

## Data preprocessing

- Data cleaning: fill missing values, smoothing of the data (binning method), identify outliers, resolve inconsistencies/contradictions. To find outliers use clustering.
- Data integration: Put together multiple databases.
- Data transformation: normalize data, create new attributes.
- Data reduction: represent the data in a reduce format that produces similar analytical results. For example: remove unimportant attributes or cluster data and take a class representative or remove randomly some data records. Simple random removal may have poor performance in the presence of skewed data. Stratified sampling: cluster data, find the percentage of the data in each class and sample using keeping in mind the percentage.
- Binning method, entropy based method. Binning method means partitioning data into bins and smooth each bin by taking the mean/mode/median/boundary...

**Association rule mining** Rule strength measure

$$\text{support}(X \rightarrow Y) = \frac{\text{count}(X \cup Y)}{n}$$

$$\text{confidence}(X \rightarrow Y) = \frac{\text{count}(X \cup Y)}{\text{count}(X)}$$

## 1 Apriori frequent itemset algorithm

init-pass obtains the items in lexicographic order.

### Algorithm Apriori( $T$ )

```
 $C_1 \leftarrow \text{init-pass}(T);$ 
 $F_1 \leftarrow \{f \mid f \in C_1, f.\text{count}/n \geq \text{minsup}\};$  //  $n$ : no. of transactions in  $T$ 
for ( $k = 2$ ;  $F_{k-1} \neq \emptyset$ ;  $k++$ ) do
     $C_k \leftarrow \text{candidate-gen}(F_{k-1});$ 
    for each transaction  $t \in T$  do
        for each candidate  $c \in C_k$  do
            if  $c$  is contained in  $t$  then
                 $c.\text{count}++;$ 
            end
        end
     $F_k \leftarrow \{c \in C_k \mid c.\text{count}/n \geq \text{minsup}\}$ 
end
return  $F \leftarrow \bigcup_k F_k;$ 
```

```

Function candidate-gen( $F_{k-1}$ )
   $C_k \leftarrow \emptyset$ ;
  forall  $f_1, f_2 \in F_{k-1}$ 
    with  $f_1 = \{i_1, \dots, i_{k-2}, i_{k-1}\}$ 
    and  $f_2 = \{i_1, \dots, i_{k-2}, i'_{k-1}\}$ 
    and  $i_{k-1} < i'_{k-1}$  do
       $c \leftarrow \{i_1, \dots, i_{k-1}, i'_{k-1}\}$ ;           // join  $f_1$  and  $f_2$ 
       $C_k \leftarrow C_k \cup \{c\}$ ;
      for each  $(k-1)$ -subset  $s$  of  $c$  do
        if ( $s \notin F_{k-1}$ ) then
          delete  $c$  from  $C_k$ ;           // prune
      end
    end
  end
  return  $C_k$ ;

```

## 2 MS Apriori

sort in ascending

```

Algorithm MSapriori( $T, MS, \phi$ ) //  $\phi$  is for support difference constraint
   $M \leftarrow \text{sort}(I, MS)$ ;
   $L \leftarrow \text{init-pass}(M, T)$ ;
   $F_1 \leftarrow \{\{i\} \mid i \in L, i.\text{count}/n \geq \text{MIS}(i)\}$ ;
  for ( $k = 2; F_{k-1} \neq \emptyset; k++$ ) do
    if  $k=2$  then
       $C_k \leftarrow \text{level2-candidate-gen}(L, \phi)$ 
    else  $C_k \leftarrow \text{MSCandidate-gen}(F_{k-1}, \phi)$ ;
    end;
    for each transaction  $t \in T$  do
      for each candidate  $c \in C_k$  do
        if  $c$  is contained in  $t$  then
           $c.\text{count}++$ ;
          if  $c - \{c[1]\}$  is contained in  $t$  then
             $c.\text{tailCount}++$ 
          end
        end
       $F_k \leftarrow \{c \in C_k \mid c.\text{count}/n \geq \text{MIS}(c[1])\}$ 
    end
  return  $F \leftarrow \bigcup_k F_k$ ;

```

Figure 1: MsApriori

data set  $T$  and the sorted items  $M$ , to produce the seeds  $L$  for generating candidate itemsets of length 2, i.e.,  $C_2$ . **init-pass()** has two steps:

1. It first scans the data once to record the support count of each item.
2. It then follows the sorted order to find the first item  $i$  in  $M$  that meets  $\text{MIS}(i)$ .  $i$  is inserted into  $L$ . For each subsequent item  $j$  in  $M$  after  $i$ , if  $j.\text{count}/n \geq \text{MIS}(i)$ , then  $j$  is also inserted into  $L$ , where  $j.\text{count}$  is the support count of  $j$ , and  $n$  is the total number of transactions in  $T$ .

Figure 2: Init Pass

```

Function level2-candidate-gen( $L, \phi$ )
  1  $C_2 \leftarrow \emptyset$ ; // initialize the set of candidates
  2 for each item  $i$  in  $L$  in the same order do
  3   if  $i.\text{count}/n \geq \text{MIS}(i)$  then
  4     for each item  $h$  in  $L$  that is after  $i$  do
  5       if  $h.\text{count}/n \geq \text{MIS}(i)$  and  $|\text{sup}(h) - \text{sup}(i)| \leq \phi$  then
  6          $C_2 \leftarrow C_2 \cup \{\{i, h\}\}$ ; // insert the candidate  $\{i, h\}$  into  $C_2$ 

```

Fig. 2.7. The level2-candidate-gen function

```

Function MSCandidate-gen( $F_{k-1}, \phi$ )
  1  $C_k \leftarrow \emptyset$ ; // initialize the set of candidates
  2 forall  $f_1, f_2 \in F_k$  // find all pairs of frequent itemsets
  3   with  $f_1 = \{i_1, \dots, i_{k-2}, i_{k-1}\}$  // that differ only in the last item
  4   and  $f_2 = \{i_1, \dots, i_{k-2}, i'_{k-1}\}$ 
  5   and  $i_{k-1} < i'_{k-1}$  and  $|\text{sup}(i_{k-1}) - \text{sup}(i'_{k-1})| \leq \phi$  do
  6      $c \leftarrow \{i_1, \dots, i_{k-1}, i'_{k-1}\}$ ; // join the two itemsets  $f_1$  and  $f_2$ 
  7      $C_k \leftarrow C_k \cup \{c\}$ ; // insert the candidate itemset  $c$  into  $C_k$ 
  8   for each  $(k-1)$ -subset  $s$  of  $c$  do
  9     if ( $c[1] \in s$ ) or ( $\text{MIS}(c[2]) = \text{MIS}(c[1])$ ) then
  10       if ( $s \notin F_{k-1}$ ) then
  11         delete  $c$  from  $C_k$ ; // delete  $c$  from the set of candidates
  12     endfor
  13 endfor
  14 return  $C_k$ ; // return the generated candidates

```

Fig. 2.8. The MSCandidate-gen function

Figure 3: Candidate Gen MS

## GSP mining algorithm

### ■ Very similar to the Apriori algorithm

**Algorithm GSP( $S$ )**

```

1   $C_1 \leftarrow \text{init-pass}(S);$  // the first pass over  $S$ 
2   $F_1 \leftarrow \{\langle \{f\} \rangle \mid f \in C_1, f.\text{count}/n \geq \text{minsup}\};$  //  $n$  is the number of sequences in  $S$ 
3  for ( $k = 2; F_{k-1} \neq \emptyset; k++$ ) do // subsequent passes over  $S$ 
4     $C_k \leftarrow \text{candidate-gen-SPM}(F_{k-1});$ 
5    for each data sequence  $s \in S$  do // scan the data once
6      for each candidate  $c \in C_k$  do
7        if  $c$  is contained in  $s$  then
8           $c.\text{count}++;$  // increment the support count
9        end
10     end
11      $F_k \leftarrow \{c \in C_k \mid c.\text{count}/n \geq \text{minsup}\}$ 
12 end
13 return  $\bigcup_k F_k;$ 

```

**Fig. 12.** The GSP Algorithm for generating sequential patterns

When he says is the same as the subsequence... Means that you have to remove the parenthesis and see if the elements are in the same order.

**Function candidate-gen-SPM( $F_{k-1}$ )**

1. **Join step.** Candidate sequences are generated by joining  $F_{k-1}$  with  $F_{k-1}$ . A sequence  $s_1$  joins with  $s_2$  if the subsequence obtained by dropping the first item of  $s_1$  is the same as the subsequence obtained by dropping the last item of  $s_2$ . The candidate sequence generated by joining  $s_1$  with  $s_2$  is the sequence  $s_1$  extended with the last item in  $s_2$ . There are two cases:
  - the added item forms a separate element if it was a separate element in  $s_2$ , and is appended at the end of  $s_1$  in the merged sequence, and
  - the added item is part of the last element of  $s_1$  in the merged sequence otherwise.

When joining  $F_1$  with  $F_1$ , we need to add the item in  $s_2$  both as part of an itemset and as a separate element. That is, joining  $\langle \{x\} \rangle$  with  $\langle \{y\} \rangle$  gives us both  $\langle \{x, y\} \rangle$  and  $\langle \{x\} \{y\} \rangle$ . Note that  $x$  and  $y$  in  $\{x, y\}$  are ordered.

2. **Prune step.** A candidate sequence is pruned if any one of its  $(k-1)$ -subsequence is infrequent (without minimum support).

**Fig. 13.** The candidate-gen-SPM() function

## 4 Index for Classifiers

Metric	Description / Formula
Accuracy	$\frac{\text{Correctly Classified Examples}}{\text{Total Examples}}$
Precision	$\frac{TP}{TP+FP}$ : Of predicted positives, how many are correct.
Recall/sensitivity (TPR)	$\frac{TP}{TP+FN}$ : Of actual positives, how many are identified.
$F_1$ Score	$2 \times \frac{p \cdot r}{p+r}$ : Combines precision and recall.
Specificity (TNR)	$\frac{TN}{TN+FP}$ : True negative rate.
False Positive Rate (FPR)	$1 - \text{TNR}$
ROC Curve	Plot: FPR (x-axis) vs. TPR (y-axis).

Table 1: Summary of Classification Metrics

## 5 Decision trees

```

Algorithm decisionTree( $D, A, T$ )
1  if  $D$  contains only training examples of the same class  $c_j \in C$  then
2    make  $T$  a leaf node labeled with class  $c_j$ 
3  elseif  $A = \emptyset$  then
4    make  $T$  a leaf node labeled with  $c_j$ , which is the most frequent class in  $D$ 
5  else //  $D$  contains examples belonging to a mixture of classes. We select a single
6    // attribute to partition  $D$  into subsets so that each subset is purer
7     $p_0 = \text{impurityEval-1}(D)$ ;
8    for each attribute  $A_i \in \{A_1, A_2, \dots, A_k\}$  do
9       $p_i = \text{impurityEval-2}(A_i, D)$ 
10   end
11   Select  $A_g \in \{A_1, A_2, \dots, A_k\}$  that gives the biggest impurity reduction,
     computed using  $p_0 - p_i$ ;
12   if  $p_0 - p_g < \text{threshold}$  then //  $A_g$  does not significantly reduce impurity  $p_0$ 
13     make  $T$  a leaf node labeled with  $c_j$ , the most frequent class in  $D$ .
14   else //  $A_g$  is able to reduce impurity  $p_0$ 
15     Make  $T$  a decision node on  $A_g$ ;
16     Let the possible values of  $A_g$  be  $v_1, v_2, \dots, v_m$ . Partition  $D$  into  $m$ 
       disjoint subsets  $D_1, D_2, \dots, D_m$  based on the  $m$  values of  $A_g$ .
17     for each  $D_j$  in  $\{D_1, D_2, \dots, D_m\}$  do
18       if  $D_j \neq \emptyset$  then
19         create a branch (edge) node  $T_j$  for  $v_j$  as a child node of  $T$ ;
20         decisionTree( $D_j, A - \{A_g\}, T_j$ ) //  $A_g$  is removed
21       end
22     end
23   end
24 end

```

**Entropy**  $D$  is the dataset.  $C$  are the different classes in the dataset.

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \text{Pr}(C_j) \log_2 \text{Pr}(C_j)$$

Is a positive value. The sum of the probabilities of the classes is 1. The more the data is purer, the more the entropy value is close to zero. Worst entropy is 1 (all the classes are equally distributed).

$$\text{entropy}_{A_i}(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times \text{entropy}(D_j)$$

This value is the entropy if we choose to partition the data over attribute  $A$ . There are  $v$  possible values for the attribute  $A$ .

$$\text{gain}(D, A_i) = \text{entropy}(D) - \text{entropy}_{A_i}(D)$$

The higher the gain the better

## 6 Naive based classification

Product rule

$$\Pr(a_1, a_2) = \Pr(a_1)\Pr(a_2|a_1)$$

Product rule for conditional probability

$$\Pr(a_1, a_2|c) = \Pr(a_2|c)\Pr(a_1|c, a_2)$$

General case

$$\Pr(a_1 \cdots a_n | c) = \Pr(a_1 | c)\Pr(a_2 | c, a_1) \cdots \Pr(a_n | c, a_1 \cdots a_{n-1})$$

Conditional independence assumption

$$\Pr(a_i | c, a_1 \cdots a_{i-1}) = \Pr(a_i | c)$$

Goal:

$$\Pr(c | (a_1 \cdots a_n)) = \frac{\Pr(c)\Pr((a_1 \cdots a_n) | c)}{\Pr(a_1 \cdots a_n)}$$

Using the law of total probability

$$\Pr(a_1 \cdots a_n) = \sum_{r=1}^{|C|} \Pr(c_r)\Pr((a_1 \cdots a_n) | c_r)$$

Using the product rule and the conditional independence assumption

$$\Pr(c | (a_1 \cdots a_n)) = \frac{\Pr(c) \prod_{i=1}^n \Pr(a_i | c)}{\sum_{r=1}^{|C|} \Pr(c_r) \prod_{i=1}^n \Pr(a_i | c_r)}$$

Adjusting the probability to account for attribute values that don't occur with that class.

$$\Pr(A_i = a_i | C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda m_i}$$

Instead if there are missing values we just ignore them and use what we have

- $n_j$ : # examples with  $C=c_j$  in training data
  - $n_{ij}$ : # examples with both  $A_i=a_i$  and  $C=c_j$
  - $m_i$ : # possible values of attribute  $A_i$ .
  - Normally, we use  $\lambda=1$
- 

## 7 Naive based text classification

$$\begin{aligned}\Pr(c|d) &= \frac{\Pr(c)\Pr(d|c)}{\Pr(d)} \\ &= \frac{\Pr(c) \prod_{k=1}^{\text{all word in d}} \Pr(w_k|c)}{\sum_{r=1}^{|C|} \Pr(c_r) \prod_{k=1}^{\text{all word in d}} \Pr(w_k|c_r)} \\ \Pr(c) &= \frac{\sum_{i=1}^{|D|} \Pr(c|d_i)}{|D|} \\ \Pr(w|c) &= \frac{\sum_{i=1}^{|D|} N_i \Pr(c|d_i)}{\sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c|d_i)}\end{aligned}$$