COMP 9517 Computer Vision

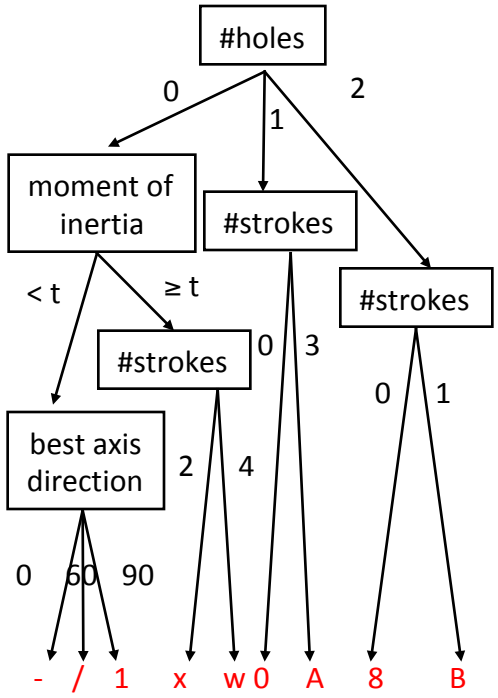
Pattern Recognition (2)

Decision Trees

- Introduction
  - Most practical pattern recognition methods address problem where feature vectors are real-valued and there exists some notion of metric
  - There are classification problems involving ***nominal data*** where instance descriptions are discrete and without any natural notion of similarity or even ordering
    - For example: {high, medium, low}, {red, green, blue}
  - How can we use such nominal data for classification?
    - Use rule-based or syntactic pattern recognition methods

Decision Trees

- Approach
  - To classify a pattern through a sequence of questions
  - A sequence of questions in a directed ***decision tree*** or simply ***tree***
- Decision Tree Overview
  - Structure
    - Nodes in the tree represent features
    - Leaf nodes contain the class labels
    - One feature (or a few) at a time to split search space of patterns
    - Each branching node has one child for each possible value of the parent feature
  - Classification
    - Begins at the root node, follows the appropriate link to a leaf node
    - Assigns the class label of the leaf node to the test pattern



Decision Trees

- Construction of Decision Tree
  - Binary decision tree is a binary tree structure that has a decision function associated with each node
  - Simple case: numeric feature values, decision function compares value of a feature to a threshold. The decision function selects left/right branch if the value of the feature is less / greater than the threshold
  - Advantages: at each node, only feature to be used and threshold value need be stored
  - For any given set of training examples, there may be more than one possible decision tree to classify them
  - We must select features that give the `best' tree based on some criterion
  - Smallest tree preferred

Decision Trees

- Construction of Decision Tree
  1. Select a feature to place at the node (the first one is the root)
  2. Make one branch for each possible value
  3. For each branch node, repeat step 1 and 2 using only those instances that actually reach the branch
  4. When all instances at a node have the same classification, stop developing that part of the tree
- How to determine which feature to split on?
  - One way is to use measures from information theory
  - ***Entropy*** and ***Information Gain***

Decision Tree

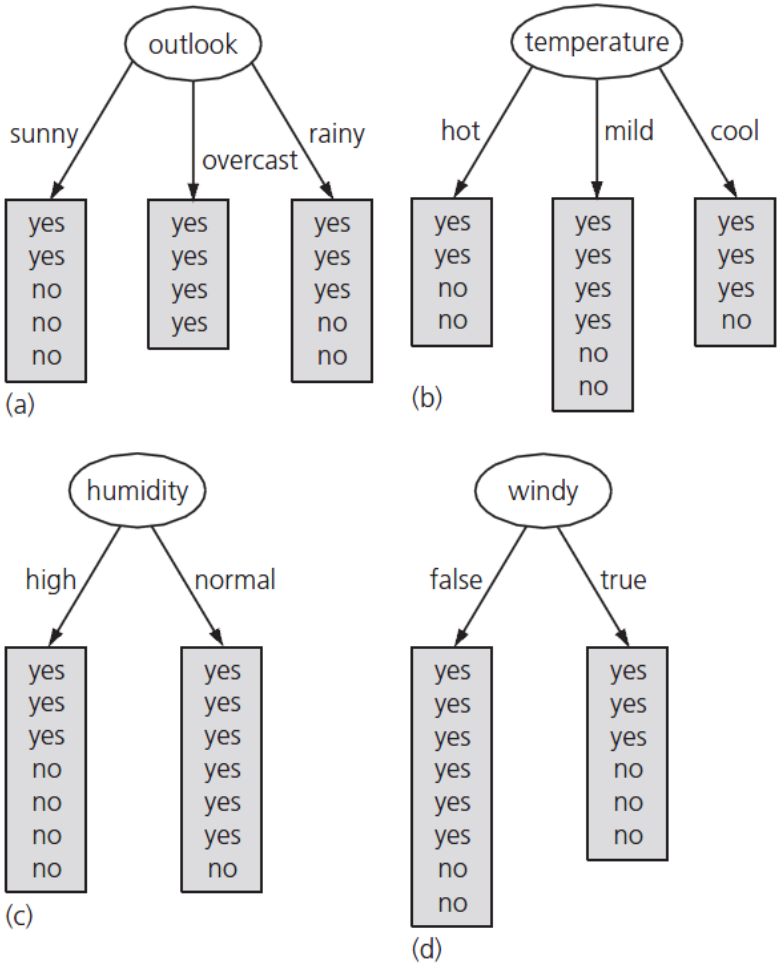
- Entropy
  - To construct optimal decision trees from training data, we need a definition of optimality
  - One simple criterion is ***entropy***, based on information theory

    The entropy of a set of events x = {x_1, x_2, ..., x_n}

$$H(x) = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i)$$

    where $p(x_i)$ is the probability of event x_i
  - Entropy may be viewed as the average uncertainty of the information source.

Decision Tree



- Entropy before Split

$$Entropy([9+,5-]) = 0.940$$

- Entropy after Split

$$Entropy([2+,3-]) = 0.971$$

$$Entorpy([4+,0-]) = 0.0$$

$$Entorpy([3+,2-]) = 0.971$$

$$Entropy([2+,3-],[4+,0-],[3+,2-]) = 0.693$$

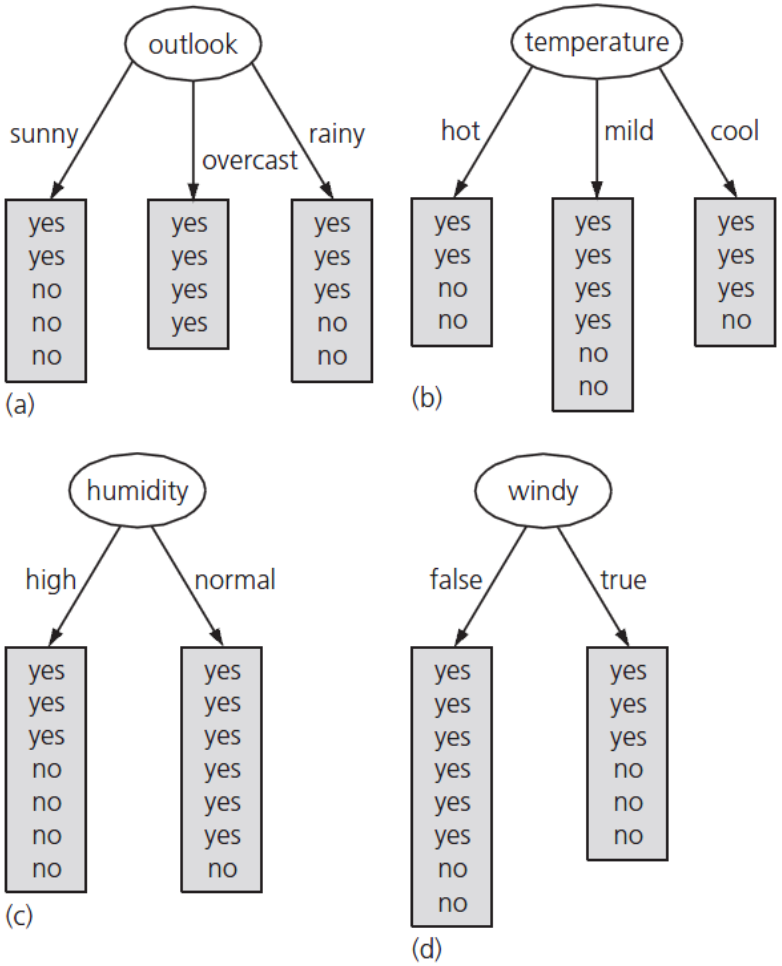
Decision Tree

- Information Gain
  - *Information gain* is an entropy-based measure to evaluate features and produce optimal decision trees
  - The information gain *G (C, F)* of class variable *C* relative to a collection of example S, is defined as

$$Gain(C,F) = Entorpy(S) - \sum_{f \in F} \frac{|S_f|}{|S|} Entropy(S_f)$$

  - Use the feature with highest info gain to split on
    - Prior probabilities can be estimated using frequency of associated events in the training data

Decision Tree



- Entropy before Split

  $Entropy([9+,5-]) = 0.940$

- Entropy after Split

  $Entropy([2+,3-]) = 0.971$

  $Entorpy([4+,0-]) = 0.0$

  $Entorpy([3+,2-]) = 0.971$

  $Entropy([2+,3-],[4+,0-],[3+,2-]) = 0.693$

- Gain

  $Gain(outlook) = 0.247$

  $Gain(temperature) = 0.029$

  $Gain(humidity) = 0.152$

  $Gain(windy) = 0.048$

- So select "outlook"

Ensemble Learning

- Ensemble learning uses multiple models to improve predictive performance from those obtained from any of the constituent models
- Multiple models can be created
  - by different classifiers/learning algorithm
  - by different parameters for the same algorithm
  - by different training examples

Random Forests

- Random forests are an ensemble learning method that constructs an ensemble of decision trees by training and outputs the class that is the mode of the classes output by individual trees
- The Breiman's algorithm:
  - Training
    - Let the number of training instances be N and the number of features be M
    - Sample N instances at random with replacement from the original data
    - At each node, m<<M features are selected at random out of the M and the best split on these m is used to split the node (the value of m is held constant during the forest growing)
    - Each tree is grown to the largest extend possible (no pruning)
  - Testing
    - A new samples is pushed down the tree, which is assigned the label of the training samples in the terminal node it ends up in
    - Iterated over all trees in the ensemble and the mode vote of all trees is reported as the random forest prediction

Random Forests

- The forest error rate depends on two things
  - The correlation between any two trees in the forest
    - Increasing the correlation increases the forest error rate
  - The strength of each individual tree in the forest
    - A stronger tree has low error rate
    - Increasing the strength of the individual trees decreases the forest error rate
- Parameter m
  - Reducing m reduce both the correlation and the strength while increasing it increases both
  - Somewhat in between is an “optimal” range of m

Random Forests

- Main features of Random Forests
  - Unexcelled in accuracy among current algorithms
  - Efficiently on large datasets
  - Handling thousands of input features without feature selection
  - Handling missing value effectively

References and Acknowledgements

- Shapiro and Stockman, Chapter 4
- Duda, Hart and Stork, Chapter 1
- More references
  - Sergios Theodoridis, Konstantinos Koutroumbas, *Pattern Recognition*, 2009
  - Ian H. Witten, Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2005
- Some content are extracted from the above resources