Association Rules

Finding Structure in High-Dimensional Binary Data

Sundong Kim

June 9, 2026

R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in *VLDB*, 1994

J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann, 2011, Chapter 6 (Mining Frequent Patterns, Associations, and Correlations).*

Also referred to lecture note by Prof. Dr. İlyas Çiçekli

What Have We Done Without a Response Y?

A common theme: find **low-dimensional / latent structure** hidden behind high-dimensional X, using no labels.

Method	**Latent structure it finds**	**Relation to $P(X)$**
PCA	linear directions (max variance)	2nd-moment structure
Matrix completion	low-rank latent factors	low-rank assumption
K-means / hier.	discrete latent groups (clusters)	modes / dense regions
Autoencoder	nonlinear latent code	learned representation
VAE	probabilistic generative model	models $p(x)$

- These all act on *continuous / real-valued* data.
- **Next:** a tool for a very different data type — large, high-dimensional **binary** data.

Association Rules

- *Association rule analysis* finds structure in large, high-dimensional **binary** datasets — which items tend to appear *together*.
- The classic setting is *market-basket analysis*:
  - each transaction (a shopping basket) records which of p items were purchased,
  - we want to know which items tend to be bought *together*.
- Goal: find groups of items that *frequently co-occur*, and rules of the form "if a customer buys A, they also tend to buy B."
- As before, this is about the structure of the data — now the "modes" are *high-probability item combinations*.

Frequent Pattern Mining

TID	Items
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk
6	Bread, Diaper, Milk

- A *frequent pattern* is a set of items that appears *often* across transactions.
- Scanning the table, {Beer, Diaper} shows up in 3 of 6 baskets — a candidate frequent pattern.
- Market-basket data is naturally **binary**: each basket is a 0/1 vector over all items (present / absent).
- Two things we want to extract:
  - *frequent itemsets* — which combinations recur,
  - *association rules* — "if Beer & Diaper, then . . . ".

- Finding the recurring combinations first, then turning them into rules — that is the plan for the rest of this section.

Items, Itemsets, and Rules

TID	Items
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk
6	Bread, Diaper, Milk

- *Items*: Bread, Milk, Beer, Diaper, ...
- A *transaction* t is one basket — e.g. TID 2 is {Beer, Bread, Diaper, Eggs}.
- An *itemset* is any subset of items, e.g. {Beer, Diaper} (appears in TIDs 2,3,4).
- An *association rule* $A \Rightarrow B$ (A, B itemsets, $A \cap B = \varnothing$):

$$\{\text{Diaper}\} \Rightarrow \{\text{Beer}\}$$

"baskets with diapers tend to contain beer."

- We will use this same 6-transaction table throughout. Next: numerical measures to decide which rules are *interesting*.

Three Measures: Support, Confidence, Lift

For a rule $A \Rightarrow B$, with N total transactions:

- **Support** — how often the itemset appears:

$$\mathrm{supp}(A) = \frac{\#\{t : A \subseteq t\}}{N} \approx P(A).$$

  This is just an estimate of a joint probability.

- **Confidence** — a conditional probability:

$$\mathrm{conf}(A \Rightarrow B) = \frac{\mathrm{supp}(A \cup B)}{\mathrm{supp}(A)} \approx P(B \mid A).$$

- **Lift** — strength relative to independence:

$$\mathrm{lift}(A \Rightarrow B) = \frac{\mathrm{supp}(A \cup B)}{\mathrm{supp}(A)\,\mathrm{supp}(B)} \approx \frac{P(A \cap B)}{P(A)\,P(B)}.$$

On our table, {Diaper} ⇒ {Beer} (Diaper in 5 baskets, Beer in 3, both in 3):

$\mathrm{supp} = \frac{3}{6} = \frac{1}{2}$, $\mathrm{conf} = \frac{3}{5} = 0.6$, $\mathrm{lift} = \dfrac{1/2}{(5/6)(1/2)} = \dfrac{6}{5} = 1.2\ (> 1)$.

Interpreting the Measures

- **Support** measures how *common* the pattern is. Low-support rules describe rare combinations and are often ignored (and hard to estimate reliably).
- **Confidence** measures how *reliable* the implication is, but it ignores how common B is on its own.
- **Lift** corrects for this:
  - $\text{lift} = 1$: A and B are independent ($P(A \cap B) = P(A)P(B)$) — no association.
  - $\text{lift} > 1$: A and B co-occur *more* than expected by chance — positive association.
  - $\text{lift} < 1$: they co-occur *less* than expected — negative association.
- All three are statements about the joint distribution $P(X)$ — support is a joint probability, confidence a conditional, lift a departure from independence.

Stepping Back: Association Rules as Density Estimation

- Support, confidence, and lift are all estimates of pieces of the joint distribution $P(X_1, \ldots, X_p)$ over p binary items.
- But that joint has 2^p cells — far too many to estimate in full. So we lower our ambition: **find the regions of X-space where the probability mass is unusually high.**
- In other words, association rule mining is a form of *mode-finding* ("bump hunting") for high-dimensional binary data.
- This mirrors clustering, which looks for modes of a density in *continuous* space — here we look for high-probability *item combinations* instead.

Why Support and Confidence?

- **Why require minimum support?**
  - A rule with very low support may hold simply *by chance* — too few baskets to mean anything.
  - Rare combinations are also of little business value: there is little profit in promoting items customers seldom buy together.
  - So support filters out patterns that are statistically and commercially uninteresting.
- **Why require minimum confidence?**
  - Confidence measures the *reliability* of the inference $A \Rightarrow B$.
  - The higher $\mathrm{conf}(A \Rightarrow B) \approx P(B \mid A)$, the more strongly the presence of A predicts B.
- **A word of caution:** an association rule does *not* imply causation. It only signals a strong *co-occurrence* between antecedent and consequent.

The Association Rule Mining Task

- **Given:** a set of transactions T and two thresholds *minsup* and *minconf*.
- **Find:** all rules $A \Rightarrow B$ with

$$\mathrm{supp}(A \cup B) \geq \textit{minsup} \quad \text{and} \quad \mathrm{conf}(A \Rightarrow B) \geq \textit{minconf}.$$

- A brute-force approach — list every possible rule, compute both measures, discard those below threshold — is *computationally infeasible* (exponentially many rules).
- **Key observation:** all rules from the same itemset share the same support but differ in confidence. So we *decouple* the two requirements and mine in two stages:
  1. **Frequent itemset generation:** find all itemsets with $\mathrm{supp} \geq \textit{minsup}$.
  2. **Rule generation:** from each frequent itemset, form high-confidence rules.
- Stage 1 is the expensive part — and the focus of *Apriori*.

Why Finding Frequent Itemsets Is Hard

- To report rules, we first need all *frequent* itemsets — those with $\mathrm{supp} \geq$ a threshold *minsup*.
- With p items there are $2^p - 1$ candidate itemsets. A brute-force scan counting every candidate against every transaction is hopeless for realistic p.
- **Key idea (the Apriori principle):**

$$A \subseteq B \implies \mathrm{supp}(A) \geq \mathrm{supp}(B).$$

  Support can only *shrink* as an itemset grows.
- Contrapositive: **if an itemset is infrequent, every superset is infrequent too** — so we never need to count them.
- This *anti-monotone* (downward-closure) property is what makes the search feasible.

The Apriori Algorithm

A *level-wise* search: build frequent k-itemsets from frequent $(k-1)$-itemsets, pruning aggressively.

- Find all frequent 1-itemsets F_1 (one scan of the data).
- For $k = 2, 3, \ldots$ until F_{k-1} is empty:
  - **Generate** candidates C_k by joining pairs in F_{k-1}.
  - **Prune** any candidate that has an infrequent $(k-1)$-subset (by the Apriori principle it cannot be frequent).
  - **Count** support of the survivors in one data scan; keep those with $\mathrm{supp} \geq \mathit{minsup}$ as F_k.
- Output: $\bigcup_k F_k$, all frequent itemsets.
- Only *one pass over the data per level k* — crucial when the data do not fit in memory.

Apriori by Hand: One Pass ($minsup = 3$)

Six transactions over {Bread, Milk, Beer, Diaper, Coke, Eggs}:

TID	Items
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk
6	Bread, Diaper, Milk

1-itemsets (counts):

Bread 5, Milk 5, Diaper 5, Beer 3 ✓
Coke 2, Eggs 1 — pruned

2-itemsets (from frequent 1-sets):

{Bread,Milk} 4, {Bread,Diaper} 4,
{Milk,Diaper} 4, {Beer,Diaper} 3 ✓
{Bread,Beer} 2, {Milk,Beer} 2 — out

- No candidates involving Coke or Eggs are ever generated.
- Four frequent 2-itemsets survive. *Next:* build 3-itemset candidates from them and prune.

Apriori by Hand: 3-Itemsets and Pruning

Frequent 2-itemsets: {Bread,Milk}, {Bread,Diaper}, {Milk,Diaper}, {Beer,Diaper}.

- **Generate** 3-itemset candidates (union of two frequent 2-itemsets), then **prune** any whose 2-subsets are not all frequent — *without* touching the data:

Candidate	**Offending 2-subset**	**Action**
{Bread,Diaper,Milk}	all subsets frequent	count it
{Beer,Bread,Diaper}	{Beer,Bread} (supp 2)	prune
{Beer,Diaper,Milk}	{Beer,Milk} (supp 2)	prune

- Only one candidate, {Bread,Diaper,Milk}, needs a data scan — its count is **3** $\geq$ *minsup*, so it *is* frequent.
- It is the sole frequent 3-itemset. Generating 4-itemset candidates from a single 3-itemset is impossible, so the algorithm **stops**. Pruning saved us from ever counting the other two.

Stage 2: Rule Generation

- Apriori stops once it has the frequent itemsets — it only ever used *support*. **Confidence enters now.**
- For each frequent itemset K and every nonempty $A \subset K$, form the candidate rule $A \Rightarrow K \setminus A$ and keep it iff $\mathrm{conf} \geq \mathit{minconf}$.
- An itemset of size k yields $2^k - 2$ candidate rules (excluding $\varnothing \Rightarrow K$ and $K \Rightarrow \varnothing$).

Example. Frequent itemset {Beer, Diaper}, count 3. Two candidate rules ($2^2 - 2 = 2$):

Rule	supp	conf	
{Beer} $\Rightarrow$ {Diaper}	1/2	$3/3 = 1.00$	✓
{Diaper} $\Rightarrow$ {Beer}	1/2	$3/5 = 0.60$	(out if $\mathit{minconf} = 0.8$)

Same support, different confidence — confidence is *directional*, support is not.

Pruning the Rule Lattice

- Within a fixed itemset, confidence is **anti-monotone in the size of the consequent**: moving an item from the consequent to the antecedent can only *raise* confidence.
- So if a rule fails *minconf* , every rule with a *larger* consequent fails too — prune them without computing.

Illustration (the frequent itemset {Bread, Diaper, Milk}, count 3; listing its rules from the smallest consequent to the largest:)

Rule (consequent size)		conf
{Bread,Diaper} $\Rightarrow$ {Milk}	(1)	$3/4 = 0.75$
{Diaper,Milk} $\Rightarrow$ {Bread}	(1)	$3/4 = 0.75$
{Bread} $\Rightarrow$ {Diaper,Milk}	(2)	$3/5 = 0.60$
{Diaper} $\Rightarrow$ {Bread,Milk}	(2)	$3/5 = 0.60$

Confidence drops as the consequent grows ($0.75 \rightarrow 0.60$) — the same lattice pruning idea as for itemsets, now applied to rules.

Unsupervised Learning: A Unifying View

The whole toolkit, by the **data type** it is built for:

Method	**Data type**	**Structure it reveals**
PCA	continuous	directions of largest variance
K-means / hier.	continuous	modes (high-density groups)
Matrix completion	continuous, sparse	low-rank latent factors
AE / VAE	continuous	nonlinear latent representation
Association rules	**binary**	**high-probability item combinations**

- Different methods, different data — but one common goal: *describe the latent structure of unlabeled data.*
- Association rules add the *binary* corner: support, confidence, and lift are all summaries of $P(X)$, and *Apriori* makes the 2^p search tractable via downward closure.