Machine Learning

Association Rules - Beyond Apriori
Optional

Claudio Sartori

DISI
Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

Context

- Apriori is a classic algorithm for mining frequent itemsets and association rules.
- Several alternative algorithms improve on:
  - runtime efficiency,
  - memory usage,
  - output compactness.
- We will overview:
  - key alternative algorithms,
  - a decision tree for choosing among them,
  - a comparison table (runtime, memory, output size).

Horizontal vs Vertical Format

Horizontal format

- Each transaction lists the items it contains.

Example

- $T1 = \{A, B, C\}$
- $T2 = \{A, C\}$
- $T3 = \{B, D\}$

Vertical format

- Each item is stored with the list of transaction IDs (TIDs) in which it appears.

Example

- $A : \{T1, T2\}$

Why Use the Vertical Format?

Key idea

- In vertical format, support of an itemset is computed by intersecting TID-lists.

Example

- $\text{TID(A)} = \{\text{T1, T2}\}$
- $\text{TID(B)} = \{\text{T1, T3}\}$
- $\text{TID}(\text{A} \cap \text{B}) = \{\text{T1}\}$
- So support($\{$A,B$\}$) $= 1$.

Advantages

- Fast support counting via set intersection.
- Suitable for dense datasets.

Algorithms Using the Vertical Format

Eclat

- Uses TID-lists for items and itemsets.
- Mines frequent itemsets by recursive intersection of TID-lists.

dEclat

- Stores diffsets (transactions where a pattern does not occur).
- Reduces memory usage on dense datasets.

Charm

- Mines closed frequent itemsets.
- Uses vertical intersections plus closure checks to prune search.

Summary: What is the Vertical Format?

- In the horizontal format, rows correspond to transactions and contain lists of items.
- In the vertical format, rows correspond to items or itemsets and store the TID-lists.

One-sentence definition

- The vertical format represents each item or itemset by the set of transaction IDs in which it appears, enabling support computation using TID-list intersections.

FP-Growth

FP-Growth (Frequent Pattern Growth)

- Key idea: avoid explicit candidate generation.
- Compress transactions into an FP-tree.
- Recursively mine conditional FP-trees (pattern-growth).

Advantages

- Often much faster than Apriori on large datasets.
- Fewer scans of the database.
- Works well for both sparse and moderately dense data.

Eclat and dEclat

Eclat

- Uses vertical database layout (TID-lists).
- Each itemset represented by the set of transaction IDs (TIDs) where it appears.
- Frequent itemsets found by intersecting TID-lists.

dEclat (Diffset-based)

- Stores diffsets: transactions where a pattern does not occur.
- Reduces memory usage for dense datasets.

When to use

- Dense datasets or moderate number of items.
- Vertical layout is feasible and fits in memory.

Closed Itemset Mining: Charm and CLOSET+

Closed frequent itemsets

- An itemset is closed if no strict superset has the same support.
- Closed sets provide a more compact yet lossless representation.

Charm

- Vertical format (similar to Eclat).
- Uses closure properties to prune search.
- Efficiently enumerates closed frequent itemsets.

CLOSET / CLOSET+

- Pattern-growth approach (similar to FP-Growth).
- Restricts mining to closed itemsets only.

Other Alternatives: H-Mine, RARM, SON, AIS/SETM

H-Mine

- Uses a dynamic hyperlinked structure (H-struct).
- Efficient for sparse data and incremental mining.

RARM (Rapid Association Rule Mining)

- Heuristic approach.
- Quickly finds high-confidence rules without full enumeration.
- Good for exploratory analysis when exact completeness is not required.

Other Alternatives: H-Mine, RARM, SON, AIS/SETM (2)

SON algorithm

- MapReduce-style parallelization of frequent itemset mining.
- Each data chunk mined locally; candidate sets combined globally.
- Suitable for distributed environments (Hadoop, Spark).

AIS and SETM

- Historical algorithms preceding Apriori.
- Based on candidate generation with higher overhead.
- Now mostly of historical or pedagogical interest.

Non-classical Alternatives

Although not direct drop-in replacements, some modern methods approximate association structures.

Neural embedding models

- Item or transaction embeddings (similar to Word2Vec).
- Variational autoencoders or deep factorization models.
- Focus: prediction and recommendation rather than exhaustive rule enumeration.

Probabilistic models

- Bayesian networks, Markov random fields.
- Model conditional dependencies between variables.
- Often used for inference and prediction rather than enumerating all

Choice: Data Size and Structure

Step 1: Check data scale

- Is your dataset very large (many transactions, many items)?
  - If data is distributed (Hadoop / Spark):
    - Use SON or distributed FP-Growth variants.
  - If data is not distributed:
    - Use FP-Growth or H-Mine.
- If dataset is not very large:
  - Is the dataset dense (many items per transaction)?
    - Yes: use Eclat or dEclat, or Charm if closed sets are enough.
    - No (sparse): use FP-Growth or H-Mine.

Choice: Output and Performance Requirements

Step 2: Output compactness

- Do you need compact, non-redundant output?
  - Yes: use Charm or CLOSET+ (closed itemsets).
  - No: any frequent itemset miner is fine.

Step 3: Speed and approximation

- Do you need very fast, approximate results?
  - Yes: consider RARM or heuristic / sampling-based methods.

Step 4: Summary

- Large + distributed: SON.
- Large + local: FP-Growth or H-Mine.
- Dense datasets: Eclat/dEclat.
- Need compact patterns: Charm/CLOSET+.

Comparison Table: Runtime, Memory, Output Size

Algorithm	Runtime	Memory	Output size
Apriori	Medium to High	Medium	High (all frequent itemsets)
FP-Growth	Low (fewer scans)	Medium (FP-tree)	High (all frequent itemsets)
Eclat	Low on dense data	Medium to High (TID-lists)	High (all frequent itemsets)
dEclat	Low on dense data	Medium (diffsets)	High (all frequent itemsets)
Charm	Medium	Medium to High	Low to Medium (closed sets only)
CLOSET+	Low to Medium	Medium	Low to Medium (closed sets only)
H-Mine	Low on sparse data	Low to Medium	High (all frequent itemsets)
SON (distributed)	Low wall-clock (parallel)	Distributed across cluster	High (all frequent itemsets)
RARM (heuristic)	Very low	Low to Medium	Medium (high-confidence rules only)

Note: qualitative comparison; actual performance depends on data size, sparseness, and implementation details.

Takeaways

- Apriori is simple but can be inefficient on large or dense datasets.
- FP-Growth, Eclat, and H-Mine are common practical alternatives.
- Charm and CLOSET+ are preferred when you want compact, closed itemsets.
- SON is suitable for distributed big data.
- RARM and other heuristics are useful for fast, approximate mining.