Lecture 3: The Apriori Algorithm

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
August–December 2020

Association rules

- Given sets of items I and transactions T, with confidence χ and support σ, find all valid association rules $X \rightarrow Y$
  - Transactions containing X typically also contain Y
  - $X, Y \subseteq I$, $X \cap Y = \emptyset$
  - $\dfrac{(X \cup Y).count}{X.count} \geq \chi$
  - $\dfrac{(X \cup Y).count}{M} \geq \sigma$

Some points to note

- A form of data mining
- An algorithmic problem with an exact solution
  - Not the usual situation in machine learning

Frequent itemsets

- Find all $Z \subseteq I$ such that $Z.\text{count}/M \geq \sigma$
  - $Z.\text{count} \geq \sigma \cdot M$
- Naïve strategy: maintain a counter for each Z
  - One counter per subset of I, not enough memory
- With
  - 1 million items ($|I| = 10^6$),
  - 1 billion transactions ($|T| = 10^9$),
  - 10 items per transcaction ($|t_j| \leq 10$),
  - Support 1% ($\sigma = 0.01$),

  at most 1000 items can be frequent
- How can we exploit this?

Apriori

- Clearly, if Z is frequent, so is every subset $Y \subseteq Z$
- We exploit the contrapositive

Apriori observation

If Z is not a frequent itemset, no superset $Y \supseteq Z$ can be frequent

- For instance, in our earlier example, every frequent itemset must be built from the 1000 frequent items
- In particular, for any frequent pair $\{x, y\}$, both $\{x\}$ and $\{y\}$ must be frequent
- Build frequent itemsets bottom up, size 1,2,...

Apriori algorithm

- F_i : frequent itemsets of size i — Level i
- F_1: Scan T, maintain a counter for each $x \in I$
- $C_2 = \{\{x, y\} \mid x, y \in F_1\}$: Candidates in level 2
- F_2: Scan T, maintain a counter for each $X \in C_2$
- $C_3 = \{\{x, y, z\} \mid \{x, y\}, \{x, z\}, \{y, z\} \in F_2\}$
- F_3: Scan T, maintain a counter for each $X \in C_3$
- ...
- C_k = subsets of size k, every $(k-1)$-subset is in F_{k-1}
- F_k: Scan T, maintain a counter for each $X \in C_k$
- ...

Apriori algorithm

- C_k = subsets of size k, every $(k-1)$-subset is in F_{k-1}
- How do we generate C_k?
- Naïve: enumerate subsets of size k and check each one
  - Expensive!
- Observation: Any $C'_k \supseteq C_k$ will do as a candidate set
- Items are ordered: $i_1 < i_2 < \cdots < i_N$
- List each itemset in ascending order
- Merge two $(k-1)$-subsets if they differ in last element
  - $X = \{i_1, i_2, \ldots, i_{k-2}, i_{k-1}\}$
  - $X' = \{i_1, i_2, \ldots, i_{k-2}, i'_{k-1}\}$
  - $\text{Merge}(X, X') = \{i_1, i_2, \ldots, i_{k-2}, i_{k-1}, i'_{k-1}\}$

Apriori algorithm

- Merge$(X, X') = \{i_1, i_2, \ldots, i_{k-2}, i_{k-1}, i'_{k-1}\}$
  - $X = \{i_1, i_2, \ldots, i_{k-2}, i_{k-1}\}$
  - $X' = \{i_1, i_2, \ldots, i_{k-2}, i'_{k-1}\}$
- $C'_k = \{\text{Merge}(X, X') \mid X, X' \in F_{k-1}\}$
- Claim $C_k \subseteq C'_k$
  - Suppose $Y = \{i_1, i_2, \ldots, i_{k-1}, i_k\} \in C_k$
  - $X = \{i_1, i_2, \ldots, i_{k-2}, i_{k-1}\} \in F_{k-1}$ and $X' = \{i_1, i_2, \ldots, i_{k-2}, i_k\} \in F_{k-1}$
  - $Y = \text{Merge}(X, X') \in C'_k$
- Can generate C'_k efficiently
  - Arrange F_{k-1} in dictionary order
  - Split into blocks that differ on last element
  - Merge all pairs within each block

Apriori algorithm

- $C_1 = \{\{x\} \mid x \in I\}$
- $F_1 = \{Z \mid Z \in C_1, Z.\text{count} \geq \sigma \cdot M\}$
- For $k \in \{2, 3, \ldots\}$
  - $C_k = \{\text{Merge}(X, X') \mid X, X' \in F_{k-1}\}$
  - $F_k = \{Z \mid Z \in C_k, Z.\text{count} \geq \sigma \cdot M\}$
- When do we stop?
- k exceeds the size of the largest transaction
- F_k is empty

Next step: From frequent itemsets to association rules