Algorithms for Association Rules

Tutorial, IMS Singapore 10.-12. December 2003

Markus Hegland

`Markus.Hegland@anu.edu.au`

Centre for Mathematics and its Applications

Mathematical Sciences Institute

Australian National University, Canberra

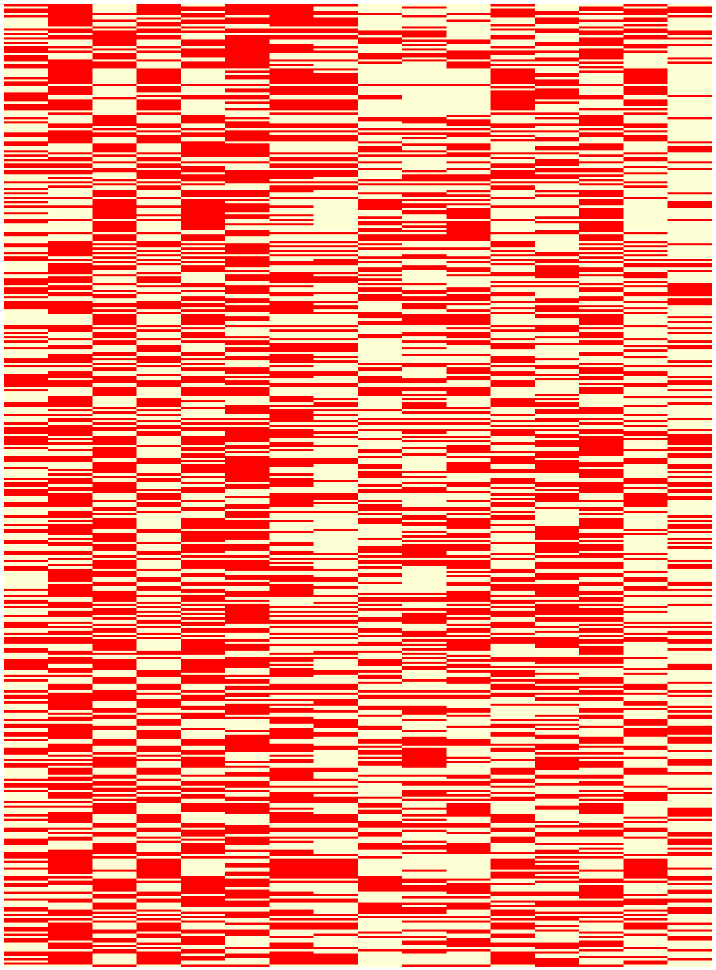
Contents

Introduction

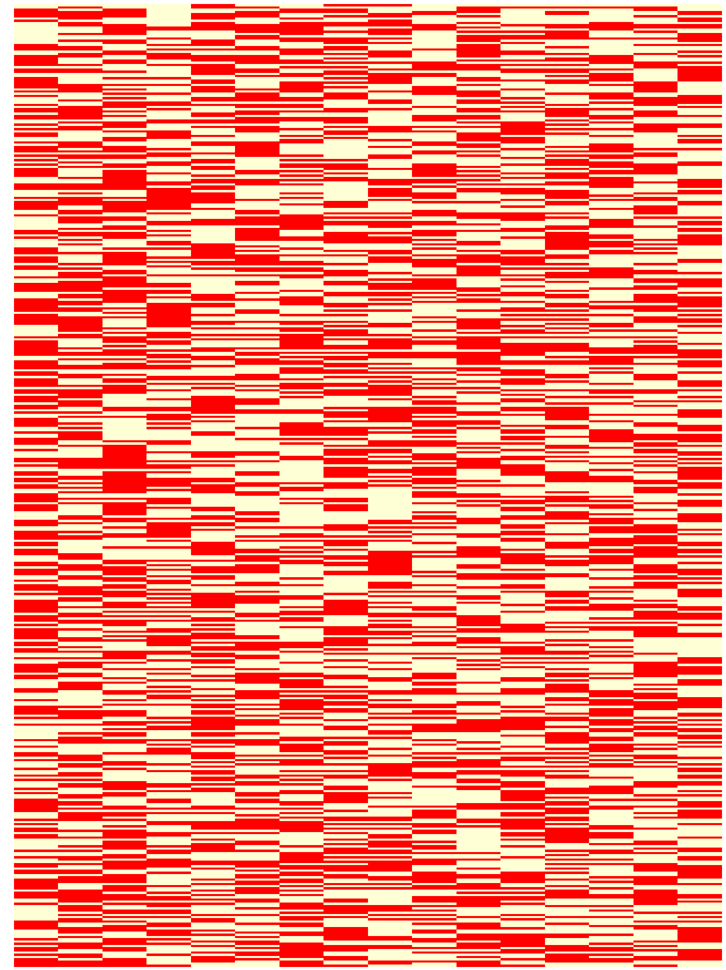
→ Association Rules: Patterns and Noise, Market Baskets, Rules

- Data Mining: Techniques, KDD, other Data Disciplines
- Applications of Data Mining: Data Challenge, Management and Science, Classification

Pattern or Noise?



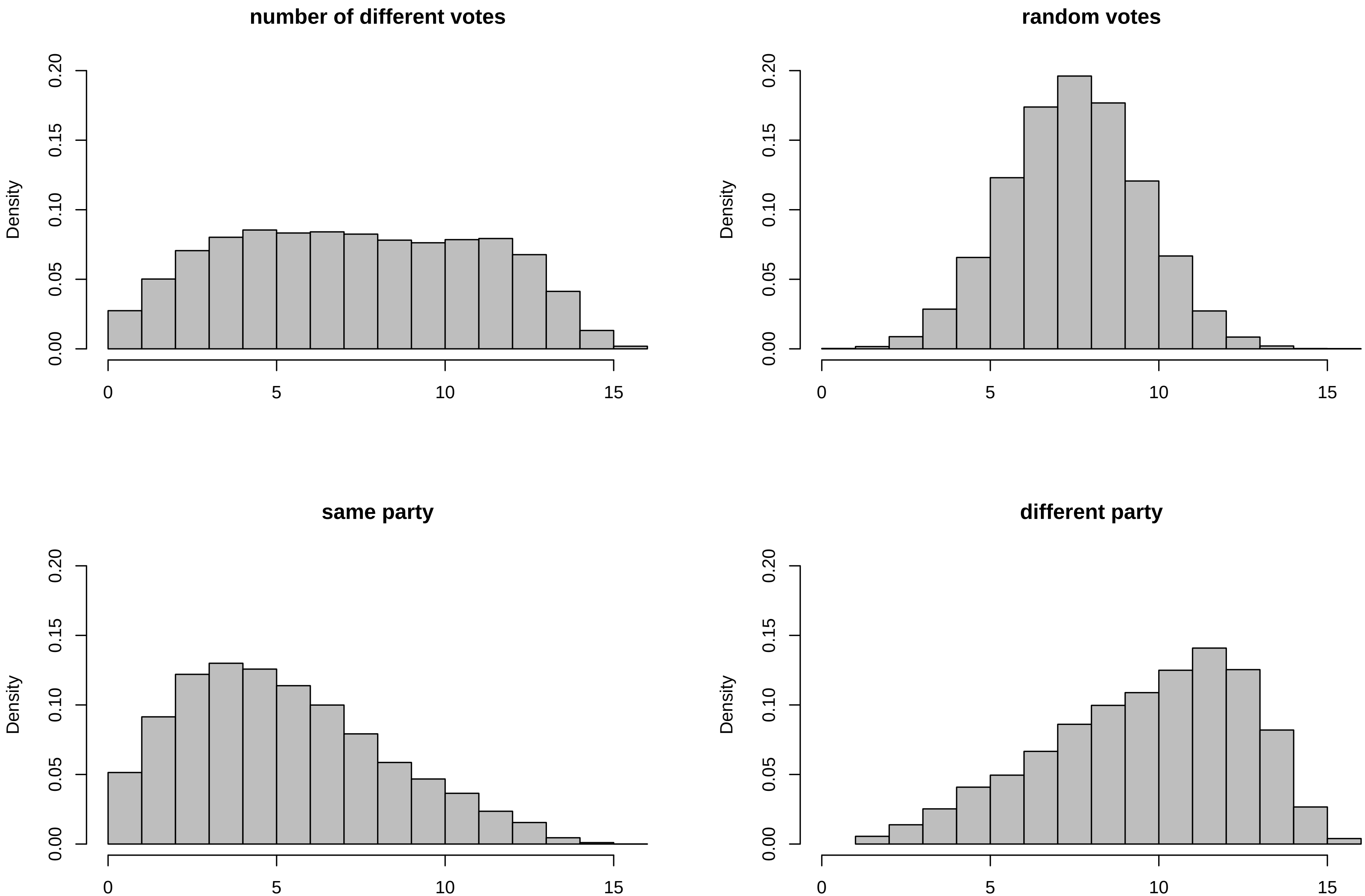
voting data



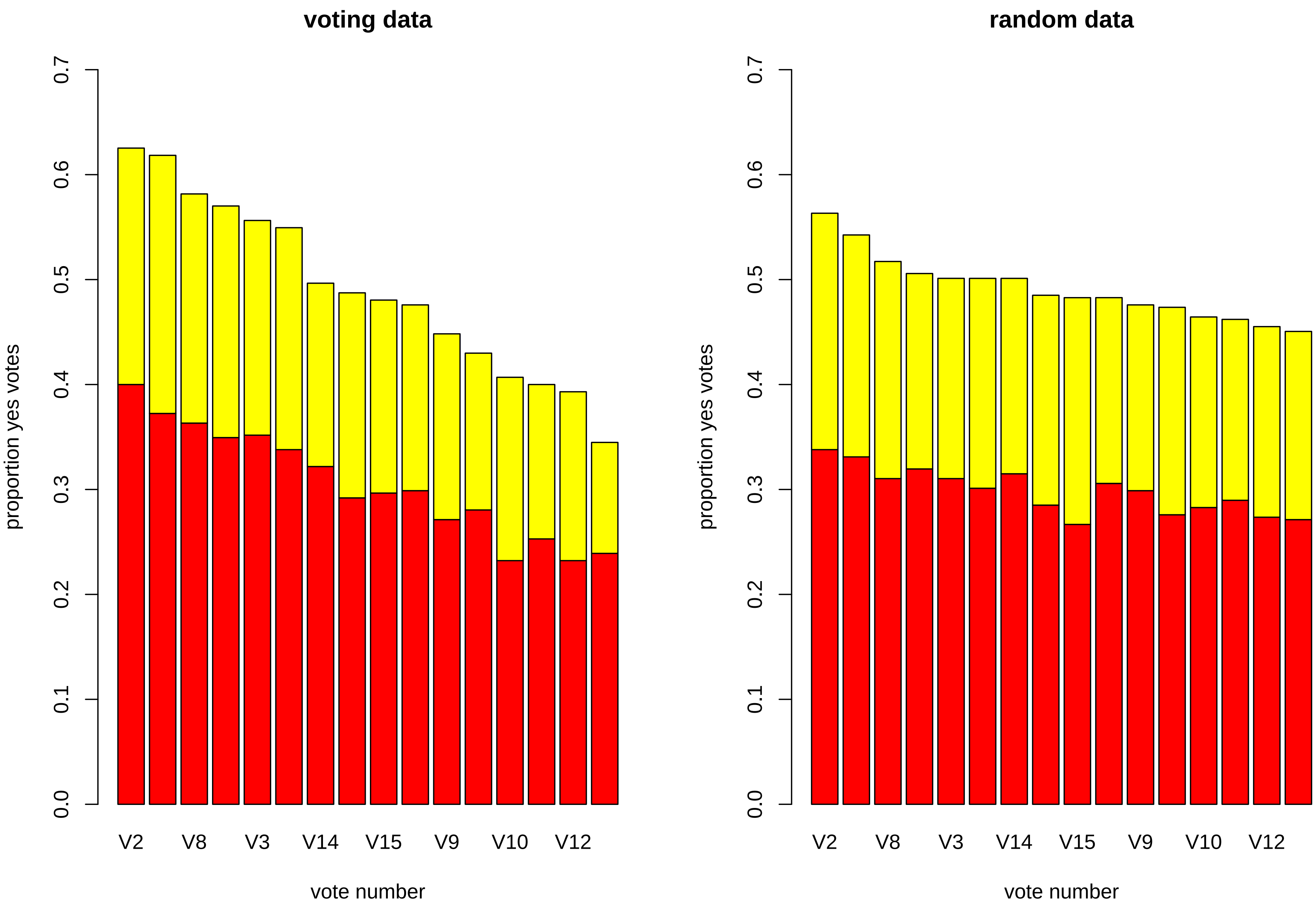
random data

1984 US House of Representatives votes: 16 votes / 435 representatives UCI ML data

Tracing the Pattern

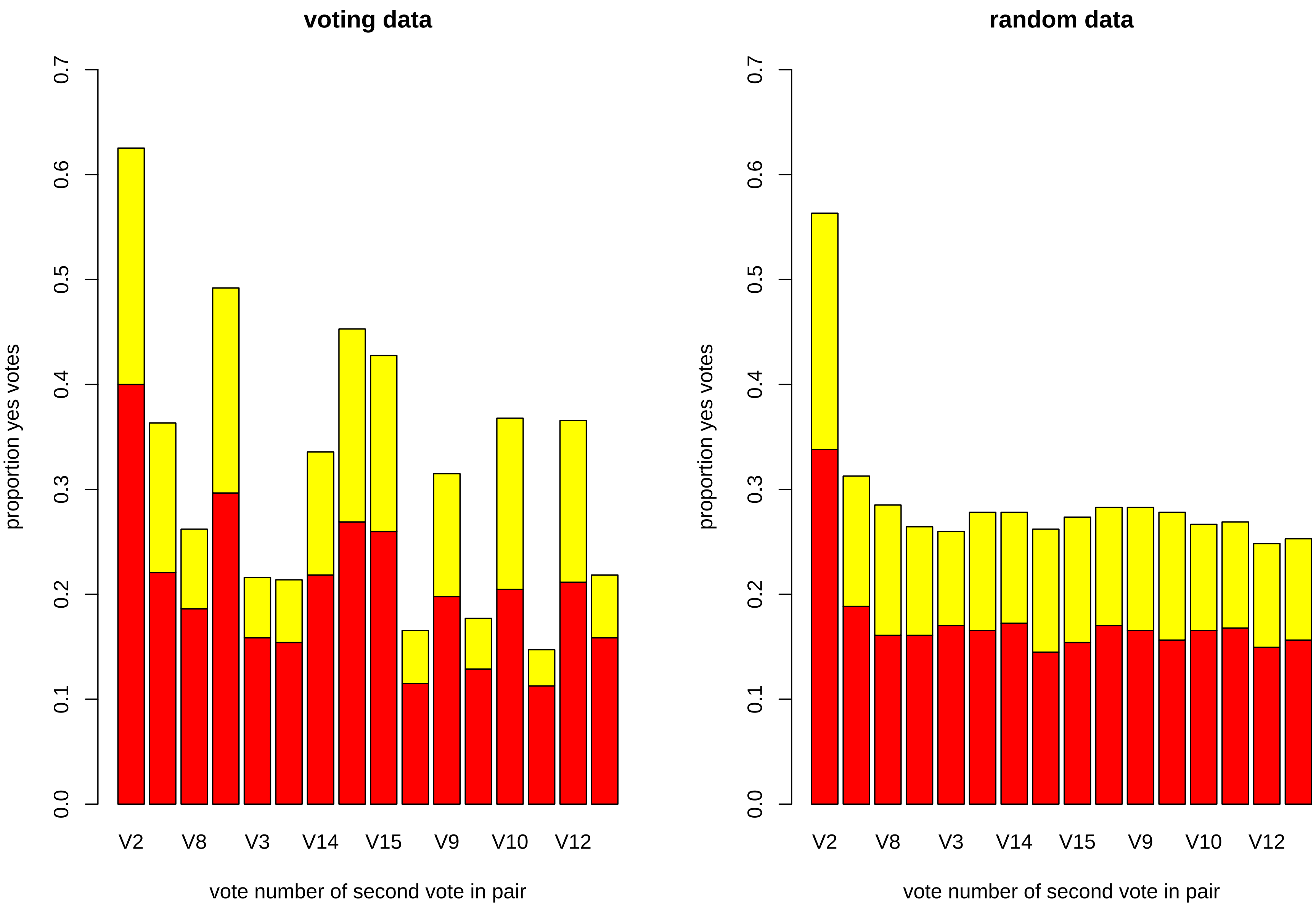


Counting Votes

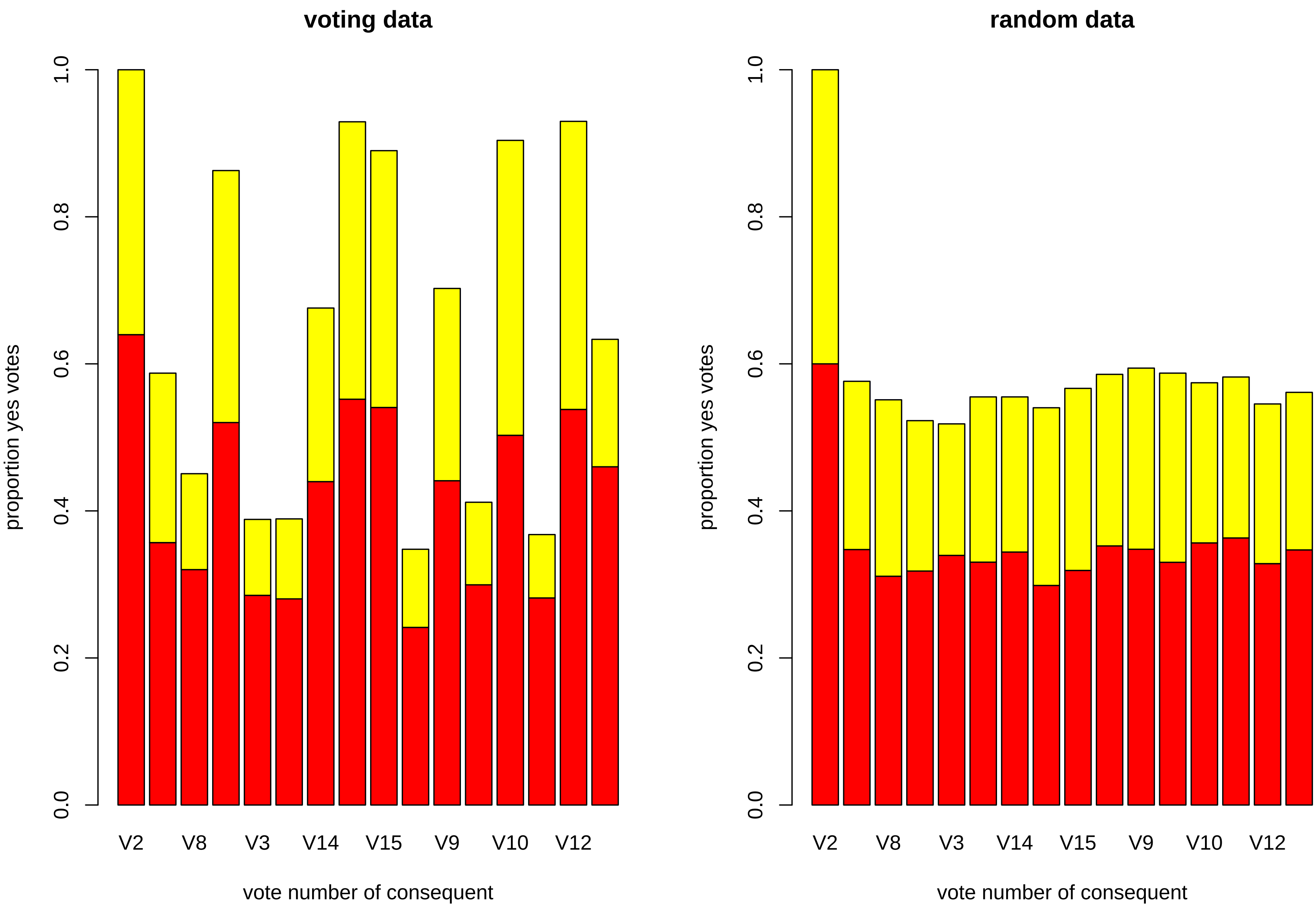


Support for all votes in US congress data

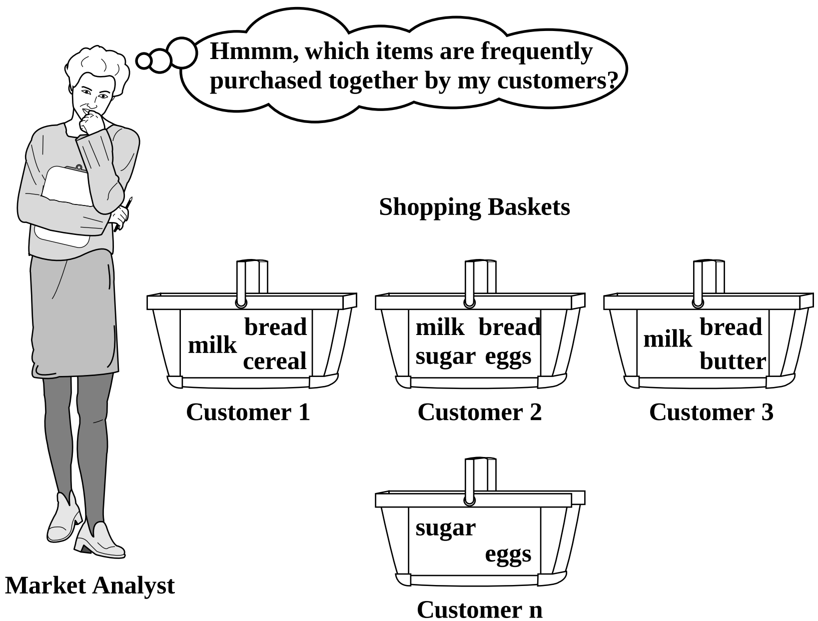
Pairs of Votes



Confidence of Rules



Market Basket Analysis



from: Han/Kamber, “Data Mining”, 2001

- Aim: Understand customer interests and behaviour
- Why: Product placement, specials, marketing
- Health: Basket of medical services
- Challenge: 10,000 items or more not uncommon

What are Rules

- "A male shopper who buys nappies on Friday night also buys beer"
- *if-then rule*: $A \longrightarrow C$
- Predicates: antecedent $A(\mathrm{X})$, consequent $C(\mathrm{X})$
- X feature vector, data $\mathrm{X}_1, \ldots, \mathrm{X}_n$ (flat file)
- $A(\mathrm{X}) = A_1(\mathrm{X}) \wedge \cdots \wedge A_k(\mathrm{X})$
- Challenge: utilise intricate structure between predicates A_i
- Rules are "understandable"

What Makes Rules Interesting

- Rules should be interpretable in domain context
- Interesting rules suggest actions and/or are unexpected
- Example: action = product placement
- Unexpectedness = contradicting beliefs from domain knowledge
  "beer and nappies purchases are unrelated"
- Many discovered rules are *uninteresting*, e.g., trivial, inexplicable or useless $\Rightarrow$ use domain knowledge / constraints

Association Rules = Rules ++

- "If customer buys milk, then she buys bread"
- *Support* = Proportion of baskets which contain both milk and bread
- *Confidence* = Proportion of the baskets with milk which contain bread as well
- Find *all* rules which have support and confidence larger than given threshold: *strong rules*
- Support + confidence $\neq$ Interestingness!

Introduction

- Association Rules: Patterns and Noise, Market Baskets, Rules

→ Data Mining: Techniques, KDD, other Data Disciplines

- Applications of Data Mining: Data Challenge, Management and Science, Classification

Data Mining Techniques

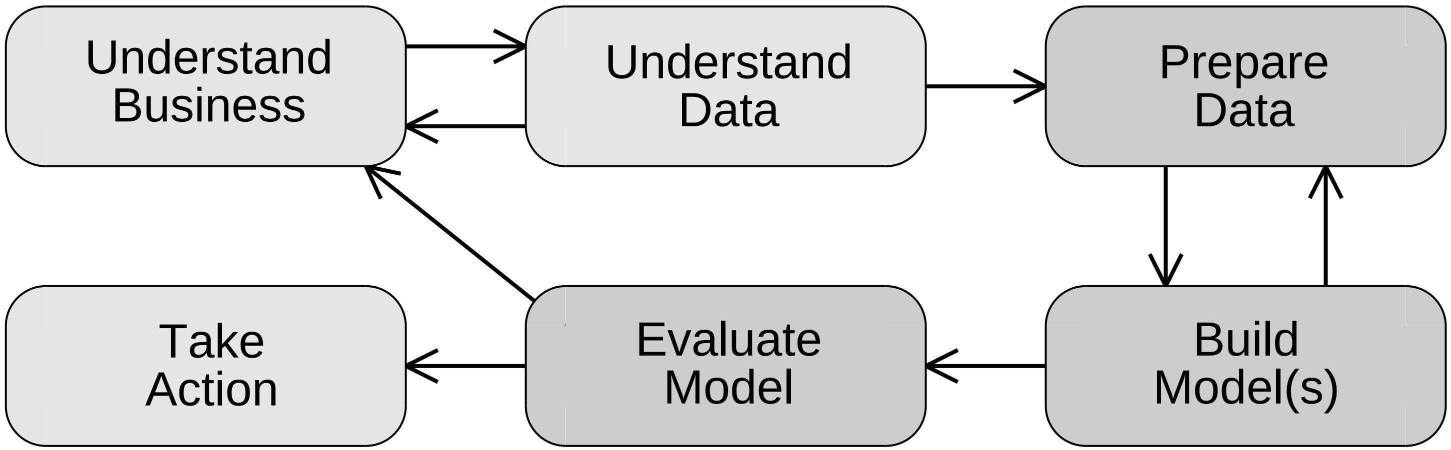
- *What they do*
  Detect patterns in data: rules, associations and functional dependencies, outliers, groupings, data distribution
- *How they do it*
  Search through data and pattern space, nonparametric modelling, filtering, aggregation
- *How well they do it*
  Errors and biases, overfitting, confounding effects, speed

A Definition of KDD

Knowledge discovery in databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

Fayyad, Piatetsky-Shapiro and Smyth (1996)

The KDD Process



- Typically 90 % in first 3 steps
- Build Model = “Data Mining”
- Iterative and interactive process

Data Mining and Database Management

- Management of transactions is not data mining
- Data access and integrity essential
- Search, summarisation and data extraction
- Models to integrate data mining and DBMS:
  - Extract all data into “flat file”
  - SQL deals with data-intensive tasks
  - Extend SQL with important primitives
  - Implement algorithms in SQL

Data Mining and Machine Learning

- Many machine learning methods used in data mining techniques: Neural nets, support vector machines, genetic algorithms, decision trees
- Data mining deals with very large data sets
- Data mining more modest than AI: Automate tedious discovery tasks, not emulate human discovery

Data Mining and Statistics

- Similar goal: analysis of data
- Same limitations: real effects masked and spurious correlations significant
- Different focus:
  - Computational – data size and complexity
  - Exploratory – search for hypothesis

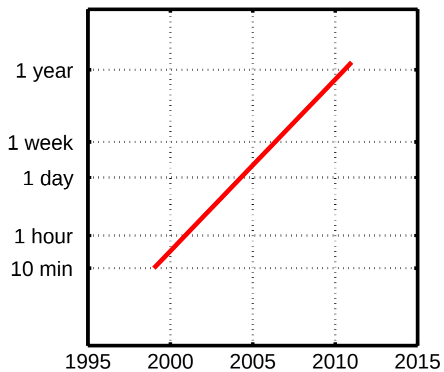
Introduction

- Association Rules: Patterns and Noise, Market Baskets, Rules
- Data Mining: Techniques, KDD, other Data Disciplines

$\longrightarrow$ Applications of Data Mining: Data Challenge, Management and Science, Classification

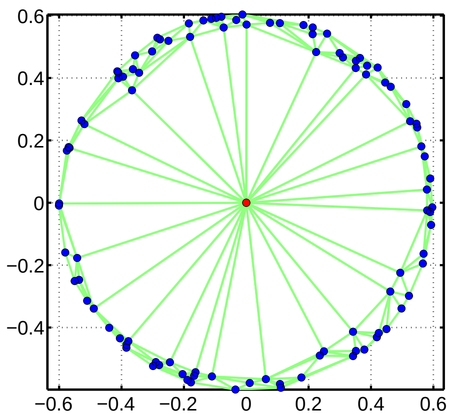
The Data Challenge

- Data size:



Automatic data acquisition
Data doubles every 18 months – need "scalable" algorithms

- Complexity:



Multimedia, text, lots of attributes, concentration, curse of dimensionality

MBA needs Data Mining

- Curse of dimensionality: With 10,000 items there are $2^{10,000} \approx 10^{3,000}$ possible market baskets
- Most market baskets have very few distinct entries
- Model: 10,000 Boolean random variables
- Contingency table has $2^{10,000}$ entries, most are zero
- Find the most common combinations of items through systematic search, discard infrequent combinations

Data Mining in Health Services

- Fraud detection in health insurance
- Assist in evidence-based medicine, detect non-conformance
- Evaluate outcomes of service providers
- Find best practice, most effective treatments
- Find patients at risk of certain ailments
- Predict outcomes based on patient parameters in intensive care
- Cross-selling and marketing in pharma. industry

Applications of Association Rule Mining

- Market basket analysis: product placement, direct marketing
- Web usage mining: weblog data analysis, e-commerce customers, improve information delivery, prime advertisement locations, web page organisation
- DNA sequence and protein structure mining: DNA tandem repeats (. . . TCGGCGGCGGA. . .), protein function prediction, sliding window
- Intrusion detection

A Data Mining Approach to Classification

- Computational challenges for classification:
  - Very large data sets
  - Large numbers of of attributes
  - Complex data
- Basic approach:
  1. Detect dominant features, frequent patterns or regularities of classes
  2. Use this information for classification
- Advantage: Interpretable models based on, e.g., rules and clusters

Classification with Apriori

- *Class Association Rule:*

  $$(X_1 = x_1) \wedge \cdots \wedge (X_k = x_k) \rightarrow Y = y$$

  item = attribute / value pair,   consequent fixed
- Prune rules with high error rate
- Build classifier using “best rules” w.r.t.
  1. Confidence   2. Support   3. Simplicity
- Minimal support 1% Confidence 50%
- Good classification and interpretability

B. Liu, W. Hsu, and Y. Ma, *Integrating classification and association rule mining*, Knowledge Discovery and Data Mining, 1998, pp. 80–86.

Searching for Rules

→ Finding Rules – a Hard Problem

- The Search for Association Rules – Apriori

The Search for Interesting Rules

1. Find the most interesting rule
2. Find all rules with interestingness $\geq$ given bound

- Search-space: $A_{i_1} \wedge \cdots \wedge A_{i_k} \longrightarrow C$ for $A_{i_j} \in \{A_1, \ldots, A_p\}, \; 0 \leq k \leq p$
- Challenge:

There are 2^p different rules!

Complexity of Rule Search

- *Theorem* [Morishita '98]
  The determination of the best rule is NP hard.
- Corollary:
  No algorithms are known which have polynomial time complexity.
- Use approximations and heuristics.

The Search Tree

- Nodes = rules $\boldsymbol{A} \longrightarrow \boldsymbol{C}$, root = $\mathbf{1} \longrightarrow \boldsymbol{C}$.
- Edges = defined by *specialisation*, i.e., if $\boldsymbol{B} \longrightarrow \boldsymbol{A}$ (e.g. $\boldsymbol{B} = \boldsymbol{A} \wedge \boldsymbol{A_s}$) then

$$\boldsymbol{A} \longrightarrow \boldsymbol{C} \quad \longmapsto \quad \boldsymbol{B} \longrightarrow \boldsymbol{C}$$

- Specialisation increases confidence
- Use domain knowledge to prune unwanted branches.
- Complicated rules are impractical $\Longrightarrow$ don't specialise too much.
- More general rules are more interesting and have larger support.

A General Search Approach: GAT

$\mathbf{level} = 1, \quad L_1 = \{1 \rightarrow C\}$
while $L_{\mathbf{level}} \neq \emptyset$ **do**
  $\mathbf{level} = \mathbf{level} + 1$
  for $\mathbf{rule}_1 \in \mathbf{Specialise}(L_{\mathbf{level}-1})$ **do**
    if $\mathbf{rule}_1$ interesting **then**
      output $\mathbf{rule}_1$
    else if $\mathbf{rule}_1$ ripe for pruning **then**
      discard $\mathbf{rule}_1$
    else
      add $\mathbf{rule}_1$ to $L_{\mathbf{level}}$

Generate and test algorithm (Provost, Aronis and Buchanan '99)

Properties of the GAT algorithm

- Expensive part: evaluation of interestingness.
- One data scan per level.
- Complexity $O(n \sum_{l=1}^{\text{maxlevel}} l s_l r_l)$ for testing interestingness.
  s specialisations, r_l rules, n data points
- Scalability in n with a large factor
- Special cases: Greedy algorithm
- Research: Effect of pruning? Alternative algorithms?

Searching for Rules

- Finding Rules – a Hard Problem

$\longrightarrow$ The Search for Association Rules – Apriori

Transactional Data

- $I = \{a_1, a_2, \ldots, a_m\}$ set of *items*
- *Transaction* $T_i \subset I$
- *Transaction Database* $DB = \langle T_1, T_2, \ldots, T_n \rangle$
- *Support* of a *pattern* (or itemset) $A \subset I$:

$$s(A) = \#\{T_i \in DB | A \subset T_i\}/n$$

- *Confidence* of *rule* $A \to B$ for $A, B \subset I$:

$$c(A \to B) = s(A \cup B)/s(A)$$

Mining for patterns and rules

- *Frequent pattern mining problem:*
  Find all predicates A which predefined support
  Such a predicate is called *frequent pattern*
- *Association rule mining:* Find all association rules with predefined support and confidence
  These are the *strong association rules*
- Two step algorithm:
  1. Find all frequent patterns A
  2. Find all predicates A_1 and A_2 such that $A = A_1 \wedge A_2$ and $A_1 \rightarrow A_2$ has predefined confidence

  Note: Only the first step requires scanning the data

The Apriori property

- *Apriori property*:

  If pattern A frequent and $B \subset A$ then B is frequent

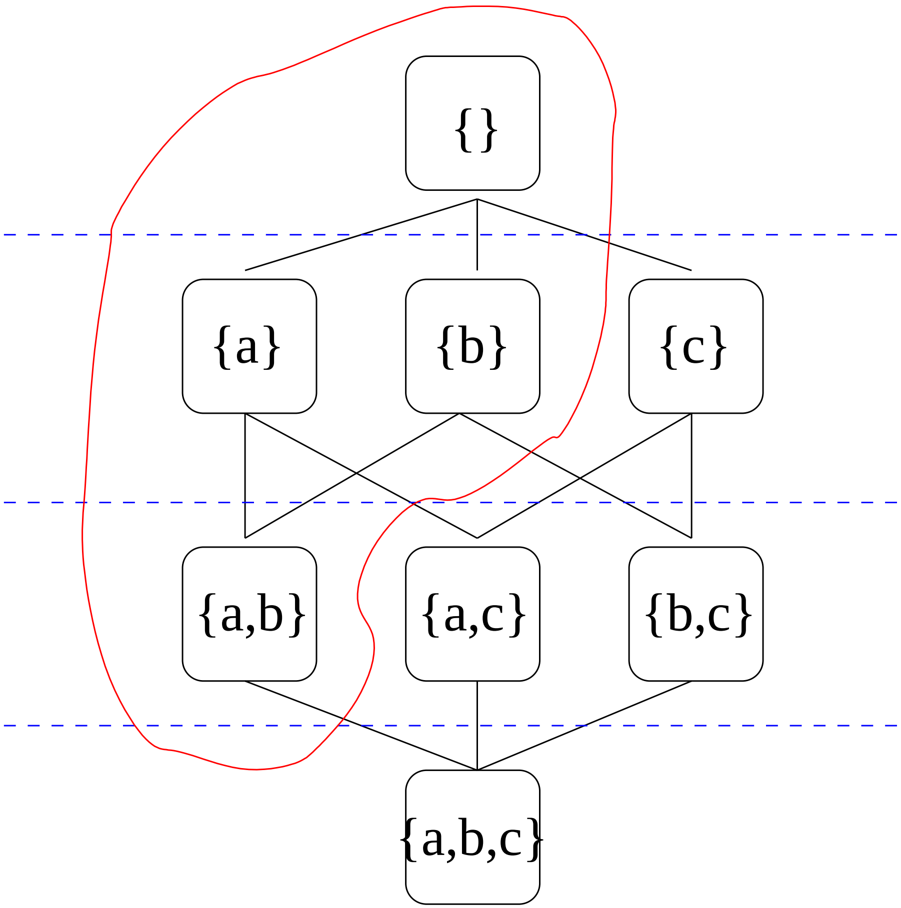
- Find association rules from the frequent itemsets

- Example:

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

R. Agrawal and R. Srikant, *Fast algorithms for mining association rules*, VLDB 94, pp. 487–499

The Search Tree for Apriori



- Itemsets are Boolean lattice
- Frequent itemsets are down-sets
- Apriori is level-wise or breadth-first search

The Algorithm

- $L_1 :=$ {frequent 1-itemsets}
  $\mathrm{level} := 1$
  while L_k is not empty **do**
  $\mathrm{level} := \mathrm{level} + 1$
  $C_{\mathrm{level}} :=$ sets of candidate itemsets
  Prune candidate sets using apriori property
  Determine the support of all candidate itemsets in C_{level}
  $L_{\mathrm{level}} :=$ frequent itemsets in C_{level} : needs DB scan
- Operation count: $O(n \sum_{l=1}^{\mathrm{maxlevel}} l s_l r_l)$ as before

Best Candidates

- $L_k = \{A, B, \ldots\} \subset 2^I$: set of frequent k-itemsets
- Store A as alphabetically ordered lists $A[1 : k]$
- *Join* operation:

$$L_k * L_k := \{A \cup B | A[: k-1] = B[: k-1], A[k] < B[k]\}$$

- **Lemma**: $L_{k+1} \subset L_k * L_k$.
  Proof: $A \in L_{k+1}$, then $A[1 : k] \in L_k$ and
  $A[: k-1] \cup A[k+1] \in L_k$.
- Smallest possible candidate itemset without scan:

$$C_{k+1} = \{A \in L_k * L_k | B \subset A \Rightarrow B \in L_{|B|}\}$$

Example (from Han/Kamber 2001)

Scan D for count of each candidate →

C_1

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

Compare candidate support count with minimum support count →

L_1

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

Generate C_2 candidates from L_1 →

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan D for count of each candidate →

C_2

Itemset	Sup. count
{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

Compare candidate support count with minimum support count →

L_2

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

Generate C_3 candidates from L_2 →

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan D for count of each candidate →

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count →

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Analysis of Apriori

→ Mathematical Modeling – Lattice of Itemsets, Probability and Predicates

- Algorithms – Search in Levelsets, Support
- Enumeration of k-Itemsets
- Bounds on the Number of Candidate Itemsets

Why mathematical modelling

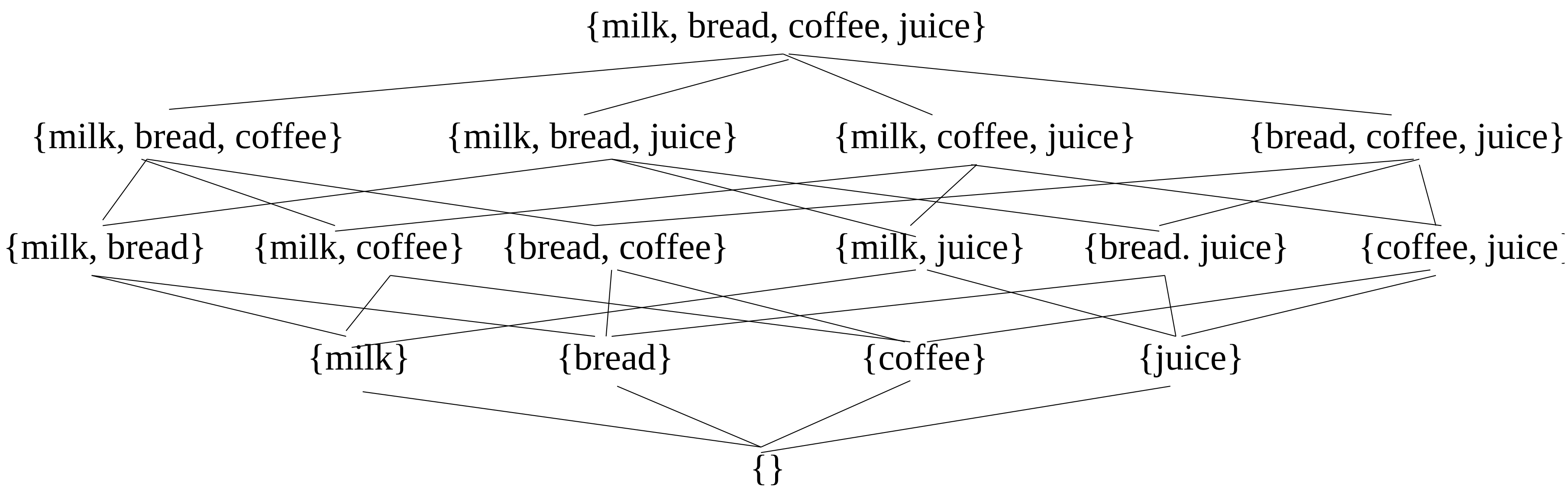
- Analysis of time complexity
- Development of new algorithms
- Implementation of algorithms and datastructures

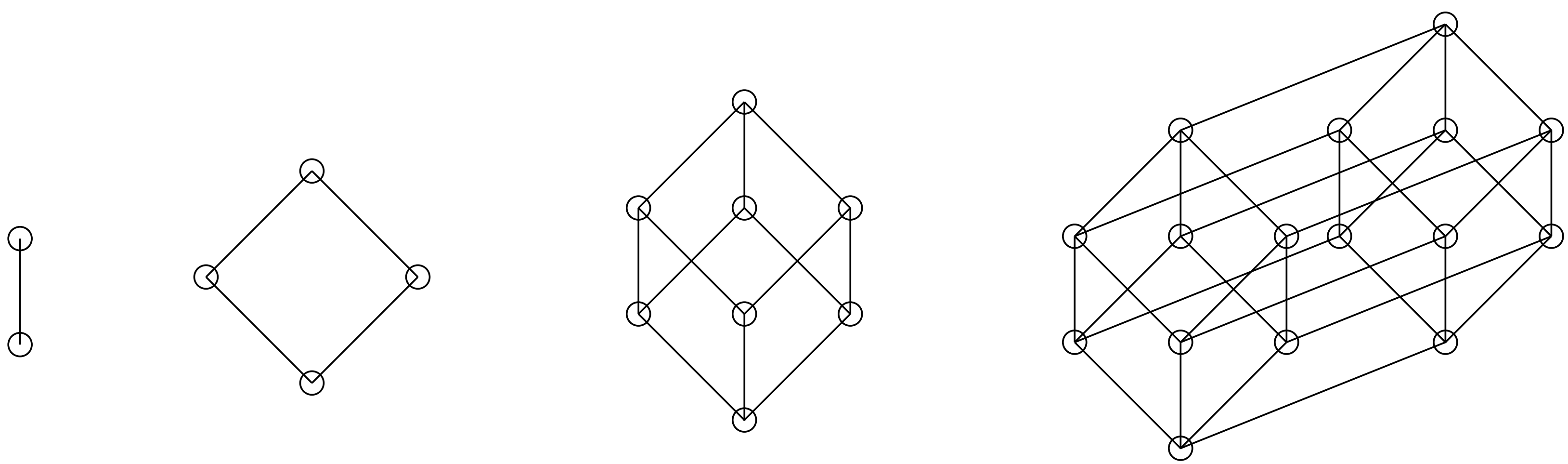
Itemsets and Bitvectors

itemset	bitvector	number
{juice, bread, milk }	$(1, 1, 1, 0, 0)$	7
{ potatos }	$(0, 0, 0, 0, 1)$	16
{bread, potatos }	$(0, 1, 0, 0, 1)$	18

- $\mathbb{X} = \{0, 1\}^d$ itemsets as bitvectors
- $|x| = \sum_{i=0}^{d-1} x_i$ size of itemset
- $\phi(x) = \sum_{i=0}^{d-1} x_i 2^i$ “number” of bitvector
- $d_H(x, y) = \sum_{i=1}^{d} |x_i - y_i|$ Hamming distance
- $x \leq y :\Leftrightarrow x_i \leq y_i$ (for all i) partial order
- There are 2^d different itemsets with d items

The Boolean Lattice of Itemsets





Probability Distribution

- $p : \mathbb{X} \to \mathbb{R}_+$ distribution, $\sum_{x \in \mathbb{X}} p(x) = 1$
- $P(A) = \sum_{x \in A} p(x)$ for $A \subset \mathbb{X}$
- $P(\mathbb{X}) = 1$, $P(\emptyset) = 0$, $P(A \cup B) \leq P(A) + P(B)$
- Sample probability, for $x_1, x_2, \ldots, x_n$:

$$P(A) = \frac{1}{n} \#\{i | x_i \in A\}$$

- Cumulative distribution function:

$$F(x) := P(\{y | y \leq x\})$$

- $F(1) = 1, \quad x \leq y \Rightarrow F(x) \leq F(y)$
- Dual cumulative distribution function:

$$F^{\partial}(x) := P(\{y | y \geq x\})$$

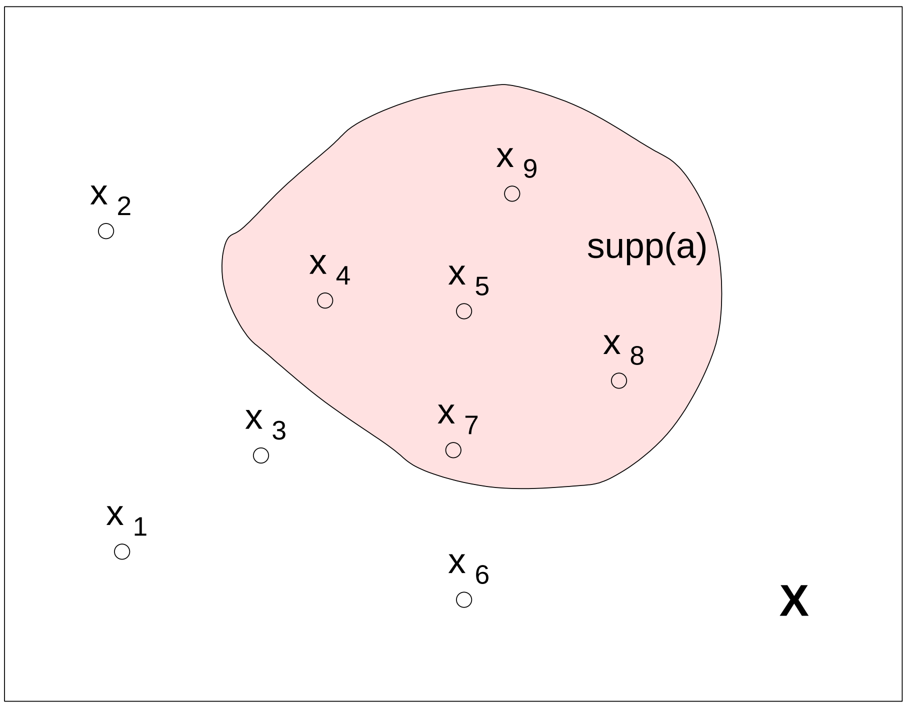
Rules and Predicates

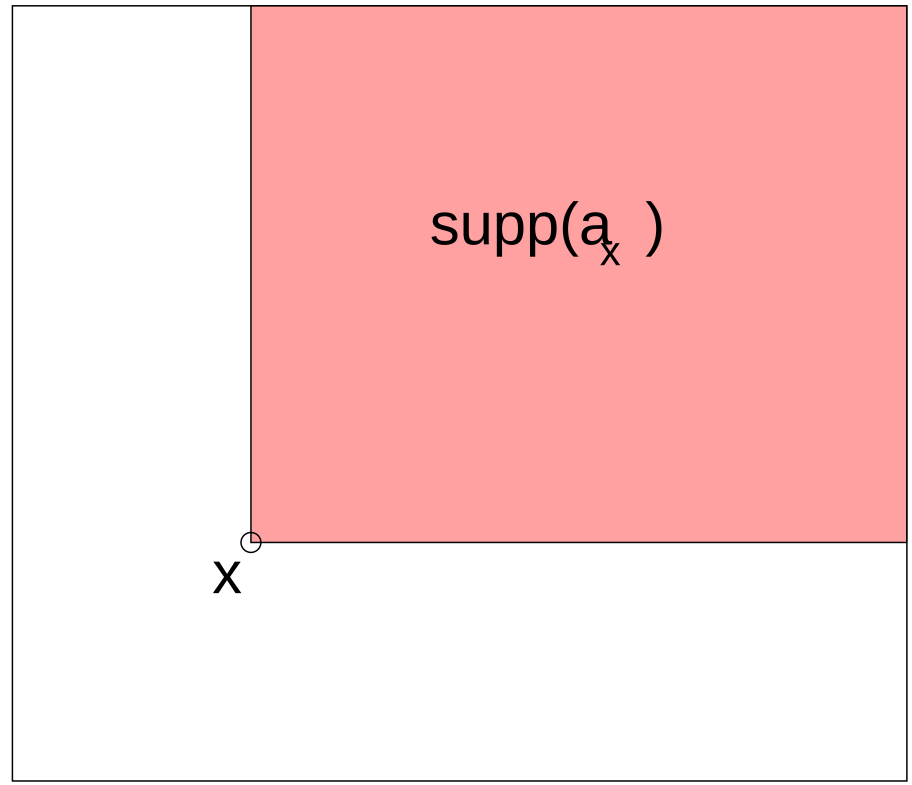
- $a : \mathbb{X} \longrightarrow \{0, 1\}$ predicate
- There are 2^{2^d} different predicates for itemsets of d items
- Example $d = 10,000$, ...
- $\mathrm{supp}(a) = \{x | a(x) = 1\}$ support of predicate a
- $s(a) = P(\mathrm{supp}(a))$ also called support
  Predicate with large support is more likely to be true
- A natural class of predicates:

$$a_y(x) = 1 \text{ if } \quad x \leq y \quad \text{and} \quad = 0 \text{ else}$$

- Antimonotone in y and monotone in x:
  If $y \leq z$ then $a_z(x) \leq a_y(x)$ but $a_x(y) \leq a_x(z)$ but
- $s(a_x) = F^{\partial}(x)$

Support of Predicates and Itemsets





Other properties of predicates

- For a sample distribution:

$$s(a) = \sum_{i=1}^{n} a(x_i)$$

- $s(a_x)$ is anti-monotone in x (as a_x is antimonotone and F is monotone)
- A predicate is a random variable with $E(a) = s(a)$ and $\mathrm{var}(a) = s(a)(1 - s(a))$
- Support of conjunction: $s(a \wedge b) \leq s(a)$
- Confidence $c(a \Rightarrow b) = s(a \wedge b)/s(a)$
- Conditional probability: $c(a \Rightarrow b) = P(b|a)$

Analysis of Apriori

- Mathematical Modeling – Lattice of Itemsets, Probability and Predicates

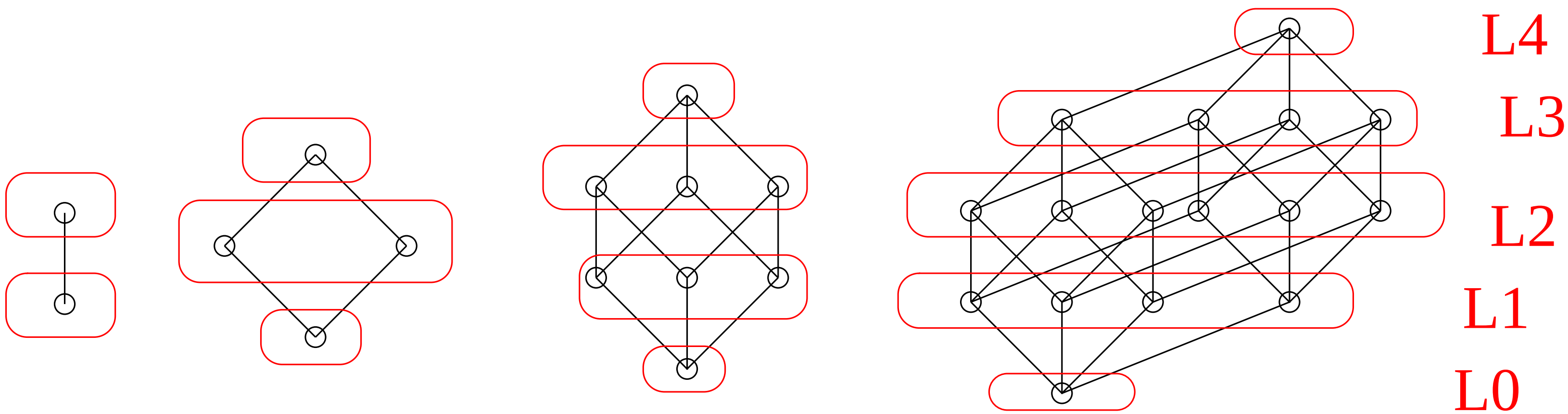
$\longrightarrow$ Algorithms – Search in Levelsets, Support

- Enumeration of k-Itemsets
- Bounds on the Number of Candidate Itemsets

Apriori Algorithm

$C_1 = \mathcal{A}(\mathbb{X})$ is the set of all one-itemsets, $k = 1$
while $C_k \neq \emptyset$ **do**
    scan database to determine support of all a_y with $y \in C_k$
    extract frequent itemsets from C_k into L_k
    generate C_{k+1}
    $k := k + 1$.

Levelsets



How to Determine the Support

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} (1,2) & (1,3,4) & (1,5) & (1,2,4) & (5) \end{bmatrix}$$

Algorithm for $z \leq x$ (time $T = (2|x| + |z|)\tau$):

1. Extract x into bitvector v: $v[x] \leftarrow 1$
2. Extract the values of v for the elements of z: $w \leftarrow v[z]$
3. If all components of $w = 1$ then $z \leq x$)
4. Set $v[x] \leftarrow 0$

n itemsets $x = x_i$ and m_k itemsets z of length k:
$T = \sum_k (2 \sum_{i=1}^n |x^{(i)}| + m_k k n)\tau \approx (\sum_k m_k k) n \tau$

Columnwise Storage

$$\left[(1,2,3,4) \quad (1,4) \quad (2) \quad (2,4) \quad (3,5)\right]$$

Algorithm to find $\#\{i | z \le x_i\}$:

1. $v[X_{z[0]}] \leftarrow 1$
2. $w[X_{z[1]}] \leftarrow v[X_{z[1]}]$
3. for $j = z[2], z[3], \ldots$
   (a) $v[X_{z[j-2]}] \leftarrow 0$
   (b) $v[X_{z[j]}] \leftarrow w[X_{z_[j]}]$
   (c) swap v and w
4. Get the support $s(a_z) = |w|$

Complexity:

$$T = 3\tau \sum_{j=1}^{d} \sum_{i=1}^{n} x_j^{(i)} z_j \approx \frac{3E(|x|)}{d} \sum_k m_k k n \tau$$

Analysis of Apriori

- Mathematical Modeling – Lattice of Itemsets, Probability and Predicates
- Algorithms – Search in Levelsets, Support

$\longrightarrow$ Enumeration of k-Itemsets

- Bounds on the Number of Candidate Itemsets

A Representation Lemma

For every $m, k \in \mathbb{N}$ there are numbers $m_s < \cdots < m_k$ such that

$$m = \sum_{j=s}^{k} \binom{m_j}{j}$$

and the m_j are uniquely determined by m.

Proof: see notes, uses induction and the identity

$$\binom{t+1}{r} - 1 = \sum_{l=1}^{r} \binom{t-r+l}{l}$$

Colexicographic Ordering

- Recall $\phi(x) = \sum_{i=0}^{d-1} x_i 2^i$ "number" of bitvector
- Colexicographic Ordering

$$y \prec z \Leftrightarrow \phi(y) < \phi(z)$$

- $y < z \Rightarrow y \prec z$
- Example: All two-itemsets with five items in colexicographic order: $(0,0,0,1,1), (0,0,1,0,1), (0,0,1,1,0), (0,1,0,0,1), (0,1,0,1,0), (0,1,1,0,0), (1,0,0,0,1), (1,0,0,1,0), (1,0,1,0,0), (1,1,0,0,0)$
- Let $[m] := \{0, \ldots, m-1\}$ and

$$[m]^{(k)} = \text{all } k\text{-itemsets using first } m \text{ items only}$$

are the first $\binom{m-1}{k}$ itemsets in colexicographic ordering

The first m k-itemsets

The set of the first m k-itemsets in colex ordering is

$$B^{(k)}(m_k, \ldots, m_s) := \bigcup_{j=s}^{k} [m_j]^{(j)} \vee e(m_{j+1}, \ldots, m_k)$$

where $C \vee y := \{z \vee y \mid z \in C\}$, the m_i are given by $b^{(k)}(m_k, \ldots, m_s) = m$ and $m_s < \cdots < m_k$

Proof:

1. Components of the union are disjoint as for $i < j$ one has $x \prec y$ if $x \in [m_i]^{(i)} \vee e(m_{i+1}, \ldots, m_k)$ and $y \in [m_j]^{(j)} \vee e(m_{j+1}, \ldots, m_k)$
2. If m_k is the highest bit set one gets:

$$B^{(k)}(m_k, \ldots, m_s) = [m_k]^{(k)} \cup B^{(k-1)}(m_{k-1}, \ldots, m_s) \vee e(m_k)$$

Analysis of Apriori

- Mathematical Modeling – Lattice of Itemsets, Probability and Predicates
- Algorithms – Search in Levelsets, Support
- Enumeration of k-Itemsets

$\longrightarrow$ Bounds on the Number of Candidate Itemsets

Simple Bounds

Apriori generates a sequence of sets of frequent k-itemsets L_k and candidate sets C_k

$$C_1 = L_1 \to C_2 \to L_2 \to C_3 \to L_3 \to C_4 \to \cdots$$

where C_k is the largest set of k-itemsets such that any subset z of a k-itemset in C_k is in $L_{|z|}$. Then

$$\sum_k |L_k| k \le \sum_k m_k k \le \sum_{k=0}^{d} \binom{d}{k} k$$

The upper bound is too pessimistic in most cases

Shadows and the Apriori Condition

The sequence $C_1, C_2, \ldots$ of itemsets is said to satisfy the apriori condition if

- C_k contains only k-itemsets
- $x \in C_k$ and $y < x$ then $y \in C_{|y|}$

The candidate sets generated by the apriori algorithm satisfy the apriori condition

The *shadow* of a set of k-itemsets C_k is defined as

$$\partial C_k := \{x \mid |x| = k - 1 \text{ and } x < z \text{ for some } z \in C_k\}$$

The sequence $C_1, C_2, \ldots$ satisfies the apriori condition $\Leftrightarrow$ $\partial C_k \subset C_{k-1}$ for all k

The Inverse Problem

Is it possible to choose something smaller than the maximal set which satisfies the apriori condition?

No, as for any sequence C_k satisfying the apriori condition there is a data set D and $\sigma > 0$ such that the C_k are the sets of frequent k-itemsets of D with support σ

Proof:
Choose $C \subset \bigcup_k C_k$ to be the set of all maximal itemsets, and $\sigma \leq 1/|C|$ and the database D be any sequence of elements which contains exactly every element of C once
Then the maximal itemsets are frequent and (by the apriori property) so are all the subsets of the maximal itemsets

What about $\sigma > 1/\#\{\text{maximal elements}\}$?

The shadow of $B^{(k)}(m_k, \ldots, m_s)$

$$\partial B^{(k)}(m_k, \ldots, m_s) = B^{(k-1)}(m_k, \ldots, m_s)$$

Proof:

- Case $s = k$

$$\partial [m_k]^k = [m_k]^{(k-1)}$$

- By recursion for $B^{(k)}(m_k, \ldots, m_s)$ and additivity of the shadow one gets $\partial B^{(k)}(m_k, \ldots, m_s) =$

$$[m_k]^{(k-1)} \cup \left(\partial B^{(k-1)}(m_{k-1}, \ldots, m_s)\right) \vee e_{m_k}$$

Compressing sets of k-itemsets

Idea: map any C_k close to $B^{(k)}(m_k, \ldots, m_s)$

- Compression of bitvector:

$$R_{ij}(z) = \begin{cases} z - e_j + e_i & \text{if } e_i \not\leq z \text{ and } e_j \leq z \\ z & \text{else} \end{cases}$$

- Not injective as $R_{ij}(y) = R_{ij}(R_{ij}(y))$
- Compression of set C of itemsets:

$$\tilde{R}_{i,j}(C) = R_{ij}(C) \cup (C \cap R_{ij}^{-1}(C)).$$

  add itemsets which remain in C after compression
- Compression Lemma: $\partial \tilde{R}_{i,j}(C) \subset \tilde{R}_{i,j}(\partial C)$
- C is compressed if $\tilde{R}_{i,j}(C) = C$ for all i, j

The Kruskal/Katona Theorem

For any k-itemset C with $|C| = b^{(k)}(m_k, \dots, m_s)$:

$$|\partial C| \geq b^{(k-1)}(m_k, \dots, m_s)$$

Proof:

- Compression reduces size of shadow
- Double induction over k and $m = |A|$
- $k = 1$ and any m (as A is compressed): $A = \{e_0, \dots, e_{m-1}\}$ thus $\partial A = \{0\}$
- $m = 1$ andy any k: $A = \{e(0, \dots, k-1)\}$ thus $\partial A = [k]^{(k-1)}$
- Rest of proof a bit technical. Idea: Partition $A = A_0 \cup A_1$ where A_0 contains elements with bit 0 not set. Induction considering different cases

Bounding the Candidate Itemsets

If C_k satisfies apriori property, $|C_k| = b^{(k)}(m_k, \ldots, m_s)$ and $p \leq s$ then

$$|C_{k+p}| \leq b^{(k+p)}(m_k, \ldots, m_s)$$

Proof:

- Assume $|C_{k+p}| > b^{(k+p)}(m_k, \ldots, m_s)$
- Then, by Kruskal-Katona:

$$|C_k| \geq b^{(k)}(m_k, \ldots, m_s, s + p - 1)$$

- However, one can see that

$$|C_k| < b^{(k)}(m_k, \ldots, m_s, s + p - 1)$$

Bound is tight, Geerts et al 2001

Performance Improvements

→ Data Distribution and Access

- Association Rules with Constraints
- Frequent Pattern Trees

Apriori TID: Transforming the Database

$$\left[\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{array}\right] \to \left[\begin{array}{cccccc} (1,2) & (1,3) & (1,5) & (2,3) & (2,5) & (3,5) \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \to \left[\begin{array}{c} (2,3,5) \\ \hline 0 \\ 1 \\ 1 \\ 0 \end{array}\right]$$

- “New items” = itemsets in L_k
- Larger sparsity, less columns for higher k
- Bound on expected time:

$$E(T) \leq 3n \frac{E(|x|)}{d} \tau \sum_k m_k$$

Analysis like for the case of determination of support

Partition: Reducing the Number of Scans

- Partition database: $DB = DB_1 \cup \cdots \cup DB_p$
- *Invariant partitioning property*:

$$s(A; DB) \leq \max_j \; s(A, DB_j)$$

  where $s(A; DB_j)$ is support of A in DB_j

- Algorithm "Partition":
  1. First DB scan: Generate all $L_k(DB_j)$
  2. Candidates: $C_k := \bigcup_j L_k(DB_j)$
  3. Second DB scan: Counts for all the C_k
- Applications: Parallel computing, very large data set, distributed data

A. Savasere, E. Omiecinski, and S. Navathe, *An efficient algorithm for mining association rules in large databases*, VLDB '95, pp. 432–443.

Performance Improvements

- Data Distribution and Access

$\longrightarrow$ Association Rules with Constraints

- Frequent Pattern Trees

Mining Items with Taxonomies

- Multiple taxonomies on items: brands/categories/product groups, sale
- Rules involving ancestors have higher support
- Model multiple taxonomies with a DAG
- Basic: include ancestors in transactions
- Normalise: Remove ancestors in frequent itemsets
- Lemma: Elements of L_k normalised $\Rightarrow$ elements of C_{k+1} are

R. Srikant and R. Agrawal, *Mining generalized association rules*, VLDB '95, pp. 407–419.

Why Constraints?

- Association rule mining process:
  1. User selects data
  2. User selects support/confidence thresholds
  3. System runs data-intensive mining
  4. System returns large numbers of rules
  5. User searches for useful information
- Problems with this approach:
  - Lack of user exploration and control – user cannot change query during mining stage
  - Lack of focus – user cannot specify candidate rules of interest

$\Longrightarrow$ Constraints for better focus and interaction

What are constraints

- Example: Price limited market baskets:

$$C(A) \quad := \quad \sum_{a \in A} c_a \leq c_{\mathrm{max}}$$

- A *constraint* C is a predicate defined on itemsets, i.e.,

$$C : 2^I \rightarrow \{T, F\}$$

  (2^I: powerset of set of items I)

- *Constrained association rules:* Association rules $A \rightarrow B$ where antecedent and consequent satisfy constraints $C_a(A)$ and $C_c(B)$ respectively

Two simple methods

- Constraints on frequent itemsets
- Trivial and sound approach (Apriori+):
  1. Find all frequent itemsets with Apriori
  2. Remove ones which do not satisfy constraints

Apriori does not make use of constraints

- Naive pushing constraints into Apriori:
  - Use constraints to prune candidate k-itemsets
  - *Can give wrong results!*
    Example: Average item price bound may not hold for frequent subsets of frequent itemset satisfying bound

Two Types of Constraints

- C is *antimonotone* iff

$$(A \subset B) \wedge C(B) \Rightarrow C(A)$$

  - Example: Prize of market basket $\leq c_{\mathrm{max}}$
  - Naive Pushing gives correct results

- C is *monotone* iff

$$(A \subset B) \wedge C(A) \Rightarrow C(B)$$

  - Example: Prize of market basket $\geq c_{\mathrm{min}}$
  - Trivial Algorithm, saving in checking constraints

R. Ng, L. Lakshmanan, J. Han, and A. Pang, *Exploratory mining and pruning optimizations of constrained associations rules*, SIGMOD 1998, pp. 13–24.

Performance Improvements

- Data Distribution and Access
- Association Rules with Constraints

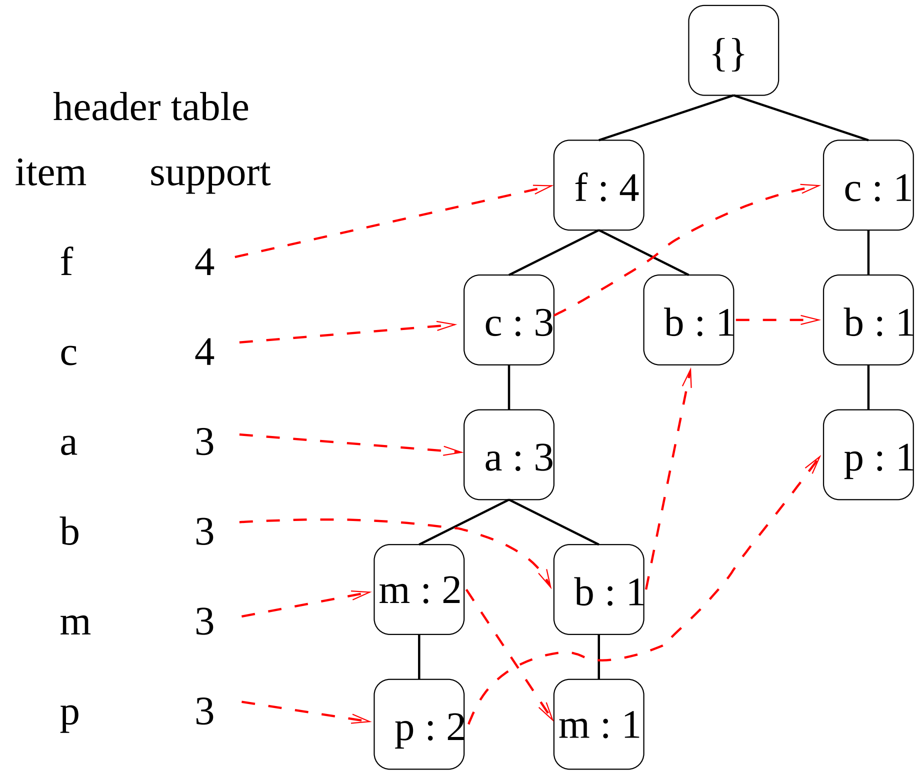
$\longrightarrow$ Frequent Pattern Trees

Limitations of the Apriori algorithm

- Large numbers of frequent itemsets are expensive: $\mathbf{10^6}$ frequent 1-itemsets require testing of $\mathbf{5 * 10^{11}}$ candidate 2-itemsets
- No good for long patterns: A frequent itemset of size $\mathbf{100}$ requires testing of $\mathbf{2^{100} \approx 10^{30}}$ smaller candidate itemsets
- Repeated scans of the DB are expensive
- Bottleneck: Candidate generation mechanism

DB compression in FP-tree

items	$s > 0.5$
f, a, c, d, g, i, m, p	f, c, a, m, p
a, b, c, f, l, m, o	f, c, a, b, m
b, f, h, j, o, w	f, b
b, c, k, s, p	c, b, p
a, f, c, e, l, p, m, n	f, c, a, m, p



- 2 scans of DB to determine frequent 1-itemsets and build FP-tree

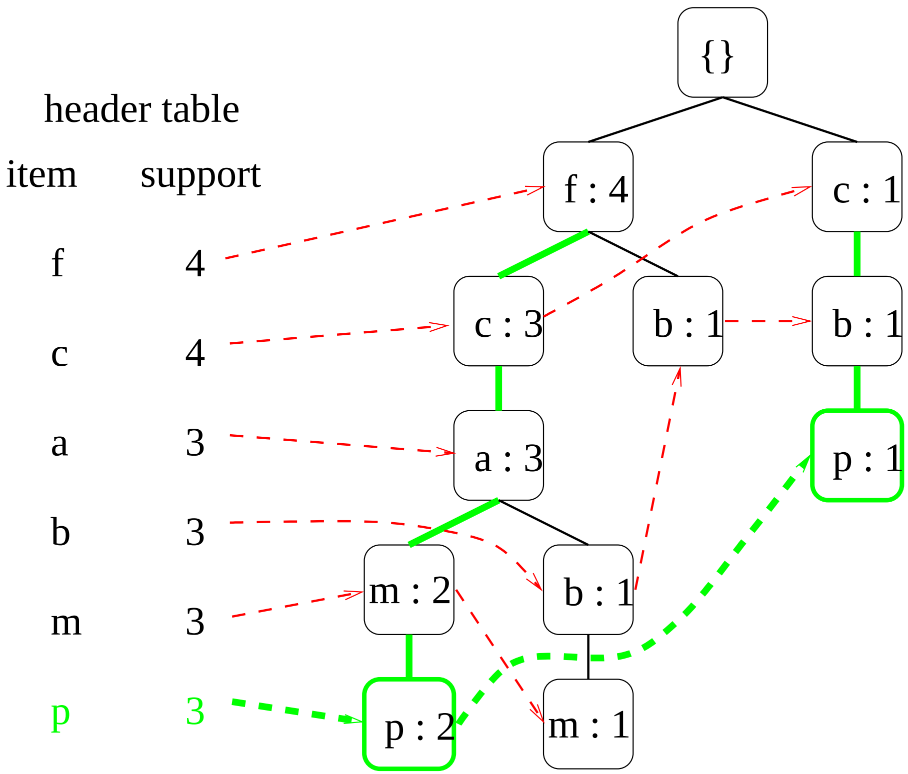
J. Han, J. Pei, and Y. Yin, *Mining frequent patterns without candidate generation*, 2000 ACM SIGMOD Intl. Conference on Management of Data, pp. 1–12.

Benefits of the FP-tree Structure

- Completeness
  - Never breaks a long pattern of any transaction
  - Contains all information for frequent pattern mining
- Compactness
  - Removing infrequent items
  - Items frequent $\Rightarrow$ likely shared
  - Never larger than original database (+ links)
  - Compression ratios of over 100 observed

From J.Han and J.Pei: Sequential Pattern Mining, PAKDD 2001

Conditional Pattern-Bases



item	conditional pattern base
c	$f : 3$
a	$fc : 3$
b	$fca : 1,\ f : 1,\ c : 1$
m	$fca : 2,\ fcab : 1$
p	$fcam : 2,\ cb : 1$

- Frequent patterns of DB are frequent patterns of a conditional pattern base
- Ordering removes some redundancy

Conditional FP-trees

item	conditional pattern base	conditional FP-tree
c	$f:3$	$(f:3)$
a	$fc:3$	$(f:3) - (c:3)$
b	$fca:1,\ f:1,\ c:1$	$\emptyset$
m	$fca:2,\ fcab:1$	$(f:3) - (c:3) - (a:3)$
p	$fcam:2,\ cb:1$	$(c:3)$

- Frequent patterns of DB from conditional FP-trees
- Apply recursively
- Tree = path $\Rightarrow$ all subsets frequent
- Separately mine prefix path and rest and combine