Decision Tree

Chenghua Lin
Computing Science
University of Aberdeen

The Supervised Classification Task

- Input: Collection of instances with a set of attributes x and a special nominal attribute Y called class attribute
- Output: A model that accurately predicts y from x on previously unseen instances
  - Previously unseen instances are called test set
- Usually input collection is divided into
  - Training set for building the required model
  - Test set for evaluating the model built

Example Data

Class Attribute

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

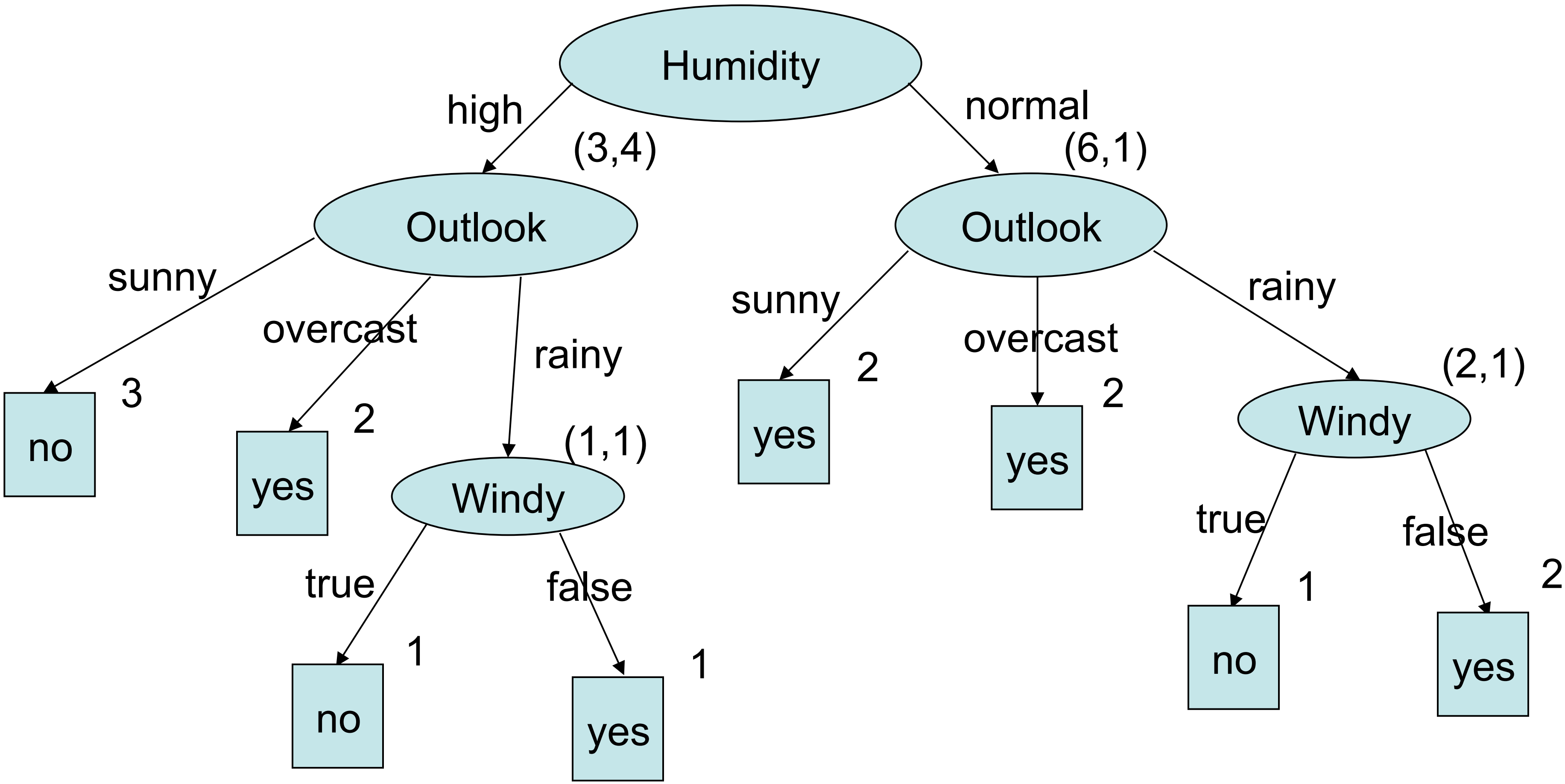
Several Techniques

- Nearest neighbour methods
- Naïve Bayes and Bayesian networks
- Discriminant analysis approach, e.g. SVM
- Decision tree based methods
  - Study in this lecture

Decision Tree Construction

- Recursive procedure
  - Select an attribute to place at the root node
  - Make one branch for each possible value of the selected attribute
  - For each branch repeat the above two steps recursively
    - Using only those instances that actually reach the branch
  - Stop developing a branch if it has instances belonging to the same class
- Several decision trees are possible
  - Based on the order of the selection of the attributes

Example Decision Tree 1



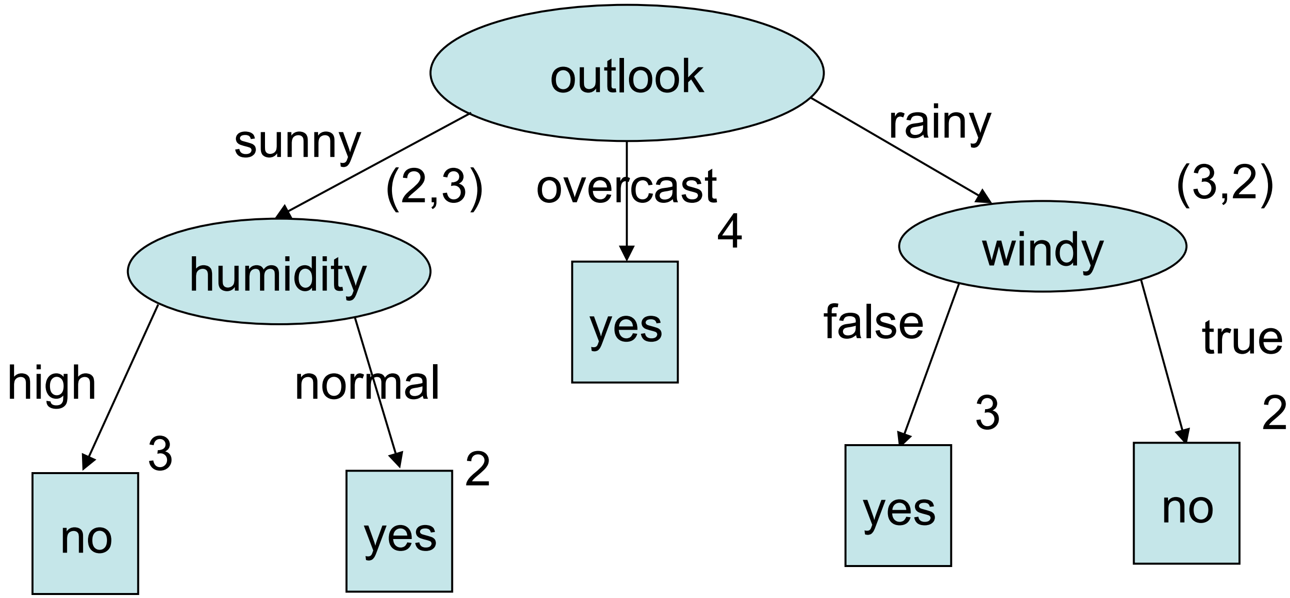
Observations:
Outlook and windy repeated in this tree
Windy tested only when outlook is rainy

Actions:

Start the tree with Outlook

Test windy when outlook is rainy

Example Decision Tree 2



Occam's Razor

- Principle stated by William of Ockham

  "Other things being equal, simple theories are preferable to complex ones"

  Informally,

  "Keep it simple, stupid!!"
- This has been the guiding principle for developing scientific theories
- Applied to our two decision trees describing the weather data
  - Decision tree 2 is preferable to decision tree 1
- Small decision trees are better
  - Attribute ordering makes the difference

Attribute Selection for Splitting

- In our case, outlook is a better attribute at the root than others
  - Because when outlook is at the root, one of its branches (overcast) immediately leads to a 'pure' daughter node which terminates further tree growth
  - Pure nodes have the same class label for all the instances reaching that node
- We need a generic function that helps to rank order attributes
  - The function should reward attributes that produce 'pure' daughter nodes

Entropy

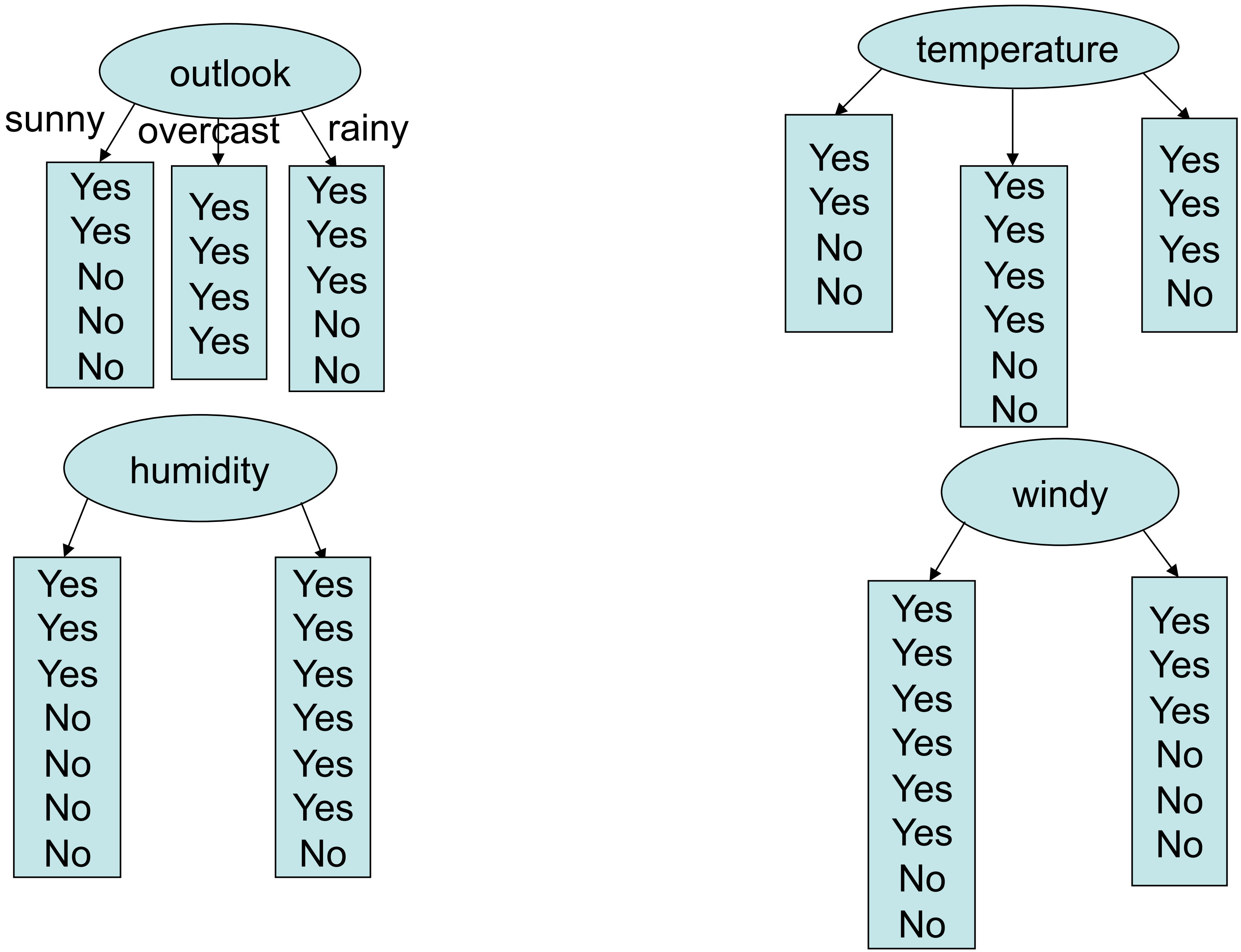
- Entropy is a measure of purity expressed in bits
- For a node in the tree, it represents the expected amount of information that would be needed to specify the class of a new instance that reached the node
- **Entropy(p_1,p_2,…,p_n) = $-p_1\log p_1 - p_2\log p_2 \ldots - p_n\log p_n$**
- Logarithms are computed for base 2 because we want the entropy measure in bits
- **p1, p2, … pn are fractions that sum up to 1**
- Because logarithms of fractions are negative, minus signs are used in the above formula to keep the entropy measure positive
- For the weather data
  - p1 = fraction of instances with play is true = 9/14
  - p2 = fraction of instances with play is false = 5/14
  - entropy(9/14,5/14) = -9/14log9/14 – 5/14log5/14

  = -9/14(log9 – log14) -5/14(log5 – log14)

  = -9/14log9 + 9/14log14 – 5/14log5 +5/14log14

  =-9/14log9 -5/14log5 +14/14log14 = (-9log9 -5log5 +14log14)/14 = 0.940 bits (fractions of bits allowed!!)

Tree stumps for the weather data



Entropy for the outlook stump

- Count the numbers of yes and no classes at the leaf nodes
  - [2,3], [4,0] and [3,2]
- Compute the entropy for each branch of the stump
  - Entropy(2/5,3/5) = 0.971 bits
  - Entropy(4/4,0/4) = 0.0 bits
  - Entropy(3/5,2/5) = 0.971 bits
- Compute the entropy for the whole stump
  - Entropy([2,3],[4,0],[3,2]) = 5/14* Entropy(2/5,3/5) + 4/14* Entropy(4/4,0/4) + 5/14* Entropy(3/5,2/5)

    =5/14*0.971+4/14*0+5/14*0.971

    =0.693 bits
- Represents the information needed in bits to specify the class for a new instance using this stump

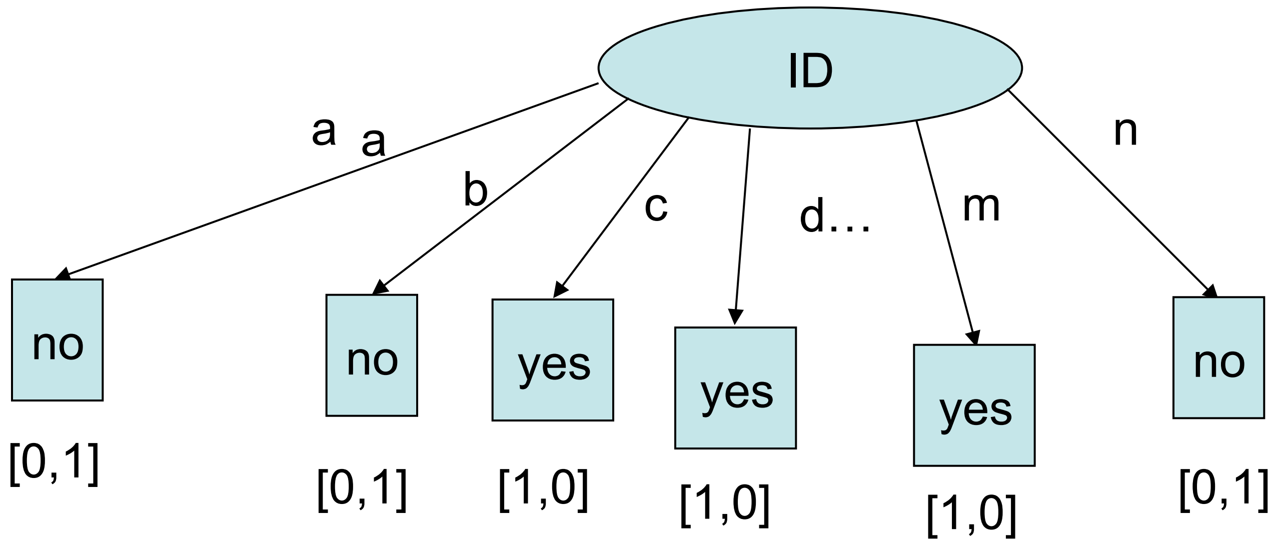
Information Gain

- When the attribute outlook was not considered, the weather data set has an entropy of 0.940 bits (as computed on slide 10)
- Entropy for the outlook stub is 0.693 bits (as computed on the previous slide)
- We made a saving of (0.940 – 0.693) bits on the information needed to specify the class of a new instance using this stump
  - This is the informational value of creating the outlook node
  - Gain(outlook) = (0.940 – 0.693) bits = 0.247 bits
- Similar computations for other stumps give us
  - Gain(temperature) = 0.029 bits
  - Gain(humidity) = 0.152 bits
  - Gain(windy)=0.048
- Because information gain is maximum for outlook, it should be selected for the root node
- Continuing with the above procedure builds the example decision tree 2
- ID3 algorithm uses information gain with the recursive procedure described earlier

Weather Data with ID

ID	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	false	no
b	sunny	hot	high	true	no
c	overcast	hot	high	false	yes
d	rainy	mild	high	false	yes
e	rainy	cool	normal	false	yes
f	rainy	cool	normal	true	no
g	overcast	cool	normal	true	yes
h	sunny	mild	high	false	no
i	sunny	cool	normal	false	yes
j	rainy	mild	normal	false	yes
k	sunny	mild	normal	true	yes
l	overcast	mild	high	true	yes
m	overcast	hot	normal	false	yes
n	rainy	mild	high	true	no

Tree Stump for the ID attribute



Entropy([0,1],[0,1],[1,0],[1,0] .....[1,0],[0,1]) =
1/14*Entropy(0/1,1/1)+1/14*Entropy(0/1,1/1) +
1/14*Entropy(1/1,0/1)+ 1/14*Entropy(1/1,0/1)+ …+
1/14*Entropy(1/1,0/1)+1/14*Entropy(0/1,1/1) =
1/14*0 + 1/14*0 + 1/14*0 …1/14*0 = 0

Gain(ID) = 0.940 – 0 = 9.940 bits

Highly branching attributes

- Weather data with ID has different id values for each instance
  - Knowing the id value is enough to predict the class
  - Therefore entropy for the stump with this attribute would be zero
    - Gain is high (0.940) and therefore this attribute will be selected first for tree construction
  - The tree constructed would be useless
    - Cannot predict new instances
    - Tells nothing about structure of the decision
- The above situation arises whenever an attribute leads to high branching
- A split always results in entropy reduction
  - Because a split reduces the number of classes
- Information gain does not account for this intrinsic information of a split

Gain Ratio

- Gain ratio is a measure used to adjust for the intrinsic information of a split
- Intrinsic information of a split is computed using the number and size of the daughter nodes produced by the split
  - Without taking the class information into account
- Intrinsic Information for the ID stump
  - Entropy([1,1,1,…,1])=14*(-1/14log1/14)=3.807 bits
- Gain Ratio = Information Gain/Intrinsic Information
- Gain ratio for the ID stump = 0.940/3.807=0.247
- Gain ratios for other stumps are
  - GainRatio(outlook) = 0.157, GainRatio(temperature)=0.019
  - GainRatio(humidity) = 0.152, GainRatio(windy) = 0.049
- In this case, ID still wins but not by a huge margin
- Additional measures used to guard against such useless attributes

Gain Ratio 2

- Gain ratio might overcompensate for intrinsic information
  - Because humidity (0.152) splits the data only into two branches, its GainRatio is nearly equal to that of outlook (0.157)
- Possible to fix this by choosing an attribute
  - with high gain ratio and
  - has InfoGain equal to the average of the InfoGains for all the attributes
- C4.5 (j4.8 in Weka) uses GainRatio and is an improvement over ID3

Summary so far

- Attribute selection for splitting achieved using measures of 'purity'
  - Information Gain
  - Gain Ratio
  - Etc.
- Issues related to decision tree construction
  - Numerical Attributes
  - Missing values
  - Overfitting and Pruning