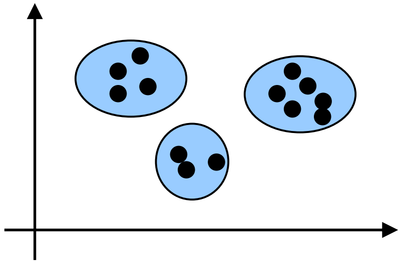
Data Mining

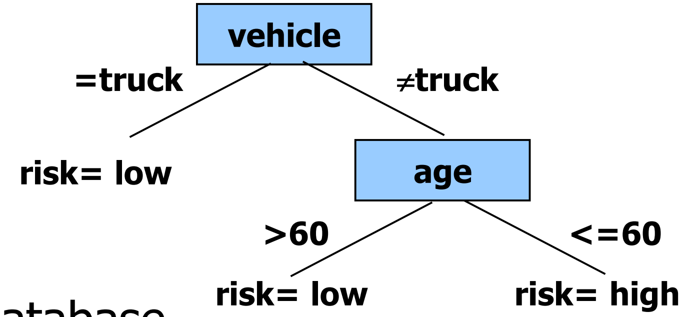
DIS Exercise Course

Data Mining

- Applying efficient algorithms for pattern detection in large datasets
- Clustering
  - Automatic identification of a finite set of categories, classes or groups (clusters)
  - Comparison using distance functions



- Classification
  - Goal: learning a classifier (e.g. a decision tree)
  - Classes are known, training data are available



- Association Rules
  - Shopping cart analysis on transactional database
  - Example: buys(PC) => buys(printer)

Association Rules (1)

- Given:
  - Set of possible articles: I
  - Multiset of transactions: $T \subseteq \mathrm{Bag}(\mathcal{P}(I))$

- Rules: $r \rightarrow k$ *[support, confidence]*
  with $r, k \subseteq I$ and $r \neq k$

- **Support**: share of transactions containing all objects r and k

$$\frac{|\{t \in T : r \cup k \subseteq t\}|}{|T|}$$

- **Confidence**: share of transactions containing r and following the rule

$$\frac{|\{t \in T : r \cup k \subseteq t\}|}{|\{t \in T : r \subseteq t\}|}$$

Association Rules (2)

- Example:
  - I = {Beer, Cigarettes, Coke, Peanuts, Chips}
  - T= {{Beer, Coke, Peanuts}, {Beer, Chips, Cigarettes}, {Beer, Chips, Cigarettes, Coke}, {Beer, Cigarettes}}

TAID	Items
001	Beer, Coke, Peanuts
002	Beer, Chips, Cigarettes
003	Beer, Chips, Cigarettes, Coke
004	Beer, Cigarettes

Support(Beer) = 100%
Support(Chips) = 50%
…
Support(Beer, Coke) = 50%
Support(Beer, Peanuts) = 25%
…
Support(Beer, Coke, Peanuts) = 25%
Support(Beer, Chips, Cigarettes) = 50%
…

Beer $\rightarrow$ Coke: Confidence = 50%
Coke $\rightarrow$ Beer: Confidence = 100%
…

Apriori Algorithm (1)

- **Frequent itemset**: itemset with support greater than threshold s
- Determining frequent itemsets is important for deriving association rules
- Efficient realisation via Apriori algorithm
- Exploitation of **Apriori property**:
  - Every subset of a frequent itemset is a frequent itemset itself
  - Support of no subset of a frequent itemset can be smaller than threshold s
- Efficient iterative implementation starting with 1-itemsets
  - Iterative evaluation of k-itemsets
  - Discard combinations containing itemsets with support less than s

Apriori Algorithm (2)

L_1 = find_frequent_1_itemsets(); //initial 1-itemsets

for(k=2; $L_{k-1} \neq \emptyset$; k++) {
  C_k = *generateCandidates* (L_{k-1}); //all L_{k-1} possible k-itemsets

  for each transaction t {
    for each candidate c $\in C_k$ {
      if (t contains c)
        c.count++;
    }
  }
  L_k = {c $\in C_k$ | c.count >= **MIN_SUP**}
}

Apriori Algorithm (3)

```
//returns all L_{k-1} possible k-itemsets
procedure generateCandidates(L_{k-1}) {
    for each itemset l_1 ∈ L_{k-1} {
        for each itemset l_2 ∈ L_{k-1} {
            if(l_1[1..k-2] == l_2[1..k-2] && l_1[k-1] < l_2[k-1]) {
                c = l_1[1..k-1],l_2[k-1];
                //are all (k-1)-item subsets frequent itemsets?
                if(!prune(c, L_{k-1}))
                    C_k.add(c);
            }
        }
    }
    return C_k;
}
```

Apriori Algorithm (4)

```
//checks whether at least one of the (k-1)-subsets of c is not in L_{k-1}
procedure prune(c, L_{k-1}) {
    for each (k-1)-subset s of c {
        if(s ∉ L_{k-1})
            return true;
    }
    return false;
}
```

Apriori Algorithm: Example (1)

TAID	Items
001	I1, I2, I5
002	I2, I4
003	I2, I3
004	I1, I2, I4
005	I1, I3
006	I2, I3
007	I1, I3
008	I1, I2, I3, I5
009	I1, I2, I3

MIN_SUP = 2

k=1: C_1

Itemset	Sup #
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

compare with MIN_SUP →

L_1

Itemset	Sup #
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

k=2: C_2

Itemset	Sup #
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

no pruning for k=2, since all subsets are frequent

compare with MIN_SUP →

L_2

Itemset	Sup #
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Apriori Algorithm: Example (1)

L_2

Itemset	Sup #
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

k=3: C_3

	Itemset
	{I1, I2, I3}
	{I1, I2, I5}
{I3, I5} $\notin L_2$	~~{I1, I3, I5}~~
{I3, I4} $\notin L_2$	~~{I2, I3, I4}~~
{I3, I5} $\notin L_2$	~~{I2, I3, I5}~~
{I4, I5} $\notin L_2$	~~{I2, I4, I5}~~

count support →

C_3

Itemset	Sup #
{I1, I2, I3}	2
{I1, I2, I5}	2

↓ compare with MIN_SUP

L_3

Itemset	Sup #
{I1, I2, I3}	2
{I1, I2, I5}	2

k=4: C_4

	Itemset
{I2, I3, I5} $\notin L_3$	~~{I1, I2, I3, I5}~~

Creating Association Rules

- Based on frequent itemsets
    1. For every frequent itemset l, create all subsets
    2. For every subset s of l, create rule **s → (l – s)**, if **confidence(s → (l – s)) > MIN_CONF**