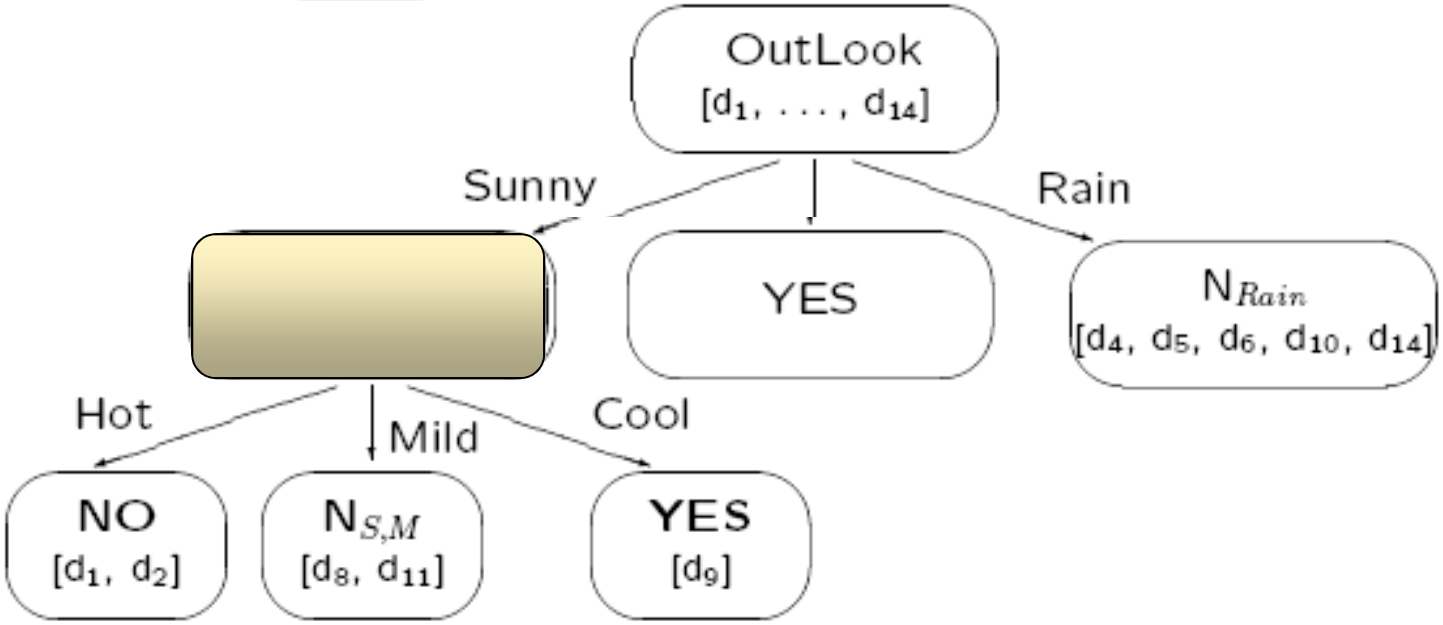
Onto N_{OC} ...

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



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 : labeled_dataset)
  if  ( y ==0 ) for all [x,y] ∈ S: return new_leaf( 0 )
  if  ( y ==1 ) for all [x,y] ∈ S: return new_leaf( 1 )
  /* else */
  x_j = ChooseBestAttribute( S )
  S_0 = { [x,y] ∈ S | x_j == 0 }
  S_1 = { [x,y] ∈ S | x_j == 1 }
  return  new_node( x_j, GrowTree( S_0 ),  GrowTree( S_1 ) )
```

- Many fields independently discovered this learning alg...
- Issues
  - > 2 labels
  - > 2 feature values
    - … continuous values ??
  - ? *ChooseBestAttribute*
  - to purity? … used all attributes?
    - pruning
  - oblique splits

Trivial…

Interesting issues…

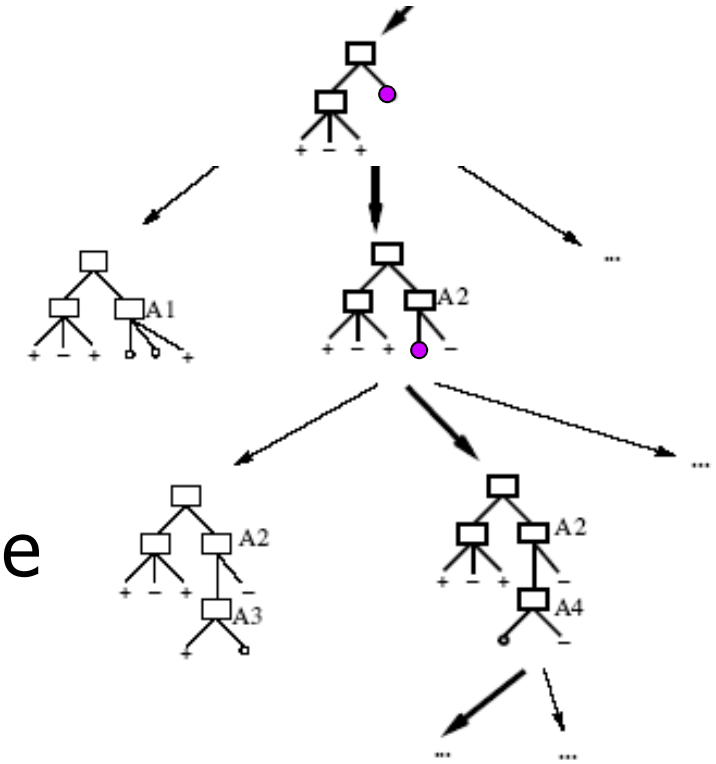
Alg for Learning Decision Trees

```
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 a 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

Jump

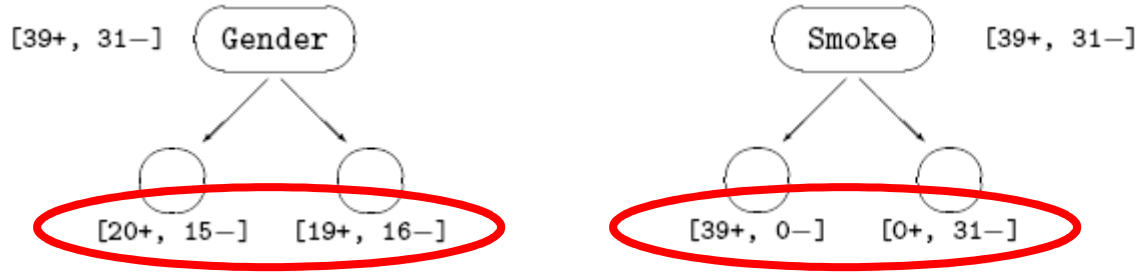
- What attribute to split on?
- Avoid Overfitting
  - When to stop?
  - Any post-processing?
    … should tree by "pruned"?
- How to evaluate classifier (decision tree) ?
  … learner?

```
GrowTree( S : labeled_dataset)
  if  ( y ==0 ) for all [x,y] ∈ S: return new_leaf( 0 )
  if  ( y ==1 ) for all [x,y] ∈ S: return new_leaf( 1 )

  /* else */
  x_j = ChooseBestAttribute( S )
  S_0 = { [x,y] ∈ S | x_j == 0 }
  S_1 = { [x,y] ∈ S | x_j == 1 }
  return  new_node( x_j, GrowTree( S_0 ),  GrowTree( S_1 ) )
```

Choosing Best Splitting Test

- How to choose best feature for split?



- After Gender split, still some uncertainty

  After Smoke split, no more Uncertainty

  $\Rightarrow$ NO MORE QUESTIONS!

  (Here, Smoke is a great predictor for Cancer)

- Want a "measure" that prefers

  Smoke over Gender

Statistics …

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

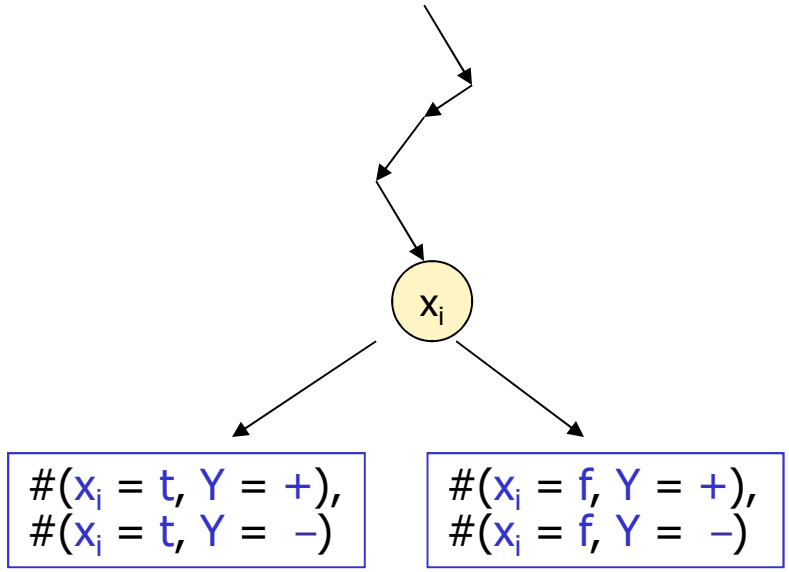
If split on x_i, produce 2 children:

- #(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 = f, Y = –)]



Desired Properties

- Score for split of data D, on x_i, related to

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

- Score S(.) should be…
  - BEST for [+0, –200]
  - WORST for [+100, –100]
  - "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 $\in \{1, \ldots, 100\}$
- Questions
  - Is it 22?
  - More than 90?
  - More than 50?
- Why?
- Q(r) = E[# of additional questions wrt set of size r]
  - = 22? $\frac{1}{100} \times Q(1) + \frac{99}{100} \times Q(99)$
  - ≥ 90? $\frac{11}{100} \times Q(11) + \frac{89}{100} \times Q(89)$
  - ≥ 50? $\frac{50}{100} \times Q(50) + \frac{50}{100} \times Q(50)$

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

Desired Measure: Entropy

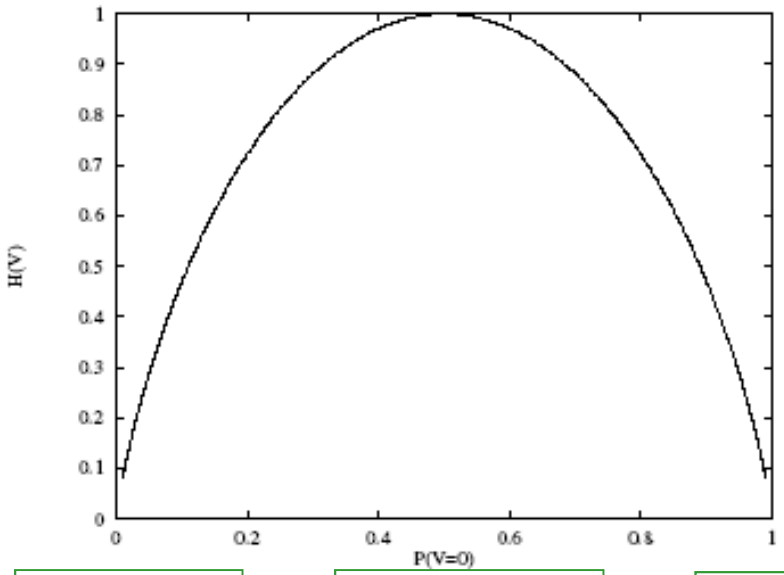
- Entropy of $V = [\ p(V = 1), p(V = 0)\ ]$:

  $H(V) = -\sum_{v_i} P(\ V = v_i\ ) \log_2 P(\ V = v_i\ )$

  $\equiv$ # of bits needed to obtain full info

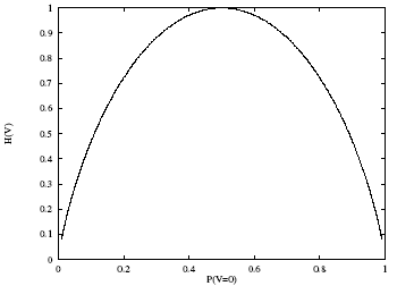
  ... average surprise of result of one "trial" of V

- Entropy $\approx$ measure of uncertainty



+ 200, – 0	+ 100, – 100	+ 0, – 200

Examples of Entropy



- Fair coin:
  - $H\left(\frac{1}{2},\frac{1}{2}\right) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}$ = 1 bit
  - ie, need 1 bit to convey the outcome of coin flip

- Biased coin:

$$H\left(\frac{1}{100},\frac{99}{100}\right) = -\frac{1}{100}\log\frac{1}{100} - \frac{99}{100}\log\frac{99}{100} \approx 0.08 \text{ bit}$$

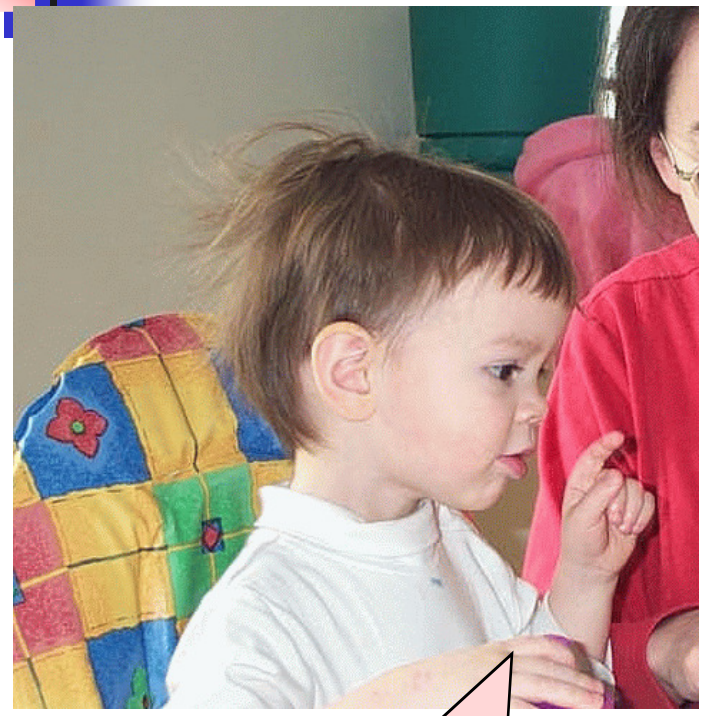
- As P(heads) $\mapsto$ 1, info of actual outcome $\mapsto$ 0

  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

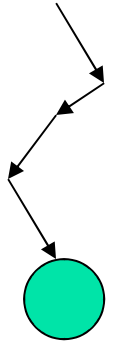
High Entropy

...the values
(locations of soup) unpredictable...
almost uniformly sampled throughout 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
    ⇒ label x as c (ie, entropy is 0)

    +200, – 0
  - If ½ are +; ½ are –
    ⇒ label x as ??? (ie, entropy is 1)

    +100, – 100
- On reaching leaf r with entropy H_r, uncertainty w/label is H_r

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

  ⇒ 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 $\left\{ \begin{array}{ll} p & \text{positive} \\ n & \text{negative} \end{array} \right\}$ 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, *D* :

  $p = n = 6 \Rightarrow$ H(½, ½) = 1 bit

  … so need 1 bit of info to classify example randomly picked from *D*

Remaining Uncertainty

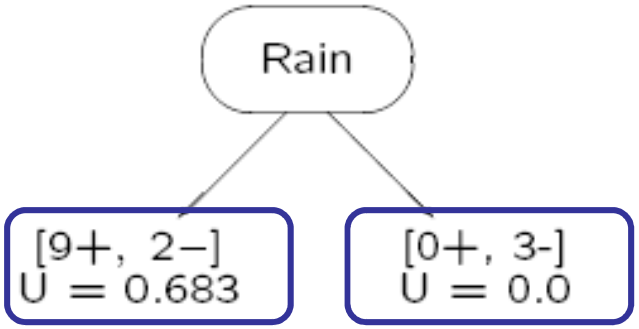
Uncert(D, A) = remaining expected entropy after splitting on A

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

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

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

S : [9+, 5−]

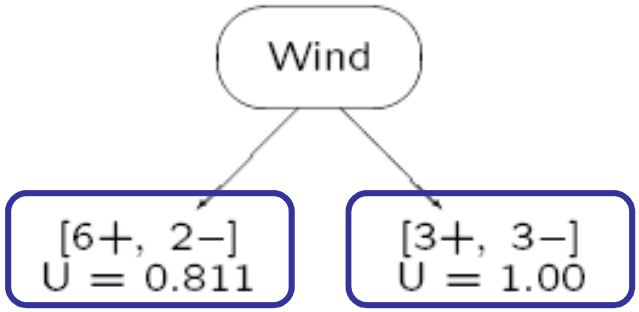


Uncert(D, Rain)

$= \frac{11}{14}\ 0.683 + \frac{3}{14}\ 0.0$

$= 0.536$

S : [9+, 5−]



Uncert(D, Wind)

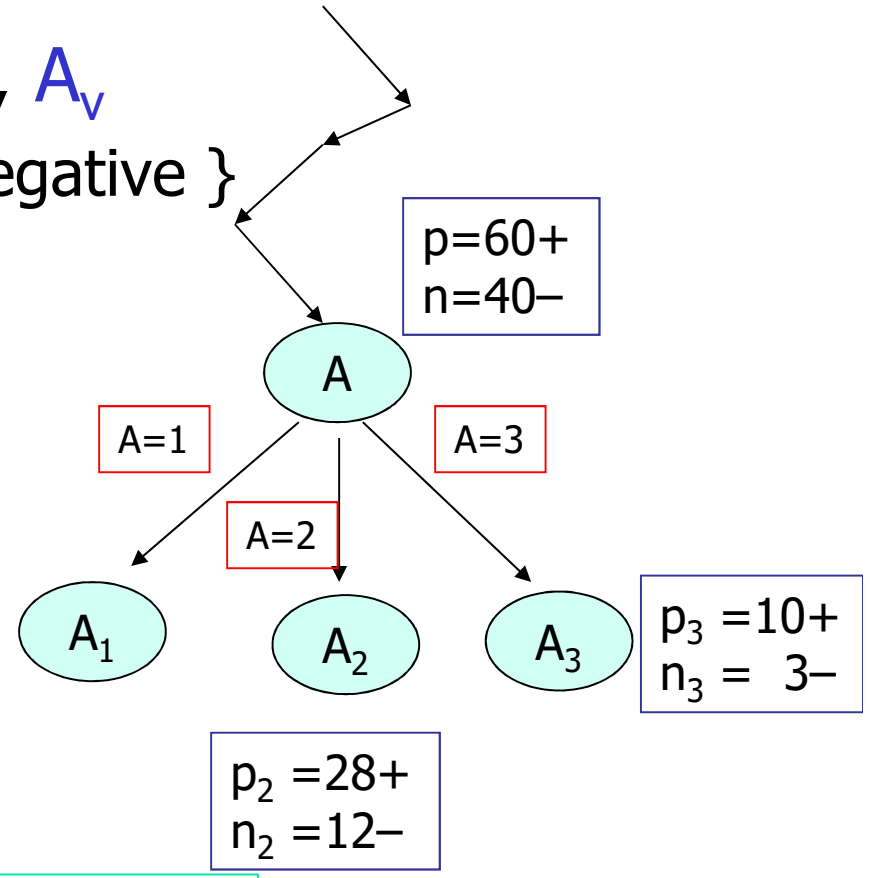
$= \frac{8}{14}\ 0.811 + \frac{6}{14}\ 1.00$

$= 0.892$

Entropy wrt Feature

- Assume [p,n] reach node
- Feature A splits into A_1, …, 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: $H(\frac{28}{40}, \frac{12}{40})$

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

Minimize Remaining Uncertainty

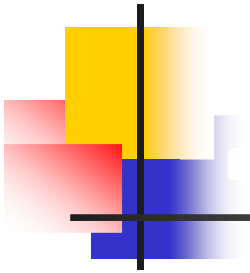
Jump

- 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

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

- Split on feature $A^* = \text{argmin}_A$ { Uncert(A) }

Skip



Occam's Razor

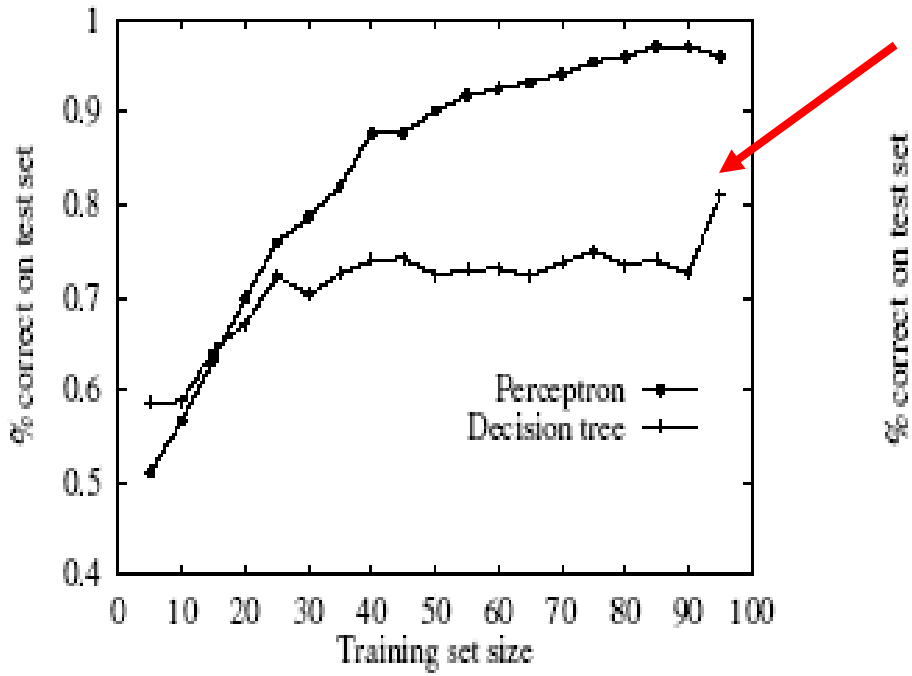
- Q: Why prefer short hypotheses?
- Argument in favor:
  - Fewer short hyps. than long hyps.

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

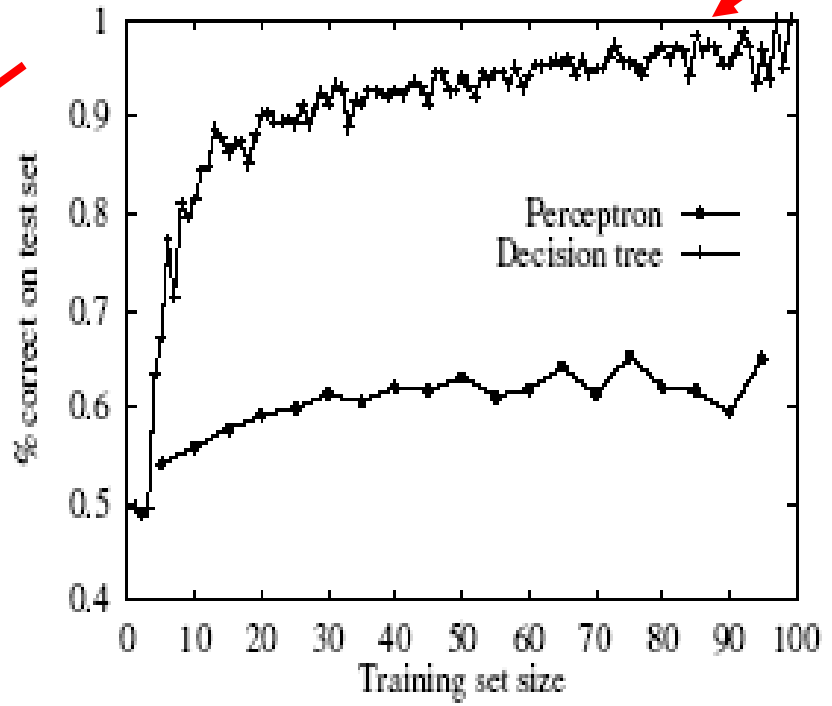
  $\Rightarrow$ a long hyp that fits data *more likely to be* coincidence
- Argument opposed:
  - $\exists$ many ways to define small sets of hyps

    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)

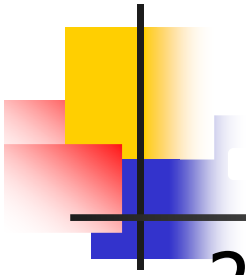


WillWait predicate

Learning Decision Trees

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

Example of Overfitting



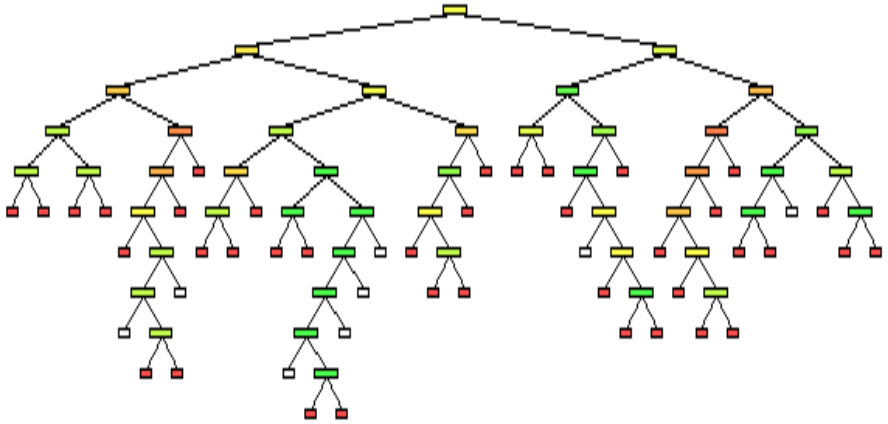
- 25% have butterfly-itis
  - Random
- ½ of patients have $F_1 = 1$
  - Eg: "odd birthday"
- ½ of patients have $F_2 = 1$
  - Eg: "even SSN"
- … for 10 features
- Note feature F_i is INDEPENDENT of class!!

- Decision Tree results
  - over 1000 patients (using these silly features) …

Jump

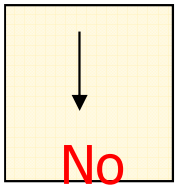
Decision Tree Results

- Standard decision tree learner:



- Error Rate:
  - Train data: 0%
  - New data: 37.5%

- Optimal decision tree:

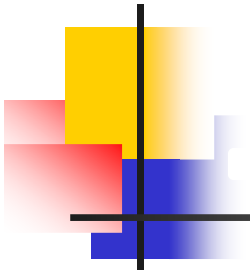


- Error Rate:
  - Train data: 25%
  - New data: 25%

Example of Overfitting

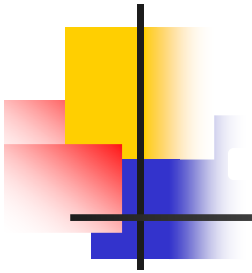
- Spse 10 binary attributes (uniform),
  but class is random: { N w/prob p = 0.75; Y w/prob 1–p = 0.25 }
- (naïve) C4.5 builds nonsensical tree w/ 119 nodes!
  ⇒ Should be SINGLE NODE!
- Error rate (hold-out): 37.5%
  ⇒ Could be only 25% (just say "No")
- Why? Tree assigns leaf "N" w/prob 1–p, "Y" w/prob 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



Mod's ?

```
GrowTree( S : labeled_dataset)
  if  ( y == 0 ) for all [x,y] ∈ S: return new_leaf( 0 )
  if  ( y == 1 ) for all [x,y] ∈ S: return new_leaf( 1 )
  /* else */
  x_j = ChooseBestAttribute( S )
  if x_j ≠ NULL
    S_0 = { [x,y] ∈ S | x_j == 0 }
    S_1 = { [x,y] ∈ S | x_j == 1 }
    tree = new_node( x_j, GrowTree( S_0 ),  GrowTree( S_1 ) )
  else
    tree = new_leaf( majority( S ) )
  return tree
```

- Currently

```
ChooseBestAttribute( S : labeled_dataset)
   best = argmin_x Uncert( S, x )
   if NotGoodEnuf( S, x) return NULL
   return best
```

- Maybe …
  - Decide if this best features is good enough?

Avoid Overfitting #1

Jump

(StopEARLY, Training-Data, DecTree)

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

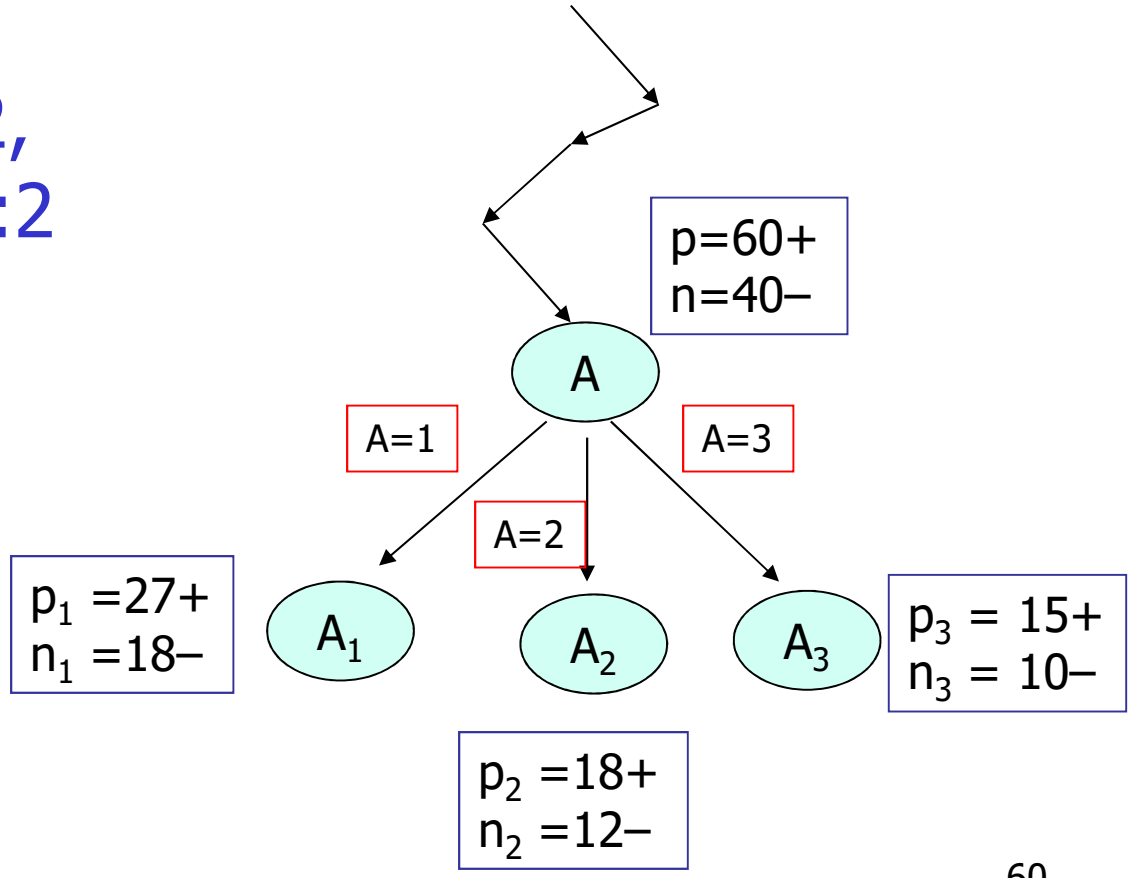
A. Use χ^2 **test**, on data S_r
  - Apply statistical test to compare
    - T_0: leaf at r (majority label) vs
    - T_A: split using feature A
  - Is error of T_A statistically better than T_0?

B. **MDL**: minimize

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

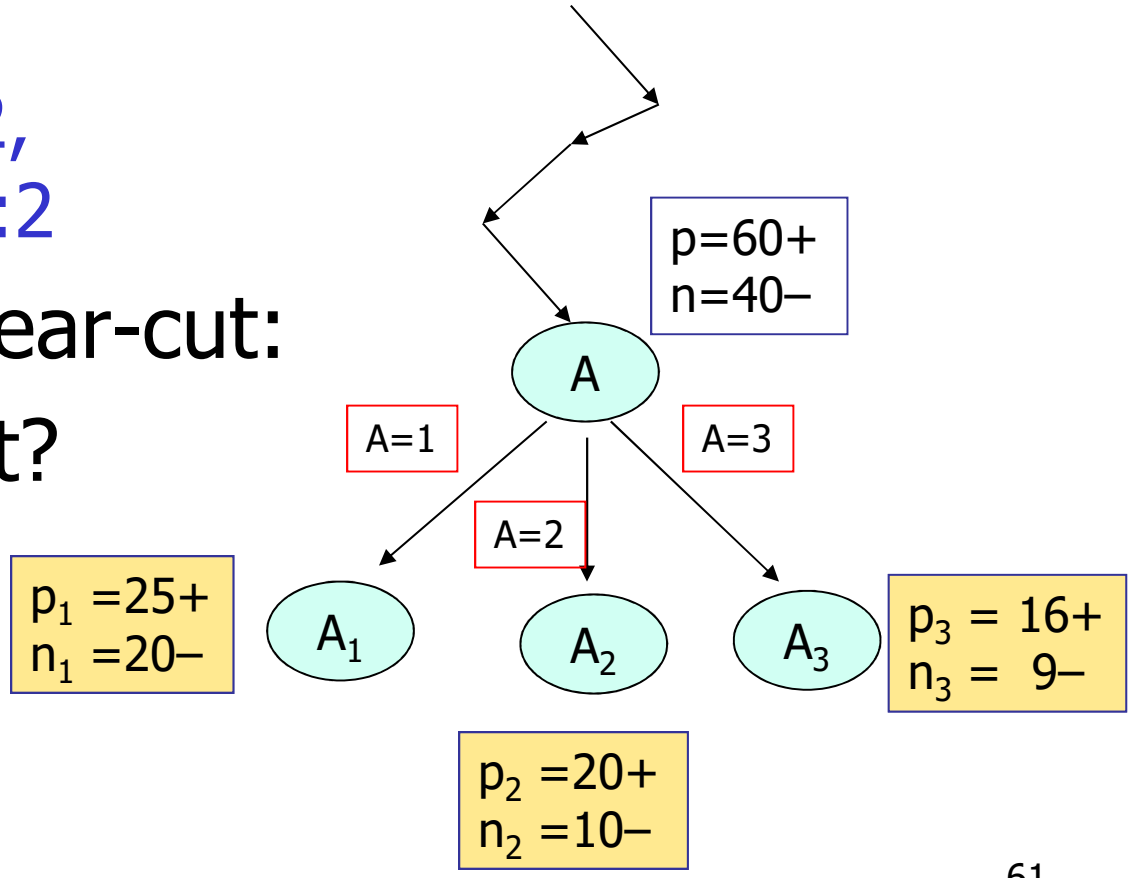
Test for Significance

- Spse A is irrelevant
  - $[p_i, n_i] \propto [p, n]$
  - So if [p,n] = 3:2, then $[p_i, n_i]$ = 3:2



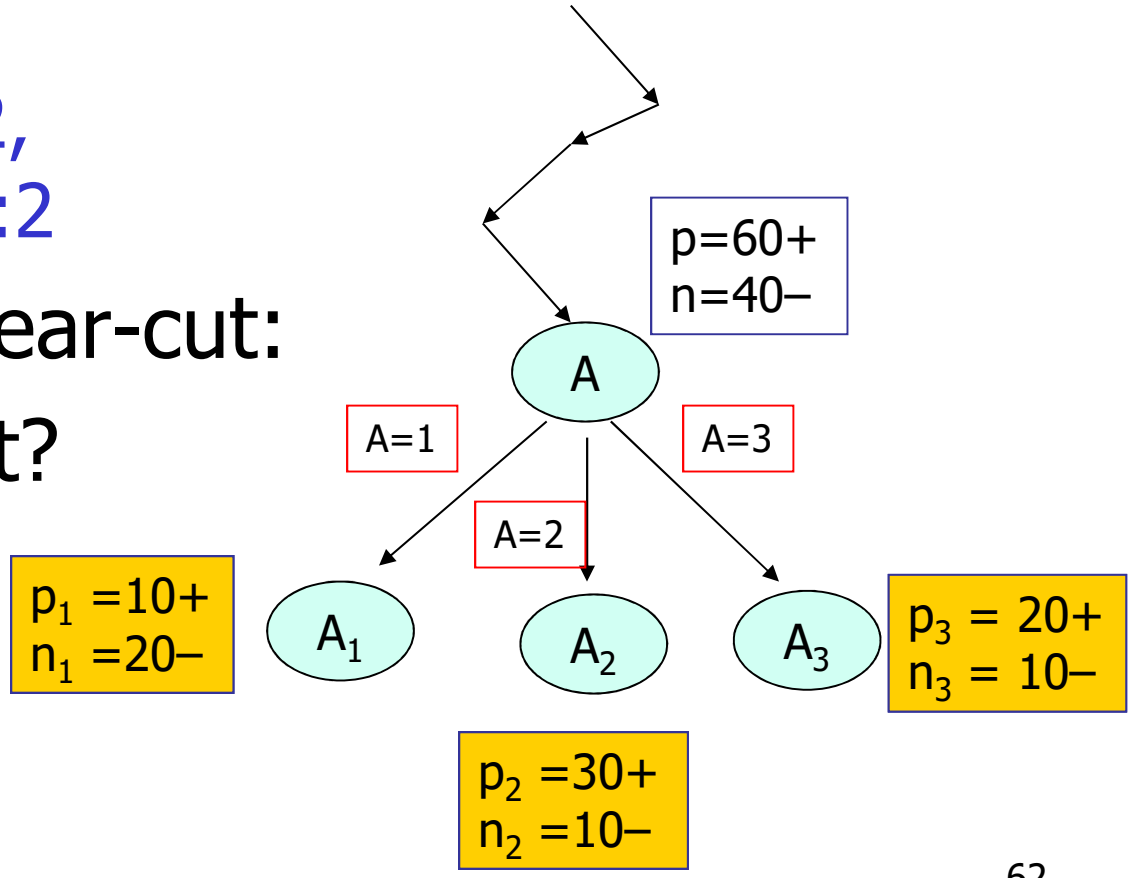
Test for Significance

- 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?



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, dist'n of class labels at node $n_r \equiv$ dist'n after splitting on A
- Observe some difference between these dist'ns.

  What is prob (under H_0) of observing this difference, given m = $|S_r|$ iid samples?
- Defn: $\left\{ \begin{matrix} p \\ n \end{matrix} \right\}$ of S_r are $\left\{ \begin{matrix} \text{positive} \\ \text{negative} \end{matrix} \right\}$
- After splitting on A, get *k* subsets
  - wrt A = i : p_i positives, n_i negatives
- If **H_0** (A irrelevant), would have
  - $\widetilde{p}_i = p \times \frac{p_i + n_i}{p+n}$ positives
  - $\widetilde{n}_i = n \times \frac{p_i + n_i}{p+n}$ negatives

χ^2 Test – con't

$$\sum \frac{(\text{expected} - \text{observed})^2}{\text{expected}}$$

- If H_0 (A irrelevant), would have

$$\widetilde{p_i} = p \times \frac{p_i + n_i}{p+n} \quad \text{positives}$$

$$\widetilde{n_i} = n \times \frac{p_i + n_i}{p+n} \quad \text{negatives}$$

- Measure of deviation:

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

- Under H_0, $D \sim \chi^2_{(k-1)(2-1)}$
  - k = size of A's domain; 2 = #classes
- If A irrelevant (H_0), would you expect to see D ?
- Extend to new internal node if A is not irrelevant… iff $D > T_{\alpha,d}$
  - $T_{\alpha,d}$ is threshold:
  - α = probability of making mistake (rejecting null hyp, when A is irrelevant)
  - d = degree of freedom

χ^2 Table

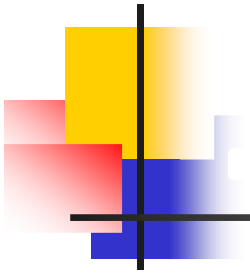
For χ^2 test, with

$\alpha = 0.05$

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

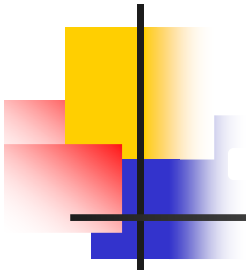


$\mathbb{A}$ wants to transmit to $\mathbb{B}$ classification function $c(\cdot)$

- simplified to:
  - $\mathbb{A}$ and $\mathbb{B}$ agree on instances $\langle x_1, \ldots, x_M \rangle$
  - What should $\mathbb{A}$ send, to allow $\mathbb{B}$ to determine M bits:
    $\langle c(x_1), \ldots, c(x_M) \rangle$
- Option#1: $\mathbb{A}$ can send M "bits"
- Option#2: $\mathbb{A}$ sends "perfect" decision tree d
  s.t. $c(x_i) = d(x_i)$ for each x_i
- Option#3: $\mathbb{A}$ sends "imperfect" decision tree d'
  + set of indices of K exceptions $F = \{ x_{i1}, \ldots, x_{iK} \}$

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

- So... Increase tree-size
  IFF (significant) reduction in #exceptions



Mod's ?

```
GrowTree( S : labeled_dataset)
  if  ( y == 0 ) for all [x,y] ∈ S: return new_leaf( 0 )
  if  ( y == 1 ) for all [x,y] ∈ S: return new_leaf( 1 )
  /* else */
  x_j = ChooseBestAttribute( S )
  if x_j ≠ NULL
    S_0 = { [x,y] ∈ S | x_j == 0 }
    S_1 = { [x,y] ∈ S | x_j == 1 }
    tree = new_node( x_j, GrowTree( S_0 ),  GrowTree( S_1 ) )
  else
    tree = new_leaf( majority( S ) )
  return tree
```

```
ChooseBestAttribute( S : labeled_dataset)
  best = argmin_x Uncert( S, x )
  if NotGoodEnuf( S, x) return NULL
  return best
```

$NotGoodEnuf_{MDL}(S, x)$:

- I_{Tree} = bits to encode subtree rooted in x
- I_{Stop} = bits to encode the exceptions (wrt default)
- Return ($I_{Tree} < I_{Stop}$)

Issues: How to encode tree? Instances? ….

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
    Set $h \leftarrow h_i$

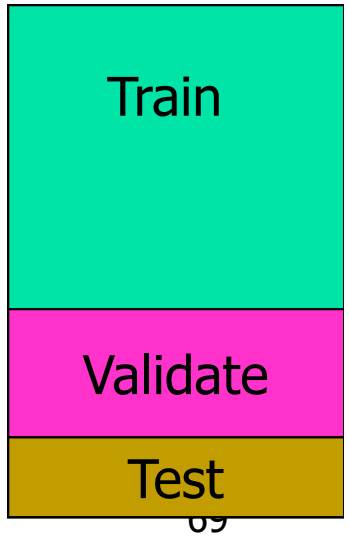
- How to decide if h_i better than h?

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

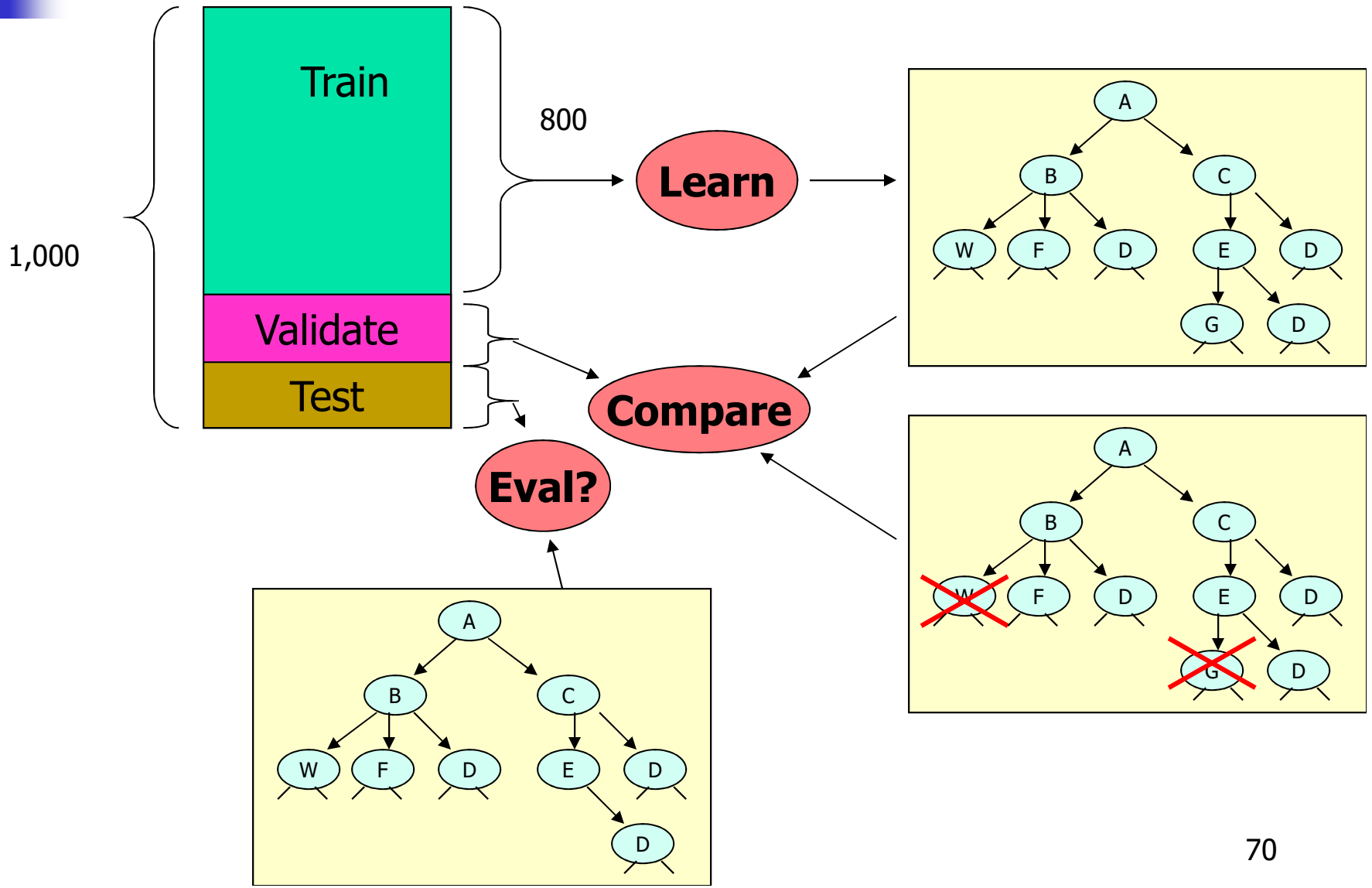
   Problematic if small total # of samples

2. ~~Pessimistic Pruning~~

   ~~... re-use training samples ...~~

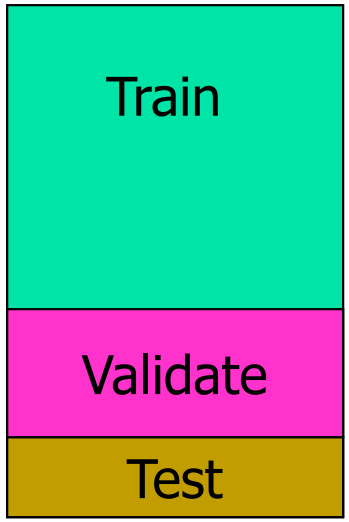


Using Validation Set



Avoid Overfitting#2.1 "Reduced-Error Pruning"

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



Alg: Do until further pruning is harmful:

1. Consider pruning each leaf node
   * Evaluate accuracy on *validation* set
2. Greedily remove the node that most improves accuracy on *validation* set

- Produces possibly smaller subtree... "regularization"
- What if data is limited?

Avoid Overfitting #3
Using Rule Post-Pruning

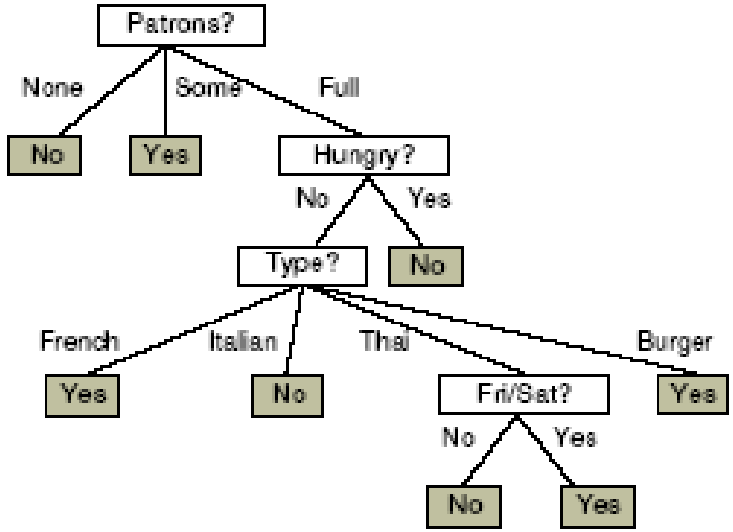
1. Grow decision tree.
   Fit data as well as possible.
   Allow overfitting.
2. Convert tree to equivalent set of rules:
   - One rule for each path from root to leaf.
3. Prune each rule independently of others
   - ie, delete preconditions that improve its accuracy
4. Sort final rules into best sequence-for-use,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

---

- 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
  - …

Skip

Attributes with Many Values

- Problem: *Uncert* prefers attribute with many values, as Entropy ≈ ln(k)
  - Problem: what about using
    Date = Jun 3 1996 {as LOTS of Date's ⇒ k is large }
    PatientId = 217331 {as LOTS of PatientId's ⇒ k is large }
- If using a multi-way split:
  perhaps use *EntropyRatio* instead of Entropy

$$\text{EntropyRatio}(S, A) = \frac{\text{Entropy}(S, A)}{\text{SplitInfo}(S, A)}$$

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

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

- Other Approaches:
  - Test one value versus all of the others?
  - Group values into two disjoint subsets?

Finding Split for Real-Valued Features

- Best threshold θ_j for attribute X_j
  - Sort training instances by $x_{i,j}$

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

  - For each possible threshold $\theta \in \Re$,

	$< \theta$	$\geq \theta$
A	$n_{A,l}(\theta)$	$n_{A,r}(\theta)$
B	$n_{B,l}(\theta)$	$n_{B,r}(\theta)$
(both)	$n_l(\theta)$	$n_r(\theta)$

  - Compute mutual information $MI(\theta)$ wrt label {A, B}:

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

- Use $\theta^* = \text{argmax}_\theta\{ MI(\theta) \}$

Finding Split for Real-Valued Features

For quality of resulting DecTree:
find θ_j within each CrossValidation fold

- Best threshold θ_j for attribute X_j
  - Sort training instances by $x_{i,j}$

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

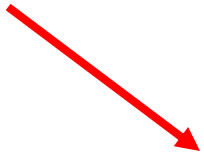
- Eg, for θ=1.2:

< 1.2	≥ 1.2
$n_{A,l}$=3	$n_{A,r}$=1
$n_{B,l}$=1	$n_{B,r}$=4
n_l = 4	n_r=5

  - Mutual information = 0.2294
- Sequentially set $\theta = \frac{x_{i,j} + x_{i,j+1}}{2}$ over all j
- Note: just need to consider j s.t. $y_j \neq y_{j+1}$!

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
  - …



Skip

Desired Properties

Repeat!

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

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

- 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

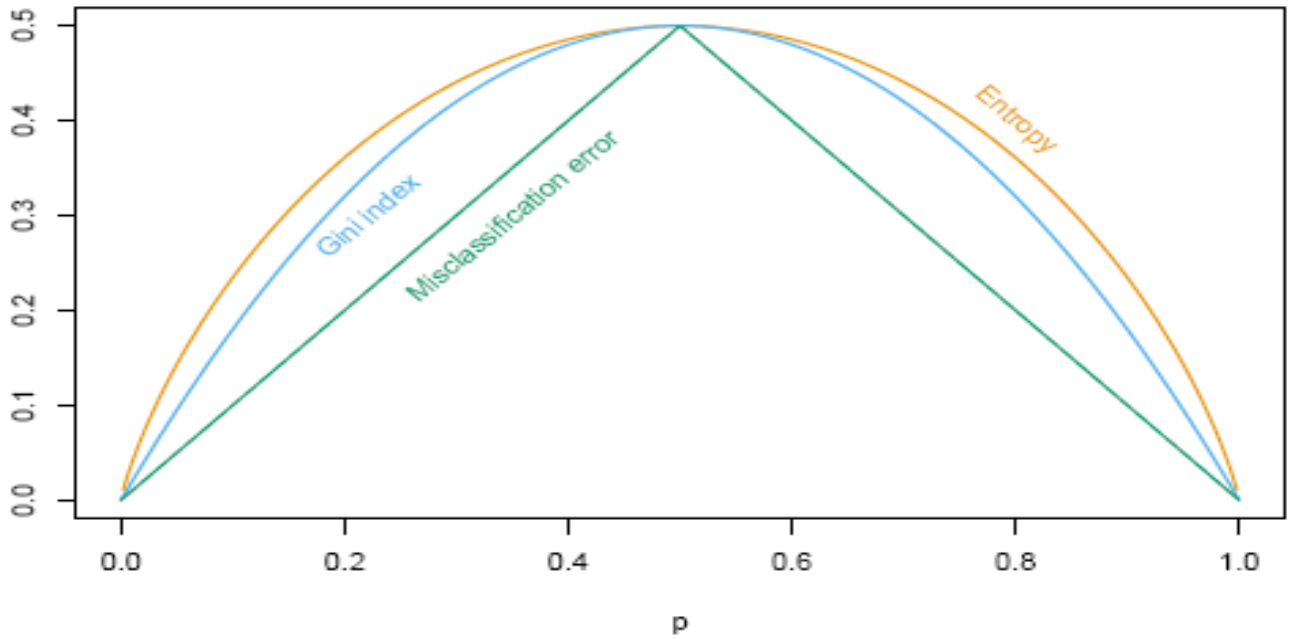
Other Splitting Criteria

- Why use *Uncert* as splitting criterion?
- Want:
  - Large "use me" value if split is [85, 0, 0, …, 0]
  - Small "avoid me" value if split is [5, 5, 5, …, 5]
- True of *Uncert*, *Uncert_Ratio*… and also for…
- χ^2 statistical tests:
  For each attr A, compute deviation:

$$\chi^2(\mathrm{A}) = \sum_{i=1}^{k} \frac{(p_i - \widehat{p_i})^2}{\widehat{p_i}} + \frac{(n_i - \widehat{n_i})^2}{\widehat{n_i}}$$

- GINI index: $\mathrm{GINI(A)} = \sum_i \sum_{j \neq i} p_i\, p_j = 1 - \sum_i p_i^2$
- Others: "Marshall Correction"
  "G" statistic

Node Impurity Measures



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

Example of χ^2

- Attributes T_1, T_2 class c

Data:

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

T_2 =	Yes	No	c =
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:

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

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

Scores: $GINI(T_1) =$

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

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

$$= 0.21429$$

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

$$= 0.026667$$

- As $GINI(T_1) > GINI(T_2)$, split on T_1
- As "$(\tfrac{5^2}{10}+\tfrac{5^2}{10})$" 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 *Uncert(S,A)* by
  - $Gain^2(S,A) / Cost(A)$ [Tan/Schlimmer'90]
  - $[2^{Gain(S,A)} - 1] / [Cost(A)+1]^w$

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

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

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

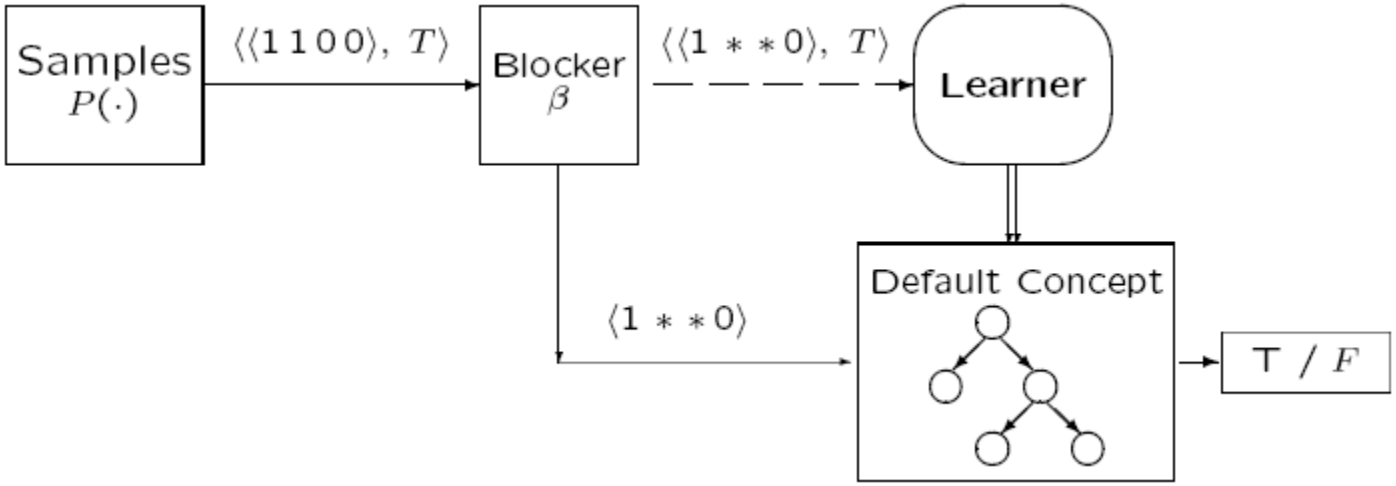
- Similar ideas ...
- ISSUE: Why are values missing?
  - Transmission Noise
  - "Bald men wear hats"
  - "You don't care"

  See [Schuurmans/Greiner'94]

Handling Missing Values: Proportional Distribution

- Associate weight w_i with example $[\mathbf{x}_i, y_i]$
  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 $[\mathbf{x}_i, y_i]$ 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 $p_L \times P(\tilde{y}_L \mid x) + p_R \times P(\tilde{y}_R \mid x)$

Questions



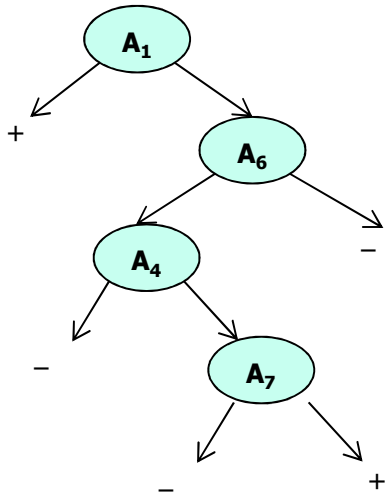
1. How to represent default concept?
2. When is best default concept learnable?
3. If so, how many instances are required?
4. Is it better to learn from …
   - Complete Samples, or
   - Incomplete Samples?

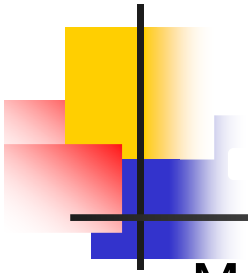
Learning Task

Input:

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

Output: (small) decision tree consistent with data





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 classification or redundant with features already present in the case description"

  $\Rightarrow$ Our Situation!!
- Why Learn?
  A: ... when experts are ...
  - not available, or
  - unable to articulate classification process

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:

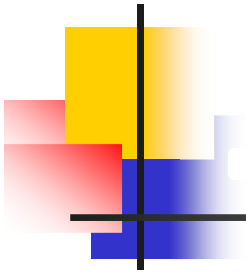
Algorithm GrowDT

Collect "enough" labeled (blocked) instances

Let ***root*** *= never-blocked feature* x_i

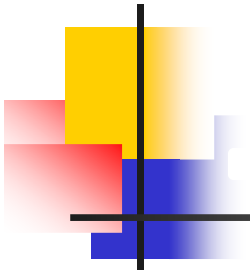
Split instances by $x_i = 1$ *vs* $x_i = 0$*,*

and recur (until purity)



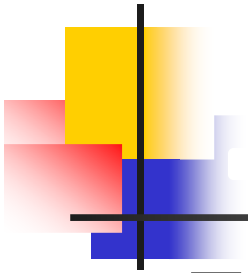
Conclusions

- Decision trees are extremely popular ML mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- Software Systems:
  - C5.0 (from ID3, C4.5) [Quinlan'93]
  - CART
  - …
- Many many applications
- Need to avoid overfitting



What we haven't discussed…

- Decision Stumps" (1-level DT)
  seem to work surprisingly well
- Efficient alg's for learning optimal "depth-k decision trees"
  … even if continuous variables
- Oblique Decision Trees
  Not just "$x_3 > 5$", but "$7\ x_4 - 3\ x_8 > 91$"
- Use of prior knowledge
  - Incremental Learners ("Theory Revision")
  - "Relevance" info
- Real-valued *outputs* – Regression Trees
- Bayesian Decision Trees
  - a different approach to preventing overfitting
- Boosting: a simple way to improve accuracy
  - … ensemble methods

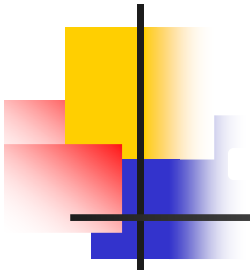


For more information

- 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

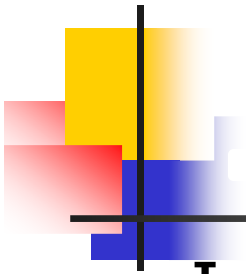


Nice Web Sites

Check out…

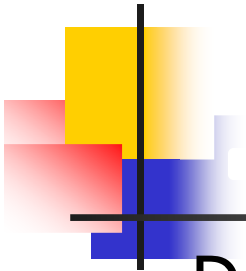
- http://webdocs.cs.ualberta.ca/~aixplore/learning/DecisionTrees/index.html

- http://www.r2d3.us/visual-intro-to-machine-learning-part-1/



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 include real-valued inputs
- Computational complexity
  - straightforward, cheap
- Coping with Missing Data
- Alternatives to Information Gain for splitting nodes



Why include these slides?

- Decision trees are often used as they are "intuitive"
  - Suggest explanations for decisions
  - … kinda …
- But typically not as accurate as other models
- Include this material as:
  - another approach to learning (beyond linear-ish models)
  - other intuitions about learning process
  - Entropy, information content, …
  - χ^2, MDL, …
  - Missing data ("You don't care", not "MCAR", …)
  - alternative tweaks…