Syllabus

Fri. 21.10.	(1)	0. Introduction
		A. Supervised Learning: Linear Models & Fundamentals
Fri. 27.10.	(2)	A.1 Linear Regression
Fri. 3.11.	(3)	A.2 Linear Classification
Fri. 10.11.	(4)	A.3 Regularization
Fri. 17.11.	(5)	A.4 High-dimensional Data
		B. Supervised Learning: Nonlinear Models
Fri. 24.11.	(6)	B.1 Nearest-Neighbor Models
Fri. 1.12.	(7)	B.2 Neural Networks
Fri. 8.12.	(8)	B.3 Decision Trees
Fri. 15.12.	(9)	B.4 Support Vector Machines
Fri. 12.1.	(10)	B.5 A First Look at Bayesian and Markov Networks
		C. Unsupervised Learning
Fri. 19.1.	(11)	C.1 Clustering
Fri. 26.1.	(12)	C.2 Dimensionality Reduction
Fri. 2.2.	(13)	C.3 Frequent Pattern Mining

Outline

1. The Frequent Itemset Problem

2. Breadth First Search: Apriori Algorithm

3. Depth First Search: Eclat Algorithm

4. Supervised Pattern Mining

Outline

1. The Frequent Itemset Problem

2. Breadth First Search: Apriori Algorithm

3. Depth First Search: Eclat Algorithm

4. Supervised Pattern Mining

Market Basket Analysis

Association rules in large transaction datasets:

- look for products frequently bought together (**frequent itemsets**).
- look for rules in buying behavior (**association rules**)

Examples:

- $\{\text{beer}, \text{pampers}, \text{pizza}\}$ (support=0.5)
  $\{\text{bread}, \text{milk}\}$ (support=0.5)
- If beer and pampers, then pizza (confidence= 0.75)
  If bread, then milk (confidence=0.75)

cid	beer	bread	icecream	milk	pampers	pizza
1	+	−	−	+	+	+
2	+	+	−	−	+	+
3	+	−	+	−	+	+
4	−	+	−	+	−	+
5	−	+	+	+	−	−
6	+	+	−	+	+	−

Transaction Data, Frequency vs Support

Let I be a set called **set of items**.
A subset $X \subseteq I$ is called **itemset**.

Let $\mathcal{D} \subseteq \mathcal{P}(I)$ be a set of subsets of I called **transaction data set**.
An element $X \in \mathcal{D}$ is called **transaction**.

The **frequency of a subset X in a data set $\mathcal{D}$** is (as always)

$$\text{freq}(X; \mathcal{D}) := |\{Y \in \mathcal{D} \mid X = Y\}|$$

The **support of a subset X in a data set $\mathcal{D}$** is the number of transactions it is a subset of:

$$\text{sup}(X; \mathcal{D}) := |\{Y \in \mathcal{D} \mid X \subseteq Y\}|$$

Note: $\mathcal{D}$ really is a multiset: a transaction could occur multiple times in $\mathcal{D}$ and then is counted as often as it occurs in computing frequency and support.

Transaction Data, Frequency vs Support / Example

$$I := \{1, 2, 3, 4, 5, 6, 7\}$$

$$\begin{array}{lllll}
\mathcal{D} := \{ & \{ & 1, 3, 5 & \} & \\
 & \{ & 1, 2, 3, 5 & \} & \\
 & \{ & 1, 3, 4, 6 & \} & \\
 & \{ & 1, 3, 4, 5, 7 & \} & \\
 & \{ & 2, 4, 7 & \} & \\
 & \{ & 1, 3, 5 & \} & \\
 & \{ & 1, 5, 7 & \} & \\
 & \{ & 1, 2, 3, 4, 5 & \} & \}
\end{array}$$

$$\text{freq}(\{1, 3, 5\}) = 2$$
$$\text{sup}(\{1, 3, 5\}) = 5$$

The Frequent Itemsets Problem

Given

- a set I (called **set of items**),
- a set $\mathcal{D} \subseteq \mathcal{P}(I)$ of subsets of I called **transaction data set**, and
- a number $s \in \mathbb{N}$ called **minimum support**,

find all subsets X of I whose support exceeds the given minimum support

$$\sup(X; \mathcal{D}) := |\{Y \in \mathcal{D} \mid X \subseteq Y\}| \geq s$$

and their support.

Such subsets $X \subseteq I$ with $\sup(X) \geq s$ are called **frequent** (w.r.t. minimum support s in data set $\mathcal{D}$).

Subsets of Frequent Itemsets are Frequent

Obviously, the support of a subset is at least as large as the one of any superset:

$$\text{for all } X \subseteq Y \subseteq I : \quad \sup X \geq \sup Y$$

For a frequent set, all its subsets are frequent.

The **Maximal** Frequent Itemsets Problem

Given

- a set I (called **set of items**),
- a set $\mathcal{D} \subseteq \mathcal{P}(I)$ of subsets of I called **transaction data set**, and
- a number $s \in \mathbb{N}$ called **minimum support**,

find all **maximal** subsets X of I whose support exceeds the given minimum support

$$\sup(X; \mathcal{D}) := |\{Y \in \mathcal{D} \mid X \subseteq Y\}| \geq s$$

and their support.
I.e., there exists no frequent superset of X, i.e., no set $X' \subseteq I$ with

- $\sup(X'; \mathcal{D}) \geq s$ and
- $X \subsetneq X'$

Surprising Frequent Itemsets

Example:
Assume item 1 occurs in 50% of all transactions and
item 2 occurs in 25% of all transactions.

- Is it surprising that itemset $\{1,2\}$ occurs in 12.5% of all transactions?
- Does a relative support of 12.5% of itemset $\{1,2\}$ signal a strong association between both items?

$$p(\{1\} \subseteq X) = 0.5, \quad p(\{2\} \subseteq X) = 0.25$$

If both items occur independently

$$\rightsquigarrow p(\{1,2\} \subseteq X) = p(\{1\} \subseteq X) p(\{2\} \subseteq X) = 0.125$$

Surprising Frequent Itemsets: Lift

$$\text{lift}(X) := \frac{\frac{1}{N}\sup X}{\prod_{x\in X}\frac{1}{N}\sup\{x\}}, \quad N := |\mathcal{D}|$$

- lift(X) > 1: itemset X is more frequent than expected (positive association)
- lift(X) < 1: itemset X is less frequent than expected (negative association)

Example:

$$\text{lift}(\{1,2\}) = \frac{\frac{1}{N}\sup\{1,2\}}{\frac{1}{N}\sup\{1\}\frac{1}{N}\sup\{2\}} = \frac{0.125}{0.5\cdot 0.25} = 1$$

Association Rules

Sometimes one is interested to extract if-then rules of the type

if a transaction contains items X, then it usually also contains items Y
allmost transactions containing X also contain Y

Find all **association rules** (X, Y), $X, Y \subseteq I, X \cap Y = \emptyset$ that

- are **exact enough** / hold in most cases:
  high **confidence**, confidence exceeds minimum confidence c:

$$\text{conf}(X, Y) := \frac{\sup(X \cup Y)}{\sup(X)} \geq c$$

- are **general enough** / frequently applicable:
  high **support**, support exceeds minimum support s:

$$\sup(X, Y) := \sup(X \cup Y) \geq s$$

Finding All Association Rules

To find all association rules that

- exceed a given minimum confidence c and
- exceed a given minimum support s

it is sufficient

1. to find all frequent itemsets that exceed a given minimum support s **and their supports** and then
2. to split each frequent itemset Z in any two subsets X, Y s.t. the rule (X, Y) meets the minimum confidence requirement.
    - start with rule $(Z, \emptyset)$ with confidence 1,
    - iteratively move one element from body to head and retain only those rules that meet the minimum confidence requirement.

To compute confidences only the support of the itemsets (and their subsets) are required.

Nominal Data as Transaction Data

Data consisting of only nominal variables can be naturally represented as transaction data.

Example:

- $X_1 : \mathrm{dom}(X_1) = \{\mathrm{red}, \mathrm{green}, \mathrm{blue}\}$: border color,
- $X_2 : \mathrm{dom}(X_2) = \{\mathrm{red}, \mathrm{green}, \mathrm{blue}\}$: area color,
- $X_3 : \mathrm{dom}(X_3) = \{\mathrm{triangle}, \mathrm{rectangle}, \mathrm{circle}\}$: shape,
- $X_4 : \mathrm{dom}(X_4) = \{\mathrm{small}, \mathrm{medium}, \mathrm{large}\}$: size.

Vector representation:

$$x = (\mathrm{green}, \mathrm{blue}, \mathrm{rectangle}, \mathrm{large})$$

Itemset representation:

$$x = \{\mathrm{border.green}, \mathrm{area.blue}, \mathrm{rectangle}, \mathrm{large}\}$$

Numerical / Any Data as Transaction Data

To represent data with numerical variables as transaction data, numerical variables have to be **discretized** to ordinal/nominal levels.

Example:

- $X_1 : \text{dom}(X_1) = \{\text{red}, \text{green}, \text{blue}\}$: border color,
- $X_2 : \text{dom}(X_2) = \{\text{red}, \text{green}, \text{blue}\}$: area color,
- $X_3 : \text{dom}(X_3) = \{\text{triangle}, \text{rectangle}, \text{circle}\}$: shape,
- $X_4 : \text{dom}(X_4) = \mathbb{R}_0^+$: diameter.

Vector representation: $x = (\text{green}, \text{blue}, \text{rectangle}, 15)$
Itemset representation: $x = \{\text{border.green}, \text{area.blue}, \text{rectangle}, \text{diameter.2}\}$

Discretization:

- $X_4' : \text{dom}(X_3) = \{1, 2, 3\}$: diameter.
  - $X_4' = 1 :\Leftrightarrow X_4 < 10,$
  - $X_4' = 2 :\Leftrightarrow 10 \leq X_4 < 20,$
  - $X_4' = 3 :\Leftrightarrow 20 \leq X_4.$

Discretization Schemes

- **equi-range**:
  - split the domain of the variable in k intervals of same size
- **equi-volume** (w.r.t. a sample/dataset $\mathcal{D}$):
  - split the domain of the variable in k intervals with same frequency

Discretization of numerical variables can be useful in many other contexts.

- e.g., discretization can be used to model non-linear dependencies.

Outline

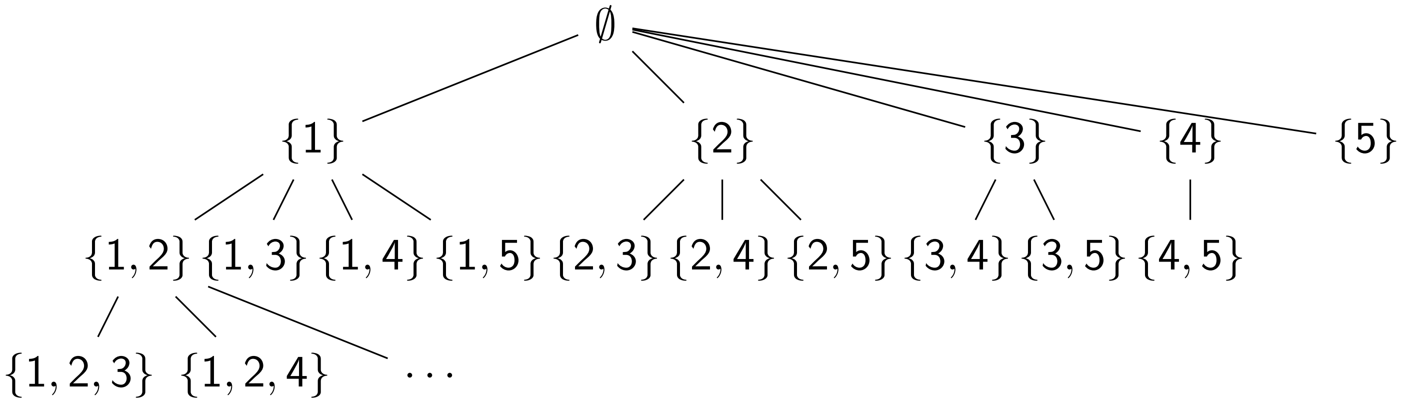
1. The Frequent Itemset Problem

2. Breadth First Search: Apriori Algorithm

3. Depth First Search: Eclat Algorithm

4. Supervised Pattern Mining

Naive Breadth First Search



Naive Breadth First Search

To find all frequent itemsets, one can employ **Breadth First Search**:

1. start with all **frequent itemsets** F_0 of size $k := 0$:

$$F_0 := \{\emptyset\}$$

2. for each $k = 1, 2, \ldots, |I|$: find all frequent itemsets F_k of size k:
3. for $k = 1, 2, \ldots, |I|$, **while** $F_{k-1} \neq \emptyset$:
   3.1 extend frequent itemsets F_{k-1} to **candidates** C_k:

$$C_k := \{X \cup \{y\} \mid X \in F_{k-1}, y \in I\}$$

   3.2 **count the support** of all candidates

$$s_X := \sup(X, \mathcal{D}), \quad X \in C_k$$

   3.3 retain only frequent candidates as **frequent itemsets** F_k:

$$F_k := \{X \in C_k \mid \sup X =: s_X \geq s\}$$

Improvement 1: Fewer Candidates

- $k-$candidates can be created from different $k-1$-subsets:

$$\{1,3,4,7\} = \{1,3,4\} \cup \{7\} = \{1,3,7\} \cup \{4\}$$

$\rightsquigarrow$ add **only larger items** to a $k-1$-subset.

- it makes no sense to add items that are themselves not frequent:

$$\sup(\{1,3,4\} \cup \{7\}) \leq \min\{\sup\{1,3,4\}, \sup\{7\}\}$$

$\rightsquigarrow$ add **only frequent items** from F_1.

- it makes no sense to create candidates with infrequent subsets:

$$\sup(\{1,3,4,7\}) \leq \min\{\sup\{1,3,4\}, \sup\{1,3,7\}, \sup\{1,4,7\}, \sup\{3,4,7\}\}$$

$\rightsquigarrow$ **fuse candidates** from two frequent itemsets from F_{k-1}, check all other subsets of size $k-1$.

Ordered Itemsets, Prefix and Head

Let us fix an order on the items I (e.g., $<$ for $I \subseteq \mathbb{N}$).
Let $X \subseteq I$ be an itemset, then

$$h(X) := \max X$$

is called **the head of X** and

$$p(X) := X \setminus \{h(X)\}$$

is called **the prefix of X**.

Example:

$$h(\{1,3,4,7\}) = 7$$
$$p(\{1,3,4,7\}) = \{1,3,4\}$$

For two $k-1$-itemsets X, Y:

$$\left.\begin{array}{l} X \cup Y \text{ yields a } k\text{-candidate} \\ \text{that extends } X \text{ by a larger item} \end{array}\right\} \iff p(X) = p(Y) \text{ and } h(X) < h(Y)$$

Improved Breadth First Search (1/2)

To find all frequent itemsets:

1. start with all **frequent itemsets F_0** of size $k := 0$:

$$F_0 := \{\emptyset\}$$

2. for $k = 1, 2, \ldots, |I|$, while $F_{k-1} \neq \emptyset$:
   2.1 extend frequent itemsets F_{k-1} to **candidates C_k**:

$$C'_k := \{X \cup \{h(Y)\} \mid X, Y \in F_{k-1}, p(X) = p(Y), h(X) < h(Y)\}$$

   2.2 **retain only candidates with frequent $k-1$-subsets (pruning):**

$$C_k := \{X \in C'_k \mid \forall x \in X : X \setminus \{x\} \in F_{k-1}\}$$

   2.3 **count the support** of all candidates

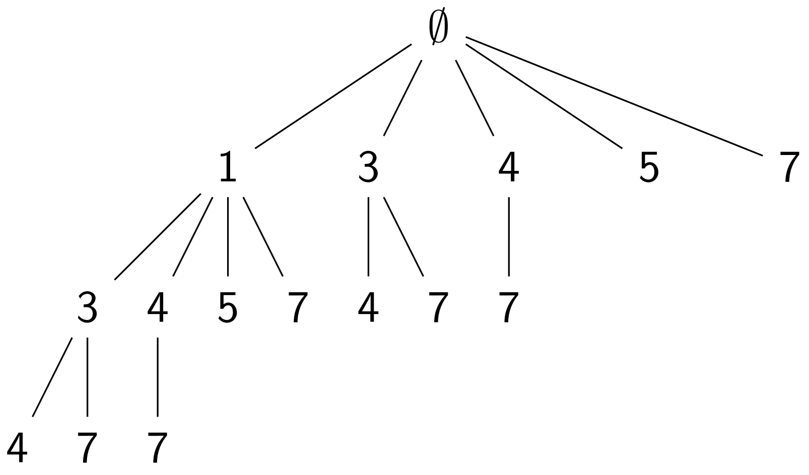
$$s_X := \sup(X, \mathcal{D}), \quad X \in C_k$$

   2.4 retain only frequent candidates as **frequent itemsets F_k**:

$$F_k := \{X \in C_k \mid \sup X =: s_X \geq s\}$$

Improvement 2: Compact Representation and Fast Candidate Creation

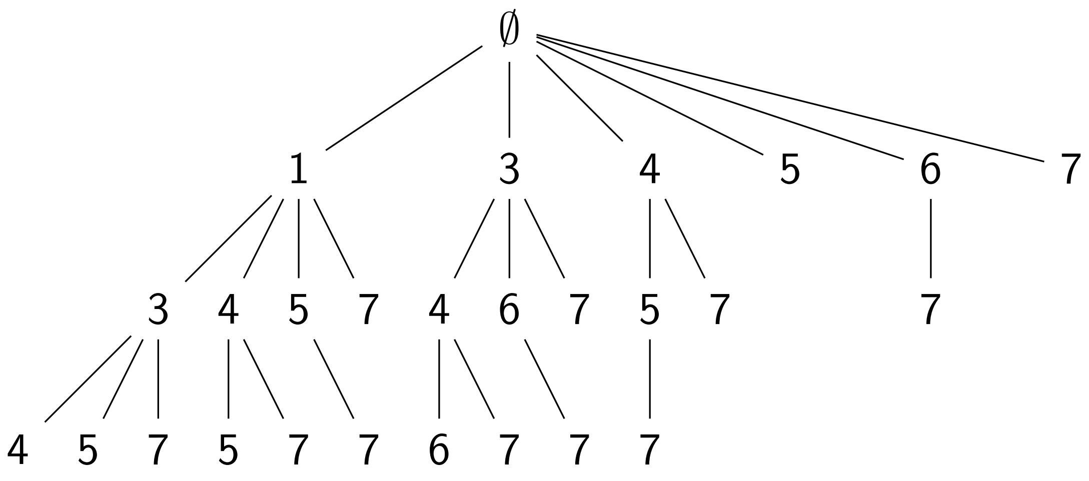
- all frequent itemsets found so far and the latest candidates can be represented compactly in a **trie**:



  - every node is labeled with a single item,
  - every node represents the subset containing all items along the path to the root.

Improvement 2: Compact Representation and Fast Candidate Creation

- to create candidates, just add all right-side siblings as children to a node.



Improvement 3: Fewer Subset Checks for Counting

- computing the support of all candidates C_k naively requires $|C_k|$ passes over the database $\mathcal{D}$.
- instead, count each transaction X into the candidate trie:
  - start at the root N: count(X, root).
  - count(X, N): count transaction X into trie rooted at N:
    1. if N is a leaf node at depth k:

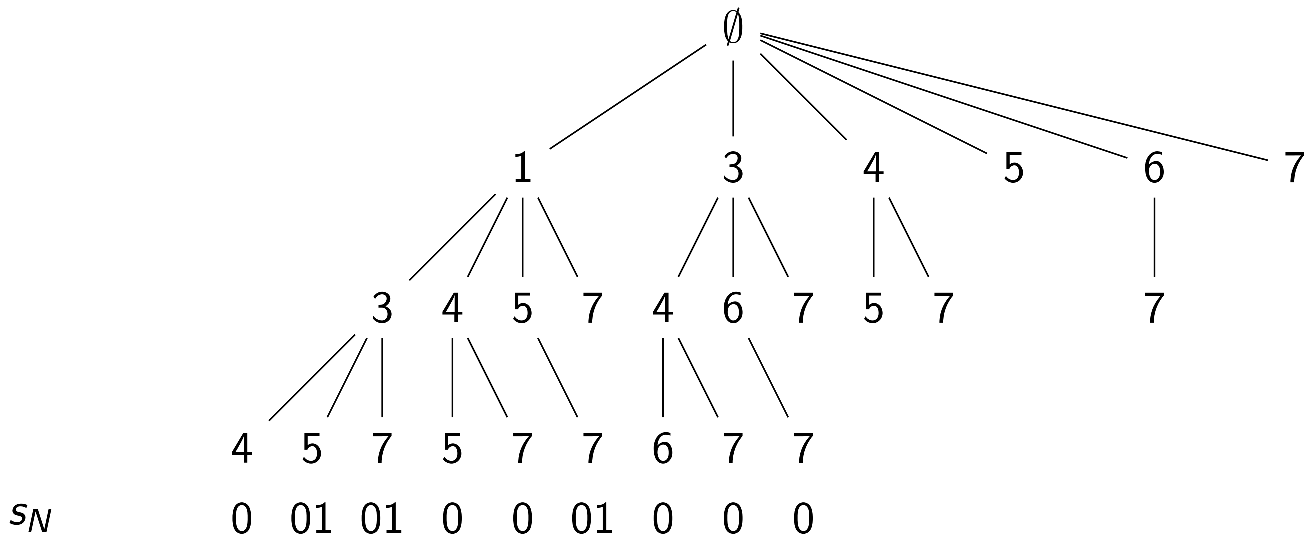
       $$s_N := s_N + 1;$$

    2. else for all child nodes M of N with item$(M) \in X$:

       $$\text{count}(X, M)$$

Example: Counting Transaction into Candidate Trie

Count $\{1, 3, 5, 7, 8\}$ into the trie:



Improved Breadth First Search (2/2): Apriori

To find all frequent itemsets with minimum support s in database $\mathcal{D}$:

```
1. create a trie $T$ with just the root node $R$ without label.
2. for $x \in I$:
       add a node $N$ to $T$ with label $x$ and parent $R$.
3. for $k := 1, 2, \ldots, |I|$, while $T$ has nodes at depth $k$:
   3.1 for $X \in \mathcal{D}$:
           count$(X, R)$.                [computing $N.s$ for nodes at depth $k$]
   3.2 for all nodes $N$ of $T$ at depth $k$:
           if $N.s < s$, remove node $N$.
   3.3 for all nodes $N$ of $T$ at depth $k$:
       3.3.1 for all right-side siblings $M$ of $N$:
                 for all nodes $L$ on the path from $N$ to $R$:
                   check if the node representing itemset$(N) \setminus \{\text{label}(L)\} \cup \{\text{label}(M)\}$
                   exists
                 if so, add a node $K$ to $T$ with the label of $M$ and parent $N$.
4. return $T$
```

Apriori: Sparse Child Arrays

To find all frequent itemsets with minimum support s in database $\mathcal{D}$:

```
1. create a trie $T$ with just the unlabeled root node $R$.
2. for $x \in I$:
       add a node $N$ to $T$ with label $x$ and parent $R$: $R$.child$[x] := N$.
3. for $k := 1, 2, \ldots, |I|$, while $T$ has nodes at depth $k$:
   3.1 for $X \in \mathcal{D}$:
           count$(X, R)$.                [computing $N.s$ for nodes at depth $k$]
   3.2 for all nodes $N$ of $T$ at depth $k$:
           if $N.s < s$, remove node $N$.
   3.3 for all nodes $N$ of $T$ at depth $k$:
       3.3.1 for all right-side siblings $M$ of $N$:
                 for all nodes $L$ on the path from $N$ to $R$:
                   check if the node representing itemset$(N) \setminus \{L.\text{label}\} \cup \{M.\text{label}\}$
                   exists
                 if so, add a node $K$ to $T$ with label of $M$ and parent $N$:
                   $N.$child$[M.$label$] := K$.
4. return $T$
```

Apriori: Algorithmic Improvements

Scalable Apriori implementations usually employ some further simple tricks:

- initially, sort items by decreasing frequency
  - count all item frequencies
  - recode items s.t. code 0 is the most frequent, code 1 the next most frequent etc.
  - remove all infrequent items from the database $\mathcal{D}$.
  - this automatically yields F_1 and their supports.
- count C_2 in a triangular matrix,
  start trie from level 3 onwards.
- remove transactions from the database once they contain no frequent itemset of F_k anymore.
- branches in the candidate trie without leaf nodes are not used for counting and candidate generation.

Outline

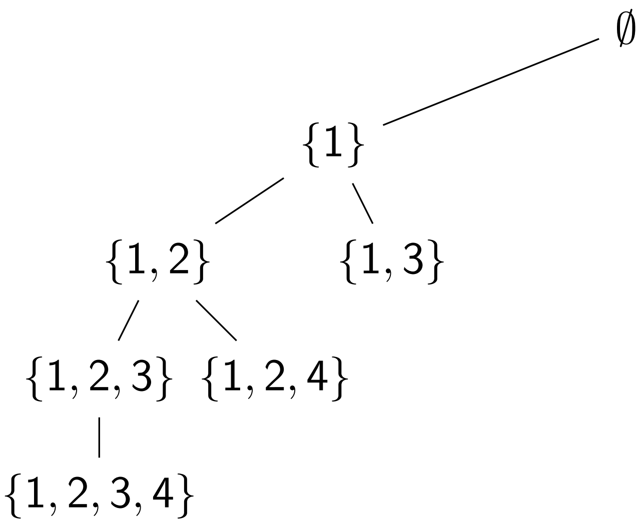
1. The Frequent Itemset Problem

2. Breadth First Search: Apriori Algorithm

3. Depth First Search: Eclat Algorithm

4. Supervised Pattern Mining

Naive Depth First Search



Naive Depth First Search

To find all frequent itemsets, one can employ **Depth First Search**:

- start with the empty itemset:

$$F := \{\emptyset\}$$

$$\text{extend-itemset}(\emptyset)$$

- extend-itemset(P):

```
for all $y \in I$:
    1. extend current prefix $P$ to candidate $X$:
           $X := P \cup \{y\}$
    2. count the support of candidate $X$:
           $s_X := \sup(X, \mathcal{D})$
    3. retain and recursively extend if candidate is frequent:
           if $s_X \geq s$ :
               $F := F \cup \{X\}$
               extend-itemset($X$)
```

Improvement 1: Fewer Candidates

- $k-$candidates can be created from different $k-1$-prefices:

$$\{1,3,4,7\} = \{1,3,4\} \cup \{7\} = \{1,3,7\} \cup \{4\}$$

$\rightsquigarrow$ add **only larger items** to a $k-1$-prefix.

- it makes no sense to add items that are themselves not frequent:

$$\text{sup}(\{1,3,4\} \cup \{7\}) \leq \min\{\text{sup}\{1,3,4\}, \text{sup}\{7\}\}$$

$\rightsquigarrow$ add **only frequent items**.

- it makes no sense to create candidates with infrequent subsets:

$$\text{sup}(\{1,3,4,7\}) \leq \min\{\text{sup}\{1,3,4\}, \text{sup}\{1,3,7\}, \text{sup}\{1,4,7\}, \text{sup}\{3,4,7\}\}$$

$\rightsquigarrow$ **fuse candidates** from two frequent $k-1$-itemsets, check all other subsets of size $k-1$.

Checking $k-1$-subsets in DFS

Checking all $k-1$-subsets:

- In BFS:
  - all frequent $k-1$-itemsets are available from last level
  - no problem
- In DFS:
  - not all $k-1$-itemsets have been checked yet !
  - traverse extension items **in decreasing item order**:
    - ensures that all $k-1$-subsets

$$(i_1, i_2, \ldots, i_{\ell-1}, \widehat{i_\ell}, i_{\ell+1}, \ldots, i_k)$$

are checked before $(i_1, i_2, \ldots, i_{\ell-1}, i_\ell, \ldots, i_{k-1})$.

Improved Depth First Search (1/2)

- start with the empty itemset:

$$F := \{\emptyset\}, J_\emptyset := \{x \in I \mid \sup\{x\} \geq s\}$$
$$\text{extend-itemset}(\emptyset, J_\emptyset)$$

- extend-itemset(P, J):

```
for all $y \in J$ in decreasing order:
  1. extend current prefix $P$ to candidate $X$: $X := P \cup \{y\}$
  2. ensure that all $k-1$-subsets are frequent:
     if $\exists \ell = 1, \ldots, k-2 : P \setminus \{P_\ell\} \cup \{y\} \not\in F$, then skip and go to next $y$
  3. count the support of candidate $X$: $s_X := \sup(X, \mathcal{D})$
  4. retain and recursively extend if candidate is frequent:
     if $s_X \geq s$ :
       $F := F \cup \{X\}$
       $J_X := \{z \in J \mid z > y, s_{P \cup \{z\}} \geq s\}$
       extend-itemset($X, J_X$)
```

Improvement 2: Project Data for Fast Support Counting

- counting the support of every candidate separately is very expensive
- first idea:
  - do not check transactions again that do not contain the prefix P
  - $\rightsquigarrow$ keep a list of transaction IDs that contain the prefix:

$$\mathcal{D} = \{X_1, \ldots, X_N\} \qquad \text{full data set}$$
$$T(P) := \{t \in \{1, \ldots, N\} \mid P \subseteq X_t\} \qquad \text{transaction cover of } P$$

  - to compute frequency of $P \cup \{y\}$,
    check only $P \cup \{y\} \overset{?}{\in} X_t$ with $t \in T(P)$
- final idea:
  - compute T recursively:

$$T(P \cup \{z\} \cup \{y\}) = T(P \cup \{z\}) \cap T(P \cup \{y\})$$

  - store extension items z together with $T(P \cup \{z\})$.

Improved Depth First Search (2/2): Eclat

- start with the empty itemset:

$$F := \{\emptyset\}, J_\emptyset := \{(x, T(x)) \mid x \in I, |T(x)| \geq s\}$$
$$\text{extend-itemset}(\emptyset, \{1, \ldots, N\}, J_\emptyset)$$

- extend-itemset(P, T_P, J):

```
for all $(y, T_y) \in J$ in decreasing order of y:
    1. extend current prefix $P$ to candidate $X$: $X := P \cup \{y\}$
    2. ensure that all $k-1$-subsets are frequent:
       if $\exists \ell = 1, \ldots, k-2 : P \setminus \{P_\ell\} \cup \{y\} \notin F$, then skip and go to next $y$
    3. compute transaction cover of candidate $X$: $T_X := T_P \cap T_y$
    4. retain and recursively extend if candidate is frequent:
           if $|T_X| \geq s$ :
               $F := F \cup \{X\}$
               $J_X := \{(z, T_{P \cup \{z\}}) \in J \mid (z, T_z) \in J, z > y, |T_{P \cup \{z\}}| \geq s\}$
               extend-itemset$(X, T_X, J_X)$
```

Outline

1. The Frequent Itemset Problem

2. Breadth First Search: Apriori Algorithm

3. Depth First Search: Eclat Algorithm

4. Supervised Pattern Mining

Pattern Encodings

Patterns can be used to describe data instances/transactions:

- in this context, patterns are sometimes called **codes**,
- the list of patterns a **codebook**, and
- the representation of a transaction by **pattern indicators** as **encoding**.

$$\begin{aligned}
\mathcal{D} &:= \{X_1, \ldots, X_N\} && \text{large transaction database}\\
F &:= \{P_1, \ldots, P_K\} && \text{frequent patterns in } \mathcal{D}\\
X'_i &= (\delta(P_k \subseteq X_i))_{k=1,\ldots,K} && \text{representation of } X_i \text{ by pattern indicators}
\end{aligned}$$

Example:

$$\begin{aligned}
F &:= \{\{1,3,5\},\{2,6\},\{9,13\}\}\\
X &:= \{1,2,3,4,5,6,7\}\\
X' &= (1,1,0)
\end{aligned}$$

Pattern Mining as Preprocessing

Given a prediction task and

a data set $\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathcal{P}(I) \times \mathcal{Y}$.

Procedure:

1. mine all frequent patterns P in the predictors of $\mathcal{D}^{\text{train}}$,
   - e.g., using Apriori on $\{x_1, \ldots, x_n\} \subseteq \mathcal{P}(I)$ with minimum support s.
2. encode predictors by $\{x_1, \ldots, x_n\}$ their pattern encodings

$$z_i := (p \subseteq x_i)_{p \in P} \in \{0,1\}^K, \quad K := |P|$$

3. learn a (linear) prediction model

$$\hat{y} : \{0,1\}^K \rightarrow \mathcal{Y}$$

on the latent features based on

$$\mathcal{D}'^{\text{train}} := \{(z_1, y_1), \ldots, (z_n, y_n)\}$$

4. treat the minimum support s (and thus the number K of latent dimensions) as hyperparameter.
   - e.g., find using grid search.

Potential Effects of Using Pattern Encodings

For transaction data / frequent itemsets:

- patterns/itemsets represent **interaction effects**:

$$\delta(\{P_1, \ldots, P_L\} \subseteq X) = \prod_{\ell=1}^{L} \delta(P_\ell \in X)$$

- possibly **useful with linear models**
  - possibly less useful with nonlinear models that model interaction effects on their own.
- frequency used as (naive) **proxy for predictivity** of an interaction.
- minimum support s treated as hyperparameter.

For structured data (sequences, graphs, images, text, etc.):

- a way to **extract features** from structured objects.
  (i.e., to create a vector representation that can be used with any machine learning algorithm)

Supervised Pattern Mining

Methods that extract not just

- **frequent patterns**,
- but **predictive patterns**:

would be useful as basis for prediction.

- but e.g., correlation of a pattern with a target variable does not have the closed-downward property
  - subsets of frequent subsets are frequent,
  - but subsets of predictive subsets may not be predictive.

Outlook

- fpGrowth
- Frequent subsequences / sequential patterns
  - Apriori can be easily adapted for sequential patterns.
  - Eclat adapted to sequential patterns: PrefixScan.
  - Additional pattern symbols: wildcards.
- Frequent subgraphs / graph patterns

Conclusion (12/2)

- Frequent Pattern Mining searches for **frequent itemsets** in large **transaction data**, i.e., aims to find all subsets with a given **minimum support**.
  - **Association rules** can be created by simply splitting frequent itemsets.
  - As subsets of frequent sets are frequent, the result set typically is huge.
    - restrict results by looking only for **maximal frequent itemsets**.
    - rank results by other measures, e.g., **lift**.
  - Any data can be represented as transaction data (evtl. with a discretization loss).
- **Apriori** enumerates all frequent itemsets using **breadth first search**:
  - only candidates with all subsets being frequent are checked (fusing of $k-1$-itemsets, **pruning**).
  - every itemset can be created just once by **sorting itemsets** and adding only larger items.
  - all k-candidates can be represented compactly in a **trie** and their support be counted efficiently in a single pass over the database.

Readings

- Apriori
  - [HTFF05], ch. 14.2,
  - [AS94],[Bor03].
- Eclat
  - [ST04], [Bor03].

References

R. Agrawal and R. Srikant.
Fast algorithms for mining association rules.
In *Proc. of the 20th Int'l Conference on Very Large Databases*, 1994.

Christian Borgelt.
Efficient implementations of apriori and eclat.
In *FIMI'03: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations*, 2003.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin.
The elements of statistical learning: data mining, inference and prediction, volume 27.
Springer, 2005.

Lars Schmidt-Thieme.
Algorithmic features of eclat.
In *FIMI*, 2004.