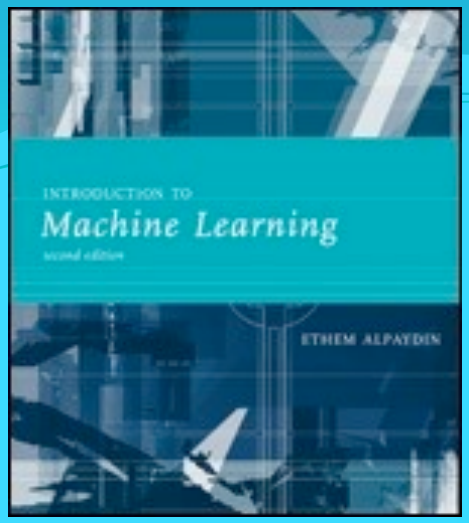


Lecture Slides for

INTRODUCTION TO

Machine Learning

2nd Edition

CHAPTER 9:

Decision Trees

ETHEM ALPAYDIN

Edited and expanded for CS 4641 by Chris Simpkins

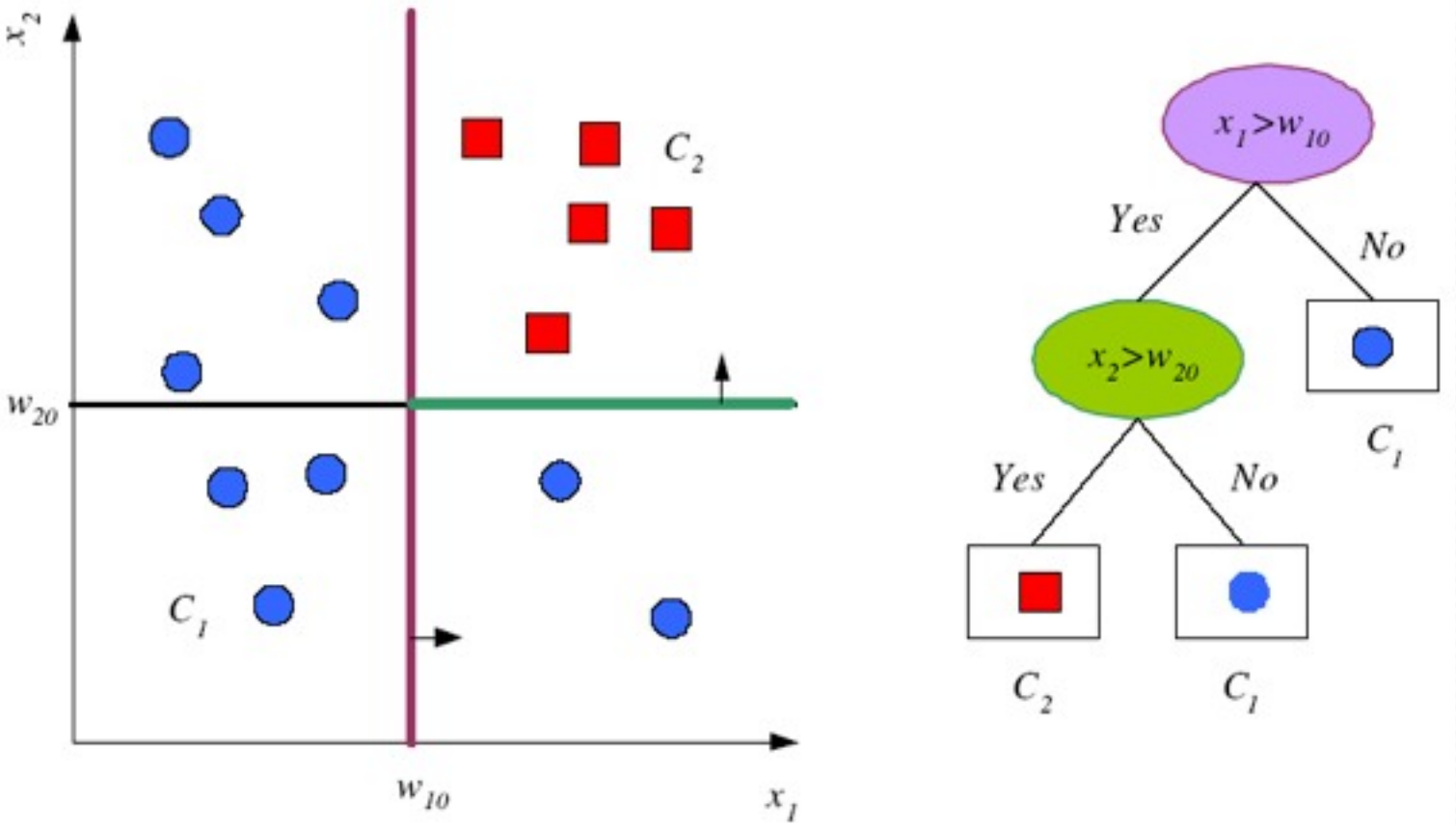
alpaydin@boun.edu.tr

http://www.cmpe.boun.edu.tr/~ethem/i2ml2e

Overview

- Univariate decision trees
- Building classification trees
- Dealing with overfitting
- Extracting rules from decision trees

Tree Uses Nodes, and Leaves



Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Numeric x_i : Binary split : $x_i > w_m$
    - Discrete x_i : n-way split for n possible values
  - Multivariate: Uses all attributes, $\boldsymbol{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit
- Learning is greedy; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)

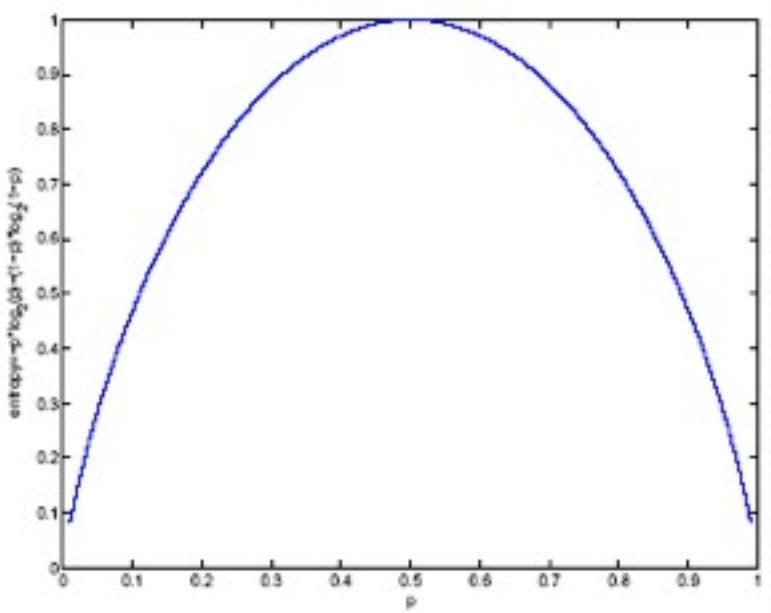
Classification Trees (ID3, CART, C4.5)

- For node m, N_m instances reach m, N^i_m belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$

- Node m is pure if p^i_m is 0 or 1
- Measure of impurity is entropy

$$\mathcal{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m$$



Best Split

- If node m is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: N_{mj} of N_m take branch j. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)

```
GenerateTree(X)
    If NodeEntropy(X)< θ_I /* eq. 9.3
        Create leaf labelled by majority class in X
        Return
    i ← SplitAttribute(X)
    For each branch of x_i
        Find X_i falling in branch
        GenerateTree(X_i)
SplitAttribute(X)
    MinEnt← MAX
    For all attributes i = 1,...,d
        If x_i is discrete with n values
            Split X into X_1,...,X_n by x_i
            e ← SplitEntropy(X_1,...,X_n) /* eq. 9.8 */
            If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
            For all possible splits
                Split X into X_1, X_2 on x_i
                e←SplitEntropy(X_1, X_2)
                If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

Summary of Main Loop

- Pick *A*, the "best" decision attribute for examples at current node using impurity measure
- For each value of *A*, create new descendant leaf nodes of current node
- Sort training examples to leaf nodes according to their values for attribute A
- If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes (turning some of them into internal nodes)

Example: Play tennis today?[1]

Day	Outlook	Temperatur	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

[1]Example from Mitchell 1997

Choosing the first split attribute

- Outlook:

$$
\begin{aligned}
I'_{root} &= -\sum_{j\in\{Sunny,Overcast,Rain\}} \frac{N_j}{N} \sum_{i\in\{Yes,No\}} p_j^i \log_2 p_j^i \\
&= -[\frac{5}{14}(\frac{2}{5}\log_2\frac{2}{5}+\frac{3}{5}\log_2\frac{3}{5})+\frac{4}{14}(\frac{4}{4}\log_2\frac{4}{4}+\frac{0}{4}\log_2\frac{0}{4})+\frac{5}{14}(\frac{3}{5}\log_2\frac{3}{5}+\frac{2}{5}\log_2\frac{2}{5})] \\
&= 0.693
\end{aligned}
$$

- Temperature:

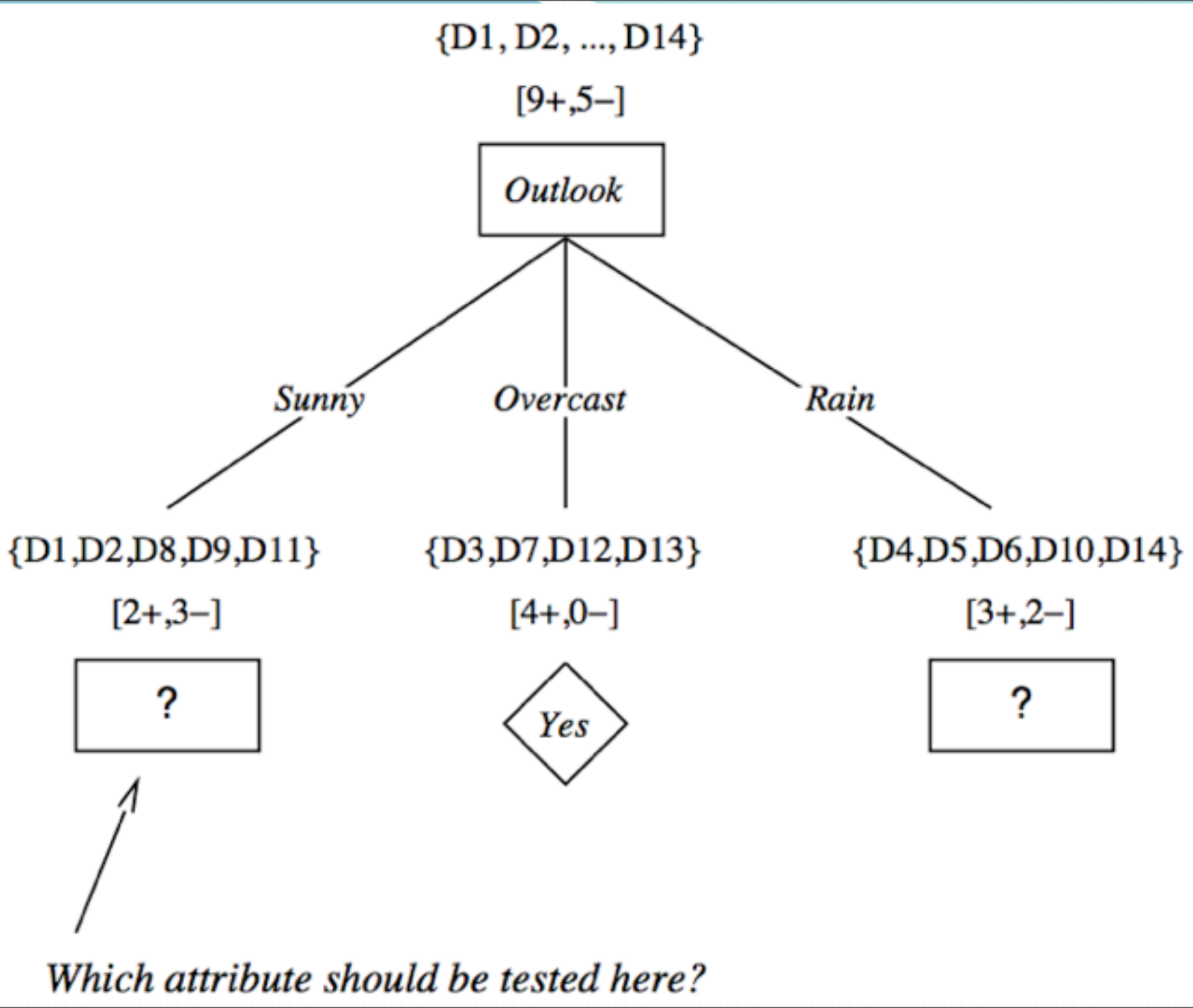
$$
\begin{aligned}
I'_{root} &= -\sum_{j\in\{Hot,Mild,Cool\}} \frac{N_j}{N} \sum_{i\in\{Yes,No\}} p_j^i \log_2 p_j^i \\
&= -[\frac{4}{14}(\frac{2}{4}\log_2\frac{2}{4}+\frac{2}{4}\log_2\frac{2}{4})+\frac{6}{14}(\frac{4}{6}\log_2\frac{4}{6}+\frac{2}{6}\log_2\frac{2}{6})+\frac{4}{14}(\frac{3}{4}\log_2\frac{3}{4}+\frac{1}{4}\log_2\frac{1}{4})] \\
&= 0.915
\end{aligned}
$$

- Humidity:

$$
\begin{aligned}
I'_{root} &= -\sum_{j\in\{High,Normal\}} \frac{N_j}{N} \sum_{i\in\{Yes,No\}} p_j^i \log_2 p_j^i \\
&= -[\frac{7}{14}(\frac{3}{7}\log_2\frac{3}{7}+\frac{4}{7}\log_2\frac{4}{7})+\frac{7}{14}(\frac{6}{7}\log_2\frac{6}{7}+\frac{1}{7}\log_2\frac{1}{7})] \\
&= 0.789
\end{aligned}
$$

- Wind:

$$
\begin{aligned}
I'_{root} &= -\sum_{j\in\{Strong,Weak\}} \frac{N_j}{N} \sum_{i\in\{Yes,No\}} p_j^i \log_2 p_j^i \\
&= -[\frac{6}{14}(\frac{3}{6}\log_2\frac{3}{6}+\frac{3}{6}\log_2\frac{3}{6})+\frac{8}{14}(\frac{6}{8}\log_2\frac{6}{8}+\frac{2}{8}\log_2\frac{2}{8})] \\
&= 0.892
\end{aligned}
$$


{D1, D2, ..., D14}
[9+,5–]
Outlook
Sunny
Overcast
Rain
{D1,D2,D8,D9,D11}
[2+,3–]
{D3,D7,D12,D13}
[4+,0–]
{D4,D5,D6,D10,D14}
[3+,2–]
?
Yes
?
Which attribute should be tested here?

Choosing an attribute for Sunny node

- X_{Sunny} = {D1, D2, D8, D9, D11}

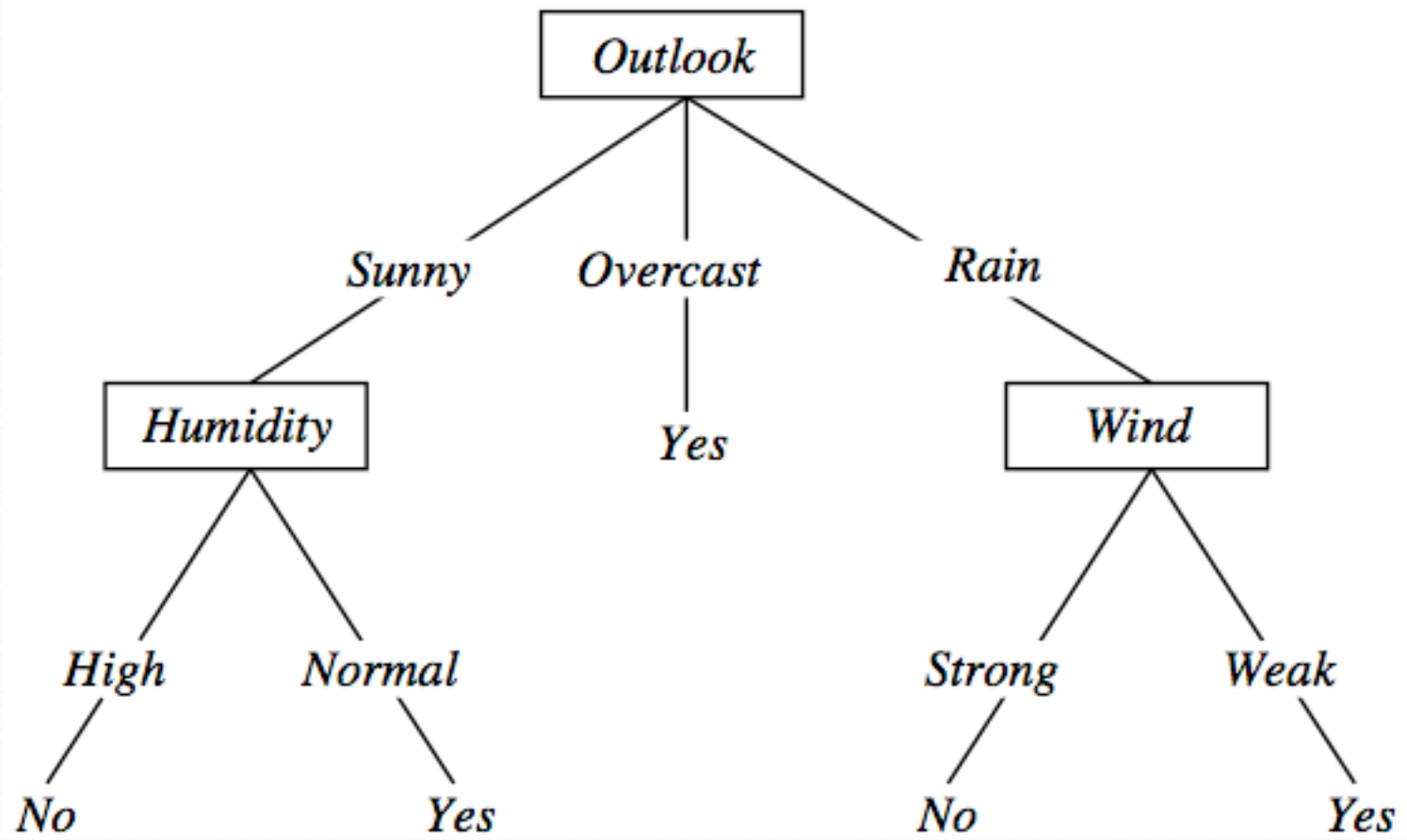
Day	Outlook	Temperatur	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

- Humidity:

$$
\begin{aligned}
I'_{sunny,humidity} &= -\sum_{j\in\{High,Normal\}} \frac{N_j}{N} \sum_{i\in\{Yes,No\}} p_j^i \log_2 p_j^i \\
&= -[\frac{3}{5}(\frac{0}{3}\log_2\frac{0}{3}+\frac{3}{3}\log_2\frac{3}{3})+\frac{2}{5}(\frac{2}{2}\log_2\frac{2}{2}+\frac{0}{2}\log_2\frac{0}{2})] \\
&= 0
\end{aligned}
$$

- Do same calculation for Temperature and Wind
- Humidity has lowest entropy, so Humidity is split attribute at Sunny node

Final Induced Tree



Observations

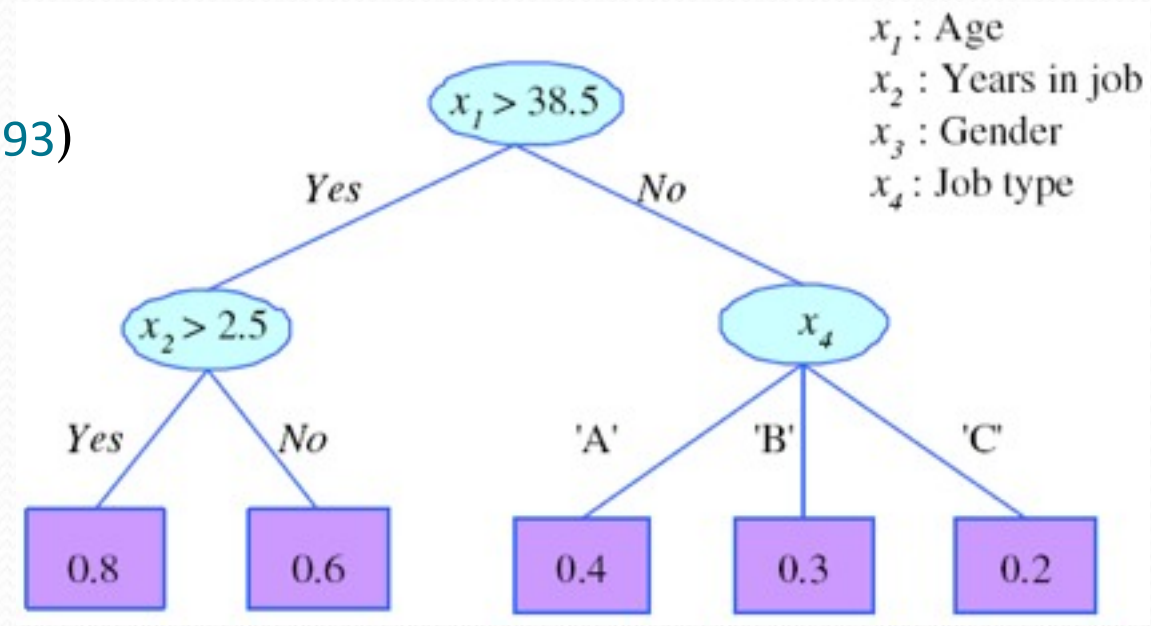
- Temperature was not needed
  - Decision tree can be used for feature extraction (a.k.a. feature selection)
- Tree is short
  - ID3 family of algorithms have preference bias for short trees
  - Occam's Razor: shorter trees are better

Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

Conclusion

- Decision trees good when:
  - Instances described by attribute-value pairs
  - Target function is discrete
  - Interpretability of learned hypothesis (e.g., as a rule set) is desired
  - Training data may be noisy
- Decision trees are a good first algorithm to try