Association Rules

- Consider an $n \times d$ binary matrix, where the columns are random variables and the rows are observations. E.g. saleable items and supermarket customers, web pages and visitors, or words and documents:

Iraq	*Korea*	*nuclear*	*mass*	*quantum*	*gravity*
1	1	1	1	0	0
1	0	1	1	0	0
0	1	1	0	0	0
0	1	1	0	0	0
0	0	1	1	1	0
0	0	1	1	0	1
0	0	0	1	1	1

- Typically very sparse

Association Rules

- Task: find all "interesting" rules of the form

$$\{ A_1, A_2, \ldots, A_p \} \implies B,$$

where $\{ A_j \}$ is a subset of the variables, and B is another variable.

- Intuitive meaning: if $A_1, \ldots, A_p$ appear on a row, then B is also likely.
  - E.g. if a document mentions *Iraq* and *weapons*, it will mention *mass*.

Association Rules

- Formal semantics: the rule's **confidence** is

$$P(B \mid A_1, \ldots, A_p) = \frac{f(A_1, \ldots, A_p, B)}{f(A_1, \ldots, A_p)}.$$

- Problem: not all rules with high confidence are interesting; consider
  - a dictionary that lists lots of unrelated words
  - this presentation: I mention *Iraq*, *quantum*, and *supermarket*, so if no-one else mentions both *Iraq* and *quantum*, the rule

$$\{\, \textit{Iraq}, \textit{quantum} \,\} \implies \textit{supermarket}$$

    has 100% confidence but is not really interesting

- Solution: a rule is interesting if it has both high confidence and high **support.**

Association Rules

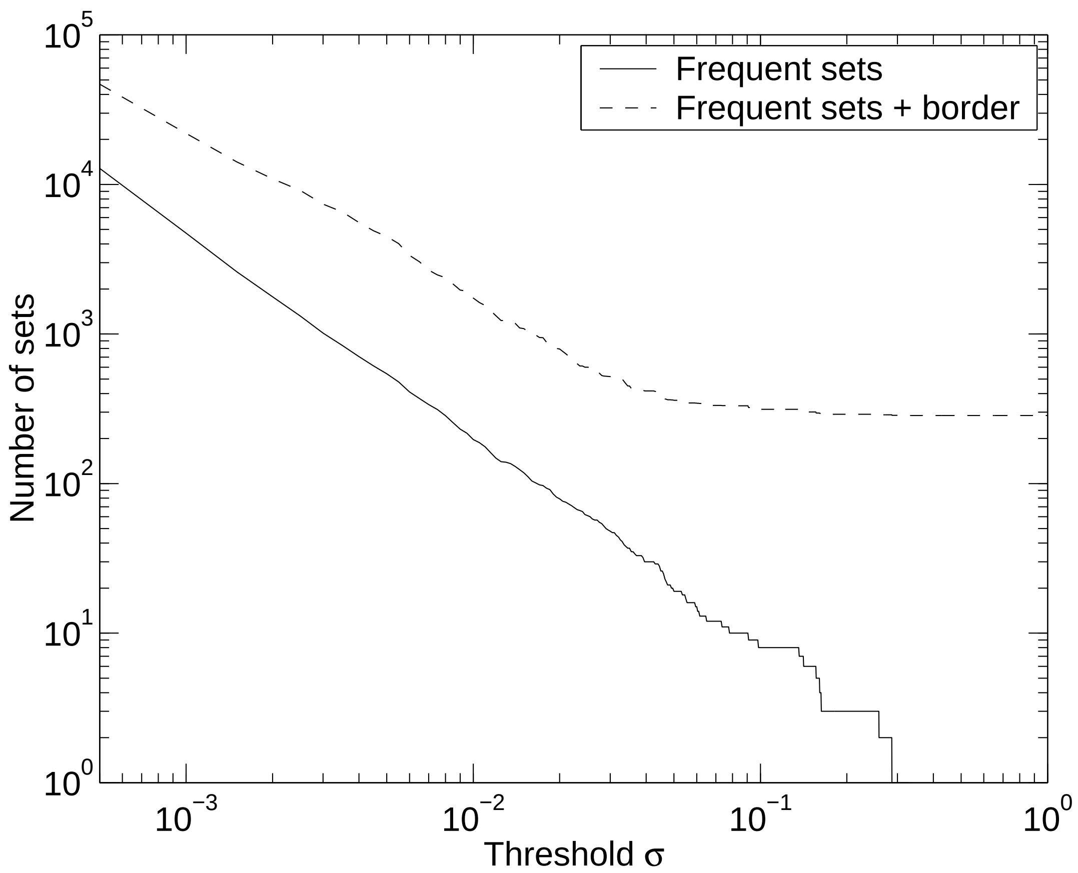
- The rule's **support** is the fraction of the data where the rule holds,

$$P(A_1, \ldots, A_p, B) = \frac{f(A_1, \ldots, A_p, B)}{n}.$$

- Now the task is to find all rules that have support over some threshold σ and confidence over some threshold γ.

Frequent Itemsets

- The association rule mining task can be reduced to finding **frequent itemsets**, i.e., sets of variables that have support $\geq \sigma$.
  - Given such a set X, we can try all rules of the form $X \setminus \{ B \} \implies B$.
- If σ is too small, there are 2^d frequent itemsets.
  - There can be no polynomial-time algorithm that finds all frequent itemsets from arbitrary data with arbitrary σ: even outputting the result will take exponential time.
  - Of course, no-one would even want to have a list of every subset of the variables.
  - A suitable value of σ depends on the data.



The Apriori Algorithm

- Suppose we know that a subset X of the variables has support s. What can we say about the supports of sets $Y \neq X$ **a priori**, without looking at the data?
  - If $Y \subset X$, the support of Y is necessarily $\geq s$.
  - Conversely, if $Y \supset X$, the support of Y is $\leq s$.
  - ➜ Support, as a set function, is **antimonotonic.**
- Therefore: if we know that X is not frequent, we can rule out all $Y \supset X$.
- Breadth-first search: let first $j \leftarrow 1$, then iterate
  1. Form j-element candidate sets.
  2. Test which candidate sets are frequent. (database scan)
  3. $j \leftarrow j + 1$.

The Apriori Algorithm

- How to form candidate sets?
- If $j = 1$, simple: all 1-element sets are candidates, since **a priori** we know nothing about them.
- In general case, we have a family of $j - 1$-element sets and we must find all j-element sets whose all immediate subsets are in our family.
- Simple solution:
  - Keep candidate and frequent set families always in lexicographic order.
  - When two frequent sets differ only in the last element, make their union a "precandidate" and check whether all its subsets are frequent.
  - E.g. $ABCD, ABCE, ABCF, \ldots$ yields $ABCDE, ABCDF, ABCEF$ but $ABDE, ACDE, BCDE$, etc., have to be checked.

The Apriori Algorithm

- There are much more sophisticated algorithms than Apriori, but on many real-world data sets, with realistic amounts of resulting frequent itemsets, Apriori is as good as any of the advanced algorithms.

- **Data mining** usually implies very large data sets; then the database pass dominates the time taken by the algorithm.

- Fundamental problem: if a, say, 20-element set is frequent, then all of its $1\,048\,575$ subsets are frequent.

- There are algorithms for mining **maximal** frequent itemsets, i.e., only the 20-element set would be mined, and not all of its subsets would be considered.

Back to Frequent Itemsets

- Frequent itemsets were originally invented for mining association rules, but they can also be useful in themselves.

- In the supermarket setting, association rules may be what we want: if $A \implies B$ but not $B \implies A$, perhaps this is related to the direction in which customers walk in the store.

- In the document setting, perhaps itemsets are more interesting.

- Both association rules and frequent itemsets are **local** descriptions about the data: there are some rows in the data matrix where the rule holds. Even though support and confidence can be defined in the language of probability, a collection of itemsets does not form a model of the data (at least not in any obvious way).

Using Frequent Itemsets to Answer Boolean Queries

- With huge data and a reasonable σ, the collection of frequent itemsets may be much smaller than the data.
- How to take advantage of itemsets?
- In addition to the Apriori rule $X \subset Y \implies f(X) \geq f(Y)$, other forms of deduction are possible:

$$f(X) = f(X \cup \{ A \}),\ A \notin Y \supset X \implies f(Y) = f(Y \cup \{ A \})$$
$$f(A \vee B \vee C) = f(A) + f(B) + f(C) - f(AB) - f(BC) - f(CA) + f(ABC)$$

Using Frequent Itemsets to Answer Boolean Queries

- The support of an arbitrary Boolean formula can be represented as a sum of supports itemsets. E.g.:
  - $f(A \wedge \neg B) = f(A) - f(AB)$
  - $f(A \vee \neg B) = 1 - f(B) + f(AB)$
  - $f(AB \vee BC \vee CA) = f(AB) + f(BC) + f(CA) - 2f(ABC)$

Using Frequent Itemsets to Answer Boolean Queries

- A possible way to approximate the support of a formula is to use the itemsets whose supports are frequent (and thus known), and just forget the unknown terms.
  - Proving error bounds a largely open question.
  - A known special case (Bonferroni's inequality): in an inclusion-exclusion like

$$f(A \vee B \vee C) = f(A) + f(B) + f(C) - f(AB) - f(BC) - f(CA) + f(ABC),$$

    if the known itemsets happen to be cut off at a level, the error is bounded by the next level:

$$[f(A \vee B \vee C) - (f(A) + f(B) + f(C))] \leq f(AB) + f(BC) + f(CA) < 3\sigma,$$

$$[f(A \vee B \vee C) - (f(A) + f(B) + f(C) - f(AB) - f(BC) - f(CA))] \leq f(ABC) < \sigma$$

Using Frequent Itemsets to Answer Boolean Queries

- A solution that is (in a way) general:
  - Consider the linear space whose basis vectors correspond to each possible row of the matrix (i.e., 2^d dimensions).
  - With a data matrix, associate the vector in this space whose coordinates are the relative frequencies of the corresponding rows in the matrix.
  - E.g., if 10% of the rows are $[1001]$, the corresponding coordinate is 0.1.
  - Now a query corresponds to an inner product with a 0–1 vector.
  - Knowledge about frequent itemsets translates into equality constraints on inner products.
  - Knowledge that an itemset is not frequent translates into an inequality constraint.
  - Linear programming yields optimal bounds on the support of a given query.
  - Drawback: complexity

Using Frequent Itemsets to Create a Model

- Modeling: approximating the joint distribution
- Frequent itemsets constrain the joint distribution somewhat, but do not (usually) determine it completely.
- Principle of Insufficient Reason: choose the distribution that has maximum entropy among all distributions that satisfy the constraints.
  - The elements of the probability space are the possible rows of the data matrix; thus, 2^d elements.
  - If a distribution p gives probability $p(x)$ to element x, its entropy is

$$H(p) = -\sum_{x} p(x) \log p(x).$$

Maximum Entropy Models

- Constraints: $f(X_j) = s_j$ for all frequent itemsets X_j
- Objective: maximize $H(p) = -\sum_x p(x) \log p(x)$
- The maximum entropy distribution has form

$$p(x) = \mu_0 \prod_j \mu_j^{1[X_j \subseteq x]}$$

  - $1[X_j \subseteq x]$ is a 0–1 function that is 1 whenever the constraint $f(X_j) = s_j$ applies to x
  - each μ_j is a constant that can be approximated from the data
  - μ_0 normalizes the distribution

Maximum Entropy Models

- Iterative Scaling algorithm
  - The distribution is represented explicitly as a vector $\vec{p}$ of 2^d elements.
  - Initialize to the uniform distribution $p(x) = 2^{-d}$, then iterate:
    * for each constraint $f(X_j) = s_j$:
      · find current $f(X_j) = \sum_{x \supseteq X_j} p(x)$
      · multiply each term in this sum by $s_j / f(X_j)$
      · multiply all other elements in the vector $\vec{p}$ by a number such that $\sum_x p(x) = 1$
    * test for convergence, stop if converged

Bibliography

- Agrawal, Imielinski, Swami: Mining Association Rules between Sets of Items in Large Databases. SIGMOD 1993.
- Mannila, Toivonen, Verkamo: Efficient algorithms for discovering association rules. KDD 1994.
- Zheng, Kohavi, Mason: Real World Performance of Association Rule Algorithms. KDD 2001.
- Mannila, Toivonen: Multiple Uses of Frequent Sets and Condensed Representations (Extended Abstract). KDD 1996.
- Pavlov, Mannila, Smyth: Beyond independence: Probabilistic models for query approximation on binary transaction data. UCI technical report ICS TR-01-09, 2001.