Unsupervised Concept Induction

- The vast majority of research in ML has dealt with supervised tasks.

  Given: attribute-value pairs that describe an object or observation
  Predict: class value

- **Flexible prediction:**

  Given: attribute-value pairs, but no knowledge of which are predictors and which are to be predicted
  Predict: any feature from any others
  Performance measure: ???

Slide CS478–1

Algorithms for Flexible Prediction

- Nearest-neighbor
- Transform supervised method:
  - Given k attributes, run the supervised algorithm k times, in each case with a different feature playing the role of the class attribute.
  - Produces k classifiers, each designed to predict one attribute as a function of the others.
- Neural network solutions
- Clustering

Slide CS478–2

Learning Association Rules

basket data: each record consists of the **transaction date** and the **items bought**.

Goal: mine association rules from market basket data.

Sample rule: *98% of customers that purchase tires and auto accessories also get automotive services done.*

Definitions

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals called *items*.

Let D be a set of transactions where each transaction $T \subseteq I$.

A transaction T *contains* X, a set of some items in I, if $X \subseteq T$.

An *association rule* is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$.

$X \Rightarrow Y$ holds in D with *confidence* c if c% of transactions in D that contain X also contain Y.

$X \Rightarrow Y$ holds in D with *support* s if s% of transactions in D contain $X \cup Y$.

Example

$D = \{$
$\{x, y\}$
$\{w, z\}$
$\{x, y\}$
$\{a, z\}$
$\{x\}$
$\{b, z\}$
$\{a, x\}$
$\{c, z\}$
$\{y\}$
$\{d, z\}\}$

Learning Problem

Given a set of transactions D, the problem of mining association rules is to generate all association rules that have support and confidence greater than the user-specified minimum support (*minsup*) and minimum confidence (*minconf*).

High-Level Algorithm

1. Find all sets of items (*itemsets*) that have transaction support above *minsup*.
   - Itemsets with minimum support are called *large* itemsets.
   - All others are called *small* itemsets.
2. Use the large itemsets to generate the desired rules.
   - For every large itemset l, find all non-empty subsets of l.
   - For every such subset a, output a rule of the form $a \Rightarrow (l - a)$ if its confidence is at least *minconf*.

Discovering Large Itemsets

- Make multiple passes over the data.
- Pass 1: count the support of individual items; determine which of them are *large*.
- Subsequent passes: Use the large itemsets from the previous pass to generate new potentially large itemsets, called *candidate* itemsets; count the actual support for these candidate itemsets and remove those below minsup.
- Continue until no new large itemsets are found.

An Algorithm for Discovering Large Itemsets

L_1 = { large 1-itemsets };
for (k=2; $L_{k-1} \neq \emptyset$; k++) do
 C_k= gen-new-candidates(L_{k-1});
 forall transactions $t \in D$ do
  $C_t = subset(C_k, t)$; //candidates contained in t
  forall candidates $c \in C_t$ do
   c.count++;
 $L_k = \{c \in C_k | \frac{c.count}{|D|} \geq minsup\}$
Return ($\bigcup_k L_k$);

Slide CS478–9

Generating New Candidates

GEN-NEW-CANDIDATES (L_{k-1})
Read each transaction t.
 -Determine which of the large itemsets in L_{k-1} are present in t.
 -Extend each such itemset l with all those large items that are present in t and occur later in the lexicographic ordering than any of the items in l.
 -Save these extensions in C.
 -Delete all itemsets $c \in C$ such that some (k-1)-subset of c is not in L_{k-1}.
 -$C_k = C_k \cup C$.
ReturnC_k.

Slide CS478–10

Example

Assume $L_3 = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{1,3,5\},\{2,3,4\}\}$.

GEN-NEW-CANDIDATES (L_{k-1}):

in response to $t = \{1,2,3,4,5\}$, produces