COL761/COL7361/AIL7026
Data Mining
Assignment 1 Report

Task 1: Frequent Itemset Mining
Apriori vs FP-Growth Comparison

---

1 Introduction

Frequent itemset mining is a fundamental task in data mining that aims to discover patterns that frequently co-occur in transactional datasets. Two classical algorithms for this task are **Apriori** and **FP-Growth (FP-Tree)**.

Apriori follows a candidate generation and pruning strategy, while FP-Growth compresses the database into a tree structure and avoids explicit candidate generation. This task evaluates and compares their empirical runtime performance at different minimum support thresholds.

2 Objective

The objective of this experiment is to:

- Execute Apriori and FP-Growth on the same dataset.
- Measure runtime at support thresholds: 5%, 10%, 25%, 50%, and 90%.
- Plot both runtimes on a single graph.
- Analyze and compare their performance characteristics.

3 Runtime Comparison on Original Dataset

The runtime of both algorithms at different support thresholds is shown in figure below.

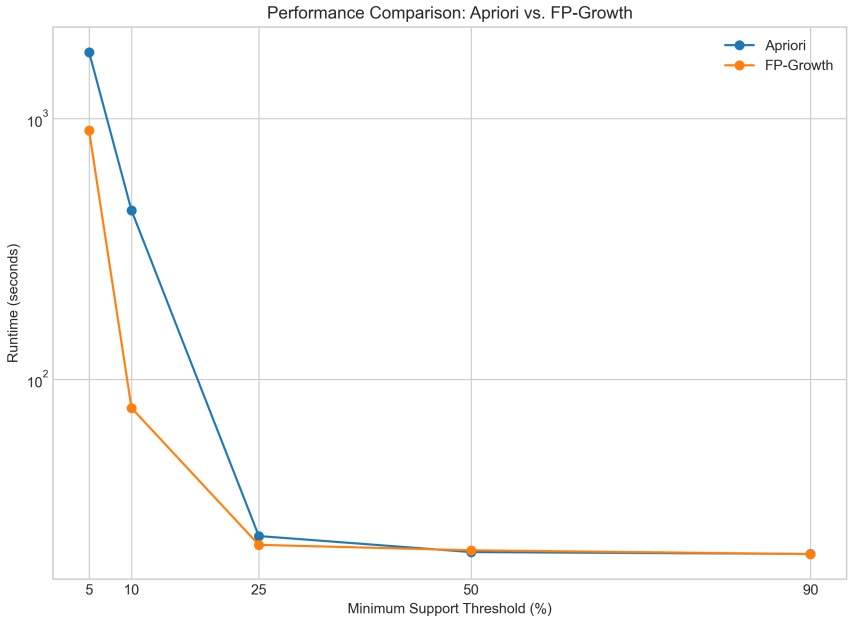

Figure 1: Runtime comparison of Apriori and FP-Growth for different minimum support thresholds on original dataset

3.1 Observations

From the plot, the following trends can be observed:

1. Apriori has significantly higher runtime at low support values (5% and 10%).
2. FP-Growth runs consistently faster than Apriori across all thresholds.
3. Runtime decreases rapidly as the support increases.
4. At high supports (50% and 90%), both algorithms have very small and similar run-times.
5. The difference between the two methods is most pronounced when the number of frequent item-sets is large.

3.2 Analysis

The differences arise due to the inherent design of the algorithms.

3.3 Apriori

- Generates candidate item-sets level-by-level.
- Requires multiple database scans.
- Performs expensive subset checks.
- Suffers from combinatorial explosion at low supports.

When the support threshold is small, many patterns are considered frequent. This leads to a large number of candidate item-sets, drastically increasing both memory usage and runtime.

3.4 FP-Growth

- Builds a compact FP-Tree representation of the database.
- Requires only two scans of the dataset.
- Avoid candidate generation.
- Uses recursive tree mining, which is computationally efficient.

Because of its compressed structure, FP-Growth handles dense datasets much more efficiently and scales better for low supports.

4 Runtime Comparison on Generated Dataset

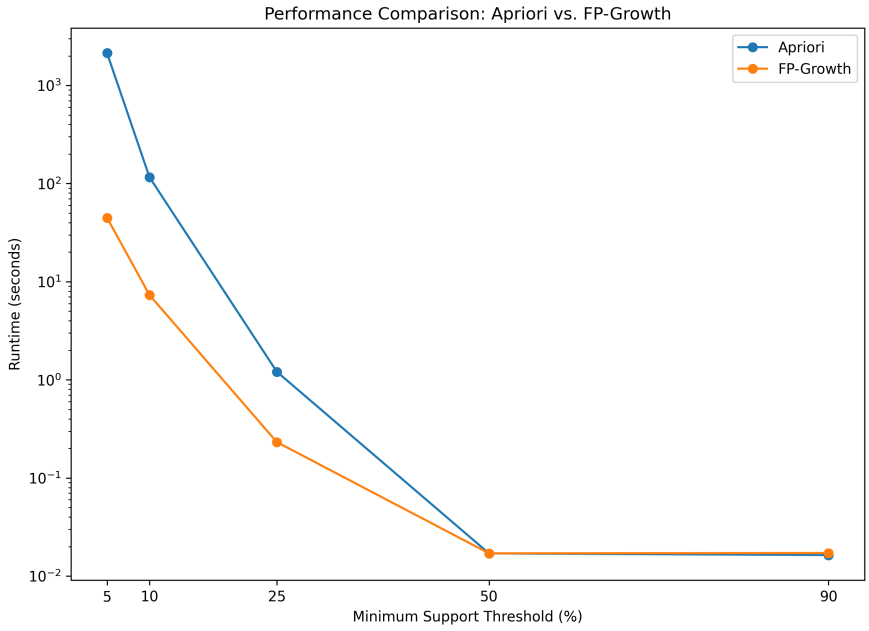

Figure 2: Runtime comparison of Apriori and FP-Growth for different minimum support thresholds on generated dataset

4.1 Dataset Generation

The dataset was constructed using a dense core-background model. A small subset of items (30 items) forms a dense core that appears in 60% of transactions. Each core transaction contains 20-28 of these items, creating high co-occurrence. The remaining transactions are generated sparsely from the full item universe to simulate noise. This structure ensures many frequent itemsets at low support thresholds, which increases the candidate search space for Apriori while allowing FP-Growth to efficiently compress the data using an FP-tree.

4.2 Observations

- At low support (5%), most combinations of the core items become frequent, causing an exponential increase in candidate generation for Apriori and resulting in very high runtime.
- However, FP-growth remains significantly faster due to avoiding candidate generation using an FP-tree structure.
- As the support threshold increases, the number of frequent itemsets decreases rapidly, causing Apriori's runtime to drop sharply.
- At high support (50-90%), both algorithms converge to nearly identical runtimes since very few frequent patterns exist.