CS 478 - Tools for Machine Learning and Data Mining

Symbolic Clustering - COBWEB

COBWEB Overview

- Symbolic approach to category formation.
- Uses global quality metrics to determine number of clusters, depth of hierarchy, and category membership of new instances.
- Categories are probabilistic. Instead of category membership being defined as a set of feature values that must be matched by an object, COBWEB represents the probability with which each feature value is present.
- Incremental algorithm. Any time a new instance is presented, COBWEB considers the overall quality of either placing it in an existing category or modifying the hierarchy to accommodate it.

Category Utility

$$CU = \sum_k \sum_j \sum_i P(F_i = v_{ij}) P(F_i = v_{ij} \mid C_k) P(C_k \mid F_i = v_{ij})$$

- $P(F_i = v_{ij} \mid C_k)$ is called the *predictability*. It is the probability that an object has value v_{ij} for feature F_i given that the object belongs to category C_k. The greater this probability, the more likely two objects in a category share the same features.
- $P(C_k \mid F_i = v_{ij})$ is called the *predictiveness*. It is the probability with which an object belongs to category C_k given that it has value v_{ij} for feature F_i. The greater this probability, the less likely objects not in the category will have those feature values.
- $P(F_i = v_{ij})$ serves as a weight. It ensures that frequently-occurring feature values exert a stronger influence on the evaluation.

CU maximizes the potential for inferring information while maximizing intra-class similarity and inter-class differences.

Tree Representation

- Each node stores:
  1. Its probability of occurrence, $P(C_k)$ (= num. instances at node / total num. instances)
  2. All possible values of every feature observed in the instances, and for each such value, its predictability.
  3. Predictiveness is computed using Bayes rule (i.e., $P(A \mid B) = \frac{P(A)P(B|A)}{P(B)}$.
- Leaf nodes correspond to observed instances.
- All links are "is-a" links (i.e., no test on feature values).
- Tree is initialized with a single node whose probabilities are those of the first instance.
- For each subsequent instance I, Cobweb($Root$, I) is invoked.

COBWEB Algorithm

Algorithm Cobweb(*Node*, *Instance*)
  If *Node* is a leaf
    Create 2 children, L_1 and L_2 of *Node*
    Set the probabilities of L_1 to those of *Node*
    Initialize the probabilities of L_2 to those of *Instance*
    Add *Instance* to *Node*, updating *Node*'s probabilities
  Else
    Add *Instance* to *Node*, updating *Node*'s probabilities
    For each child C of *Node*
      Compute CU of taxonomy obtained by placing *Instance* in C
    Let S_1 be the score of the best categorization C_1
    Let S_2 be the score of the next best categorization C_2
    Let S_3 be the score of placing *Instance* in a new category
    Let S_4 be the score of merging C_1 and C_2 into one category
    Let S_5 be the score of splitting C_1
    If S_1 is the best score
      Cobweb(C_1, *Instance*)
    Else if S_3 is the best score
      Initialize new category's probabilities to those of *Instance*
    Else is S_4 is the best score
      Let C_m be the result of merging C_1 and C_2
      Cobweb(C_m, *Instance*)
    Else if S_5 is the best score
      Split C_1
      Cobweb(*Node*, *Instance*)
    Else {possible default if C_2 exists}
      Cobweb(C_2, *Instance*)

Demo

http://www-ai.cs.uni-dortmund.de/kdnet/auto?self=$81d91eaae317b2bebb

Discussion

- Nice probabilistic model with no parameters set a priori.
- Only handles nominal features (CLASSIT extends to numerical).
- Sensitive to order of presentation of instances.
- Retains each instance, which may cause problems with noisy data.