

CS4770

Pattern Recognition

Non-Metric Methods

P. J. Narayanan
Monsoon 2006

Why?

- Feature values are often non-numerical, with no natural notion of “similarity”.
- A fruit is characterized by its shape, size, colour, and taste.
- Thus, mango has a feature vector: `[elongated, medium, yellow, sour]` and watermelon: `[round, large, green, sweet]`.
- All methods we studied needed a notion of distance. How do we classify using such nominal features?
- Feature vector is a tuple of d components.

Decision Trees

- A tree starting with a root node. A question is asked about the feature vector at each node.
- Each distinct answer leads to a different child.
- Leaf node holds a category to which the feature vector is classified.
- Such trees are known as **Decision Trees**.
- Nodes at the same level could ask different questions.

- Root node: **shape?**. round, long, oval lead down different paths.
- Next question in node round: **size?**. small, medium, large lead down different paths.
- Question in node large: **colour?**. green, yellow point down. green may reach a leaf node of watermelons. yellow may need more questions.

Building Decision Trees

- Construct decision trees from given input data.
- **CART**: Classification and Regression Trees.
- Questions:
 1. Binary split or multivalued splits?
 2. Which property to use for splitting?
 3. How to recognize leaf nodes?
 4. How to prune a large tree?
 5. What is the category label for a leaf?

Splitting

- Multiway split if multiple values for the property used.
- All can be converted to binary trees. Easier.
- Use a property that would make nodes most “**pure**”.
- **Entropy** or **Information** impurity for a node N :
$$i(N) = - \sum_i P(\omega_i) \log_2 P(\omega_i) \text{ for all } \omega_i.$$
- Variance impurity for two classes: $P(\omega_i)P(\omega_j)$

- Gini impurity: (expected error rate if category is random)

$$i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = \frac{1}{2} \left[1 - \sum_j P(\omega_j)^2 \right]$$

- Choose a split that maximizes $\Delta i(N) = i(N) - P_l i(N_l) - P_r i(N_r)$ where P_l, P_r are fractions of samples
- Use of gradient descent for minimum purity impossible!
- For real-valued features, a linear classifier can be used for splitting!

- $\Delta_B i(N) = i(N) - \sum_{k=0}^{k=B} P_k i(N_k)$ for B -way splits. This favours large B .
- Thus use *gain ratio impurity* given by

$$\Delta_B i(N) = \frac{i(N) - \sum_{k=0}^{k=B} P_k i(N_k)}{\sum_k -P_k \log_2 P_k}$$

- These greedy methods may not ensure global optimum.

Stopping Criterion

- Overfitting if split continues for too long.
- When each node is pure enough
- When change in purity small enough
- Number of elements in a node is small enough
- Validation error is small enough, etc.

Pruning or Merging

- Split till leaf nodes have minimum impurity.
Prune the tree by merging later.
- Examine nodes with a common ancestor in pairs.
- Merge if increase in impurity is small enough!

Choice of Features

- If monothetic (only one feature component in each question), boundaries will be parallel to axes.
- Could be inefficient. A staircase approximation to complex boundary results.
- If polythetic, linear boundaries can result. Complex boundaries can emerge.

Classification

- Go till a leaf node. If it has only one class label, assign that to the test vector.
- Otherwise, assign the most dominant label.
- Specific tree methods: **ID3**, **C4.5**

Structural Pattern Recognition

- Many patterns contain structural and relational information
- Too difficult to represent as feature vectors or process statistically
- Problem structure is described in terms of grammars, graphs etc.
- Classification using parsing, graph-matching etc.

String Matching

- DNA structure and many others can be expressed as strings.
- Finding a matching substring is an important problem.
- Matching, edit distance, matching with errors, matching with don't care symbols etc.
- Regular expressions

Syntactic Pattern Recognition

- Describe patterns/classes in terms of rules for generating them.
- Describe these as formal grammars.
- Non-terminals are equivalent to features from the world.
- Parse the inputs. If successful, recognize.

Grammars, Types

- $G = (T, A, S, P)$. Terminals, non-terminals, starting symbol, production or rewrite rules.
- Language generated by the grammar: $L(G)$. All strings accepted by it.
- Type 0: Unrestricted grammar. Turing machine.
- Type 1: Context-sensitive grammar. Linear-bounded automata. (Turning machine of limited tape).

- Type 2: Context-free grammar. Push-down automata. (A stack as memory).
- Type 3: Regular grammar. Finite automata. (Finite memory).

Parsing

- Bottom-up parsing: CYK algorithm for CFG.
- Top-down parsing.
- Well developed tools.

Example: Staircase

- Horizontal and vertical rectangles detected.
- $S \rightarrow hv|hvS$
- Add landing and changing of direction.
- $S \rightarrow A|SlS \quad A \rightarrow hv|hvA$

Inferring Grammars

- Start with the starting state. Terminal symbols are known.
- Initialize to random production, say, $S \rightarrow A$.
- Need positive and negative training examples. Strings generated by G and not generated by it!
- If the existing production rules cannot parse the next string, add a rule if:
 - (a) it derives the new string, and
 - (b) it does not derive any negative string!

- Eliminate redundant rewrite rules periodically.
- Type of grammar, naturally, is fixed upfront!

Stochastic grammar

- Each production rule has a probability associated with it.
- Starting symbol is replaced by a starting probability on all non-terminal symbols.
- Probabilities for a NT symbol may sum to 1.
- A sentence is generated by a grammar with a probability p . This is a product of probabilities of the starting symbol and the production rules used (if independent).

Graph Theoretic Methods

- A directed **relational graph** to represent relations between elements.
- Graph isomorphism, subgraph matching, etc., for classification.

Which Classifier is Good?

- Is one classifier provably better than others?
- **No Free-Lunch Theorem:** Without assumptions on the data/problem, no classifier is better than others or a random decision!
- **Ugly-Duckling Theorem:** There is no “best” set of features or sets of features without assumptions on the data/problem.