Indonesian Journal of Data and Science

Volume 5 Issue 2
ISSN 2715-9936
https://doi.org/10.56705/ijodas.v5i2.134

*Research Article*

# Adaptive Minimum Support Threshold for Association Rule Mining

**Matthew Tunde Ogedengbe [1,*], Sahalu B. Junaidu [2], Donfack Kana [3]**

[1] Joseph Sarwuan Tarka University, Makurdi, Nigeria, matt.oged@uam.edu.ng
[2] Ahmadu Bello University, Zaria, Nigeria, abuyusra@gmail.com
[3] Ahmadu Bello University, Zaria, Nigeria, donfackkana@gmail.com

Correspondence should be addressed to Matthew Tunde Ogedengbe; matt.oged@uam.edu.ng

Received 10 May 2024; Accepted 12 June 2024; Published 31 July 2024



**Abstract:**

In association rule mining (ARM), valuable rules are extracted from frequent item sets, selecting appropriate minimum support thresholds is essential yet challenging. Arbitrary threshold selection often results in either an overwhelming number of uninteresting rules or the omission of relevant rules. To address this issue, this study introduces an Adaptive Minimum Support (Sad) algorithm designed to dynamically adjust the support threshold based on dataset characteristics, thereby facilitating the discovery of optimal association rules. The Sad algorithm was experimented on three real-world datasets, yielding optimal minimum support thresholds of 0.065, 0.133, and 0.057 respectively. Results demonstrate the algorithm's effectiveness in adapting the support threshold to each dataset's characteristics. By optimizing the threshold, the Sad algorithm enhances the quality of discovered association rules, offering more actionable insights for decision-making.

**Keywords:** Adaptive Support Threshold, Item Sets, Transactions, Item Support.
**Dataset link:** -

## 1. Introduction

In this era of rapid technological advancement, the accumulation of vast amounts of data has become a ubiquitous phenomenon. Extracting meaningful insights from this data poses a significant challenge. To manage this challenge, data mining techniques, also known as knowledge discovery in databases (KDD), are commonly employed [1]–[3] by researchers. Among these techniques is association rule mining (ARM), which stands out as a prominent method for uncovering associations among items within large datasets [4], [5].

The extraction of frequent item sets is the fundamental process in ARM, which are sets of items that frequently co-occur in the dataset [13] – [15]. This process typically involves algorithms such as the Apriorism algorithm, which generates candidate sets and filters them based on minimum support thresholds to identify frequent item sets [1]. However, the selection of an appropriate minimum support threshold remains a critical issue [4] –[6]

The classical support-confidence framework for ARM operates in two phases: first, frequent patterns are extracted, followed by the mining of association rules from these patterns. Traditional approaches often rely on arbitrary selection of minimum support threshold by the user. However, this approach poses challenges. Setting the threshold too low may result in an excessive number of meaningless items sets which is overwhelming in decision-making. Conversely, setting it too high risks missing valuable patterns [6] –[8].

To addressing this challenge, this study introduces an adaptive minimum support algorithm. Unlike traditional methods, this algorithm dynamically generates an optimal minimum support threshold based on dataset characteristics. By doing so, it aims to produce a set of relevant and interesting rules appropriate for effective decision-making [5], [9], [14].

## 2. Method:

### Data Collections

Three real-world datasets were employed: The Student Stress Factors Dataset, sourced from Kaggle with 554 instances detailing 6 stress factors; the Psychological Capital Dataset, also from Kaggle, containing psychological capital information of 329 employees with 20 items; and the LikertPlay Dataset, comprising 1350 respondents and 6 transaction items, used for examining Likert-scale responses. These datasets formed the basis for validating and evaluating the proposed algorithms.

### Research Tools Used

The research implementation required Python 3.0 programming language for algorithm development, alongside the utilization of the Google collaboration environment with Tensor Processing Unit (TPU) as a hardware accelerator. The hardware specifications included a Lenovo ThinkPad T470 running Windows 11 Pro, with an Intel® CoreTM i7-6600U CPU @ 2.60GHz, 8GB RAM, and 256 SSD.

### Association Rule Mining

Association rule mining is a fundamental data mining technique used to extract significant associations or correlations between item sets within large datasets. Each attribute of a record in the dataset D is considered an item [18], collectively forming item sets. Transactions T represent nonempty records, if X and Y are nonempty subsets with an empty intersection, the association rule X→Y is established within the item set T [10]-[13].

a. Support

This metric quantifies the frequency of occurrence of a particular item in the dataset and is calculated as the ratio of transactions containing the item to the total number of transactions. Higher support values indicate more prevalent item sets and are crucial for identifying significant associations. Equation 1 depicts mathematical representation of item support [11].

$$Sup(X) = \frac{frequency\ (X)}{Total\ \ transactions\ (D)} \tag{1}$$

b. Confidence

This metric measures the reliability of an association rule and is calculated as the ratio of the support of the combined itemset to the support of the antecedent item. Confidence values range from 0 to 1, with higher values indicating stronger associations [14] − [16]. Mathematically, the confidence of a rule $X \rightarrow Y$, denoted as Confidence $(X \rightarrow Y)$, is calculated as the ratio of the support of the item set $(X \cup Y)$ to the support of $X$ as depicted in Equation 2.

$$Confidence(X \rightarrow Y) = \frac{Sup(X \cup Y)}{Sup(X)} \tag{2}$$

c. Lift

This metric assesses the significance of an association rule beyond chance and is calculated as the ratio of the rule's confidence to the support of the consequent item. Mathematically, Equation 3 defines lift of a rule as the ratio of the confidence of the rule to the support of $Y$:

$$Lift(X \rightarrow Y) = \frac{Confidence(X \rightarrow Y)}{Sup(Y)} \tag{3}$$

A lift value greater than 1 indicates a meaningful association between items [10], [11].

### Apriorism Algorithm

The Apriorism algorithm is a fundamental technique for association rule mining, operating on the principle of generating frequent item sets by iteratively identifying and pruning infrequent ones. It consists of two main steps: joining, which combines frequent item sets to form larger ones, and pruning, which eliminates itemset below the minimum support threshold [14], [17]. Despite its effectiveness, Apriorism faces challenges such as the curse of dimensionality, multiple database scans, and scalability issues, impacting its performance in large datasets due to arbitrary selection of minimum support threshold by the users [10]−[12].

**Proposed Adaptive Minimum Support Threshold ($S_{ad}$) Algorithm**

To enhance the performance of the Apriori algorithm in generating meaningful sets of rules, an adaptive threshold was introduced to minimize database scans and improve candidate pruning. The proposed adaptive minimum support threshold was developed through the following steps:

**Data Characteristic Extraction**

Data characteristic extraction involves gaining insights into the dataset for mining purposes [5]. Algorithm 1 outlines the essential steps for data characteristic extraction in the adaptive minimum support threshold approach. The algorithm extracts data characteristics such as unique items, item support, maximum support, average number of items in transactions, and the expected number of rules.

The steps involved in the adaptive minimum support threshold development are as follows:

1) Identified the unique items (I): Distinct items present in the dataset were identified to understand the variety and frequency of items.
2) Determine the number of items in each transaction: The count of items in each transaction was analysed to identify patterns and trends, providing insights into transaction complexity.
3) Determine the average number of items in the overall transactions ($N_{avg}$): The average number of items was calculated to understand the typical transaction size. $N_{avg}$ was utilized for estimating rule size ($R$) and setting an Initial Minimum Support ($\omega$). Average number of items in the dataset is the ratio of total number of items in all transactions to the number of unique items ($I$) as derived in Equation 4;

$$N_{avg} = \frac{\sum N}{I} \tag{4}$$

4) Calculate item support ($Sup(x)$): The support of each unique item ($x$) was computed by counting its frequency in transactions and dividing by the total number of transactions ($T$) as denoted in Equation 5 as:

$$f(x) = Number\ of\ Transaction\ containing\ x \tag{5}$$

Mathematically, support of an item ($x$) is calculated in Equation 6 as;

$$Sup(x) = \frac{f(x)}{T} \tag{6}$$

5) Determine the maximum support value ⟦($Sup_{max}$): The highest support value among items was identified, serving as an upper bound for all item support in the dataset.
6) Determine the total number of possible rules from the dataset ($R$): The expected number of rules in a dataset was determined by the formular in Equation 7 as;

$$R = 2^I - 1 \tag{7}$$

Where;
- $R$: the expected number of possible rules in the dataset
- $I$: the number of unique items in the dataset

**Algorithm 1:** Data Extraction Algorithm

```
Input: Transaction dataset (T)
Output: Data characteristics
1: begin
  // Initialize variables
2: uniqueItems ← []
3: itemCounts ← []
4: transactionCount ← 0
```

```
5: avgItems ← 0
  // Step 1: Identify unique items
6: for transaction in T:
7:    for item in transaction:
8:       if item not in uniqueItems:
9:            uniqueItems.append(item)
  // Step 2: Determine the number of items in each transaction
10:for transaction in T:
11:    transactionCount ← transactionCount + 1
12:    itemCounts.append(len(transaction))
  // Step 3: Determine the average number of items in overall
transactions
13: avgItems ← sum(itemCounts) / transactionCount
  // Step 4: Calculate item support
14: itemSupport ← {}
15: for item in uniqueItems:
16:    transactionCountForItem ← 0
17:    for transaction in T:
18:        if item in transaction:
19:            transactionCountForItem ← transactionCountForItem+1
20:    itemSupport[item] ←transactionCountForItem/transactionCount
   // Step 5: Determine the maximum support value (Sup_max)
21: maxSupport ← max(itemSupport.values())
   // Step 6: Determine the total number of possible rules
22: numRules ← 2^len(uniqueItems) - 1
   // Output data characteristics
23: return uniqueItems, itemSupport, maxSupport, avgItems, numRules
24: end
```

**Initial Minimum Support Threshold ($\omega$) Based on Data Characteristics:**

To compute the initial estimate value for the minimum support threshold ($\omega$), the following parameters were considered; maximum support value ($Sup_{max}$), number of possible rules ($R$), and average number of items ($N_{avg}$) in all transactions. These parameters were first determined and then the initial minimum support threshold derived and presented in Equation 8.

$$\omega = Sup_{max}\left(1 - \left(\frac{1}{\sqrt{R}}\right)^{\frac{1}{N_{avg}}}\right) \tag{8}$$

Where,

- $\omega$: is the initial minimum support threshold and it was derived based on the intuition that the Initial Minimum Support should increase as the number of rules ($R$) decreases, and as the number of items ($N$) increases.

- $Sup_{max}$: This represents the maximum support value in the dataset as presented in Equation 6, acting as an upper bound for the optimal support threshold.
- $1 - \frac{1}{\sqrt{R}}$: As the dataset grows larger, the decrement in ω becomes less apparent. This is because the term $\frac{1}{\sqrt{R}}$ approaches 0 as $R$ increases, causing the overall factor $(1 - \frac{1}{\sqrt{R}})$ to approach 1. Consequently, the impact of dataset size on the initial minimum support threshold becomes less significant as the dataset expands.
- $\frac{1}{N_{avg}}$: This factor ensures that the $\omega$ decreases as the number of items in a rule decrease. The rationale is that rules with fewer items are easier to interpret and should be considered even if they have less support. By raising the factor to the power of $\frac{1}{N_{avg}}$, $\omega$ decreases more rapidly as the number of items in a rule decrease

**Objective Function Based on the Interestingness Measures**

In the proposed Adaptive Minimum Support Threshold (Sad) Algorithm, the objective function formulation integrates support, lift, and confidence measures. Here's how it was achieved:

1) Compute Average Values: Average values for support, confidence, and lift were calculated to gauge their significance in the analysis., Equations 9 - 11 compute average value for support, confidence and lift respectively;

$$Sup_{avg} = \frac{\sum Sup_i}{Number\ of\ item\ sup} \tag{9}$$

$$Conf_{avg} = \frac{\sum Conf_i}{Number\ of\ Conf} \tag{10}$$

$$Lift_{avg} = \frac{\sum lift_i}{Number\ of\ lift} \tag{11}$$

2) Ranking: Measures were ranked based on their average values, with the highest-ranked measure assigned the highest weight. Therefore, the value of $\alpha$, $\beta$ and $\gamma$ were used as rank-based weights, which were assigned to support, confidence, and lift, respectively.
3) Compute Rank-Based Weights: Each measure was assigned a weight based on its rank, using Equation 12 to derive weight factors.

$$W_i = \frac{n - i + 1}{\sum_{j=1}^{n}(n - j + 1)} \tag{12}$$

Where:

- $W_i = \alpha$ is the weight assigned to support at rank *i*
- $W_i = \beta$ is the weight assigned to confidence at rank *i*
- $W_i = \gamma$ is the weight assigned to lift at rank *i*
- *i* is the rank of the measure.
- *j* represents the loop variable, iterating over each measure
- *n* is the total number of measures.

4) Combine Measures: The objective function was formulated using a linear combination of support, confidence, and lift, with the weights derived in the previous step based on the relationships between measures and their importance. The objective function was derived in Equation 13 as:

$$F(\alpha, \beta, \gamma) = \alpha * sup + \beta * Conf + \gamma * Lift \tag{13}$$

**Tree-Structured Parzen Estimator (TPE) as Hyperparameter Tuning Tool**

TPE was utilized to iteratively search for the optimal minimum support threshold within a defined range, using the objective function for optimization. Algorithm 2 provides a comprehensive framework for the adaptive minimum support threshold process, encompassing data characteristic extraction, initial minimum support threshold computation, objective function formulation, and hyperparameter tuning. It iteratively explores the search space, updating the best threshold and corresponding objective function value until convergence. Finally, the algorithm

identifies and returns the optimal threshold maximizing the objective function, ensuring efficient association rule mining.

**Algorithm 2:** Adaptive minimum support threshold

**Input:** Dataset, α, β, γ, sup, conf, lift

**Output:** S$Sup_{avg}$, $Sup_{max}$, R, N, ω, best_threshold

**//Data Characteristic Extraction**

1: Extract $Sup_{avg}$, $Sup_{max}$, R, N from the dataset

**//Compute Initial Minimum Support Threshold (ω)**

2: ω ← $Sup_{max}$*(1-(1/sqrt(R))^(1/$N_{avg}$))

**//Objective Function Formulation to compute F**

3: F ← α*sup + β*conf + γ*lift

**//Hyperparameter Tuning**

4: Initialize best_threshold and best_F

5: best_threshold ← ω

6: best_F ← objective_function(best_threshold)

**//Iterate through the search space**

7: For threshold in (ω, $Sup_{max}$):

**//Compute the objective function F within the search space**

8: F ← objective_function(threshold)

**//Update best_threshold and best_F**

9: if F > best_F:

10: best_threshold ← threshold

11: best_F ← F

12: Return best_threshold

**//Optimal Threshold Selection**

**//Compute data characteristics**

13: $Sup_{avg}$, $Sup_{max}$, R, N ← Data_Characteristics(Dataset)

**//Calculate initial_min_sup (ω)**

14: ω ← Initial_Min_Threshold($Sup_{max}$, R, N)

**//Assign weights to measures**

15: α, β, γ ← Assign_Weights_To_Measures()

**//Compute interestingness measures**

16: sup, conf, lift ← Compute_Measures(Dataset)

**//Compute optimal_threshold**

17: optimal_threshold ← Hyperparameter_Tuning(ω, $Sup_{max}$, Objective_Function(α, β, γ, sup, conf, lift))

```
    //Return optimal_threshold
18: return optimal_threshold
```

## 3. Results and Discussion

Table 1 presents a comprehensive characterization of each dataset, outlining key metrics such as the number of transactions, unique items, minimum support, maximum support, average support, and the computed initial and optimal minimum support thresholds for association rule mining. In the Student Stress Factor dataset, the Sad algorithm initially set a minimum support threshold of 0.020, which was subsequently optimized to 0.065. This adaptive adjustment illustrates the algorithm's ability to tailor support thresholds for improved pattern detection.

**Table 1.** Experimental results of the Adaptive minimum support ($S_{ad}$) algorithm

| S/N. | Datasets | Transactions | Unique items | Min. Support | Max. Support | Ave. Support | Initial Min. Support | Opt. Min. support |
|---|---|---|---|---|---|---|---|---|
| 1. | Student Stress Factor | 559 | 30 | 0.057 | 0.547 | 0.180 | 0.020 | 0.065 |
| 2. | Psychological Capital | 329 | 100 | 0.012 | 0.567 | 0.200 | 0.210 | 0.133 |
| 3. | LikertPlay | 1350 | 80 | 0.0276 | 0.370 | 0.198 | 0.200 | 0.057 |

Similarly, in the Psychological Capital dataset, the initial minimum support threshold was determined to be 0.210, later optimized to 0.133. Here, the algorithm's adaptability is highlighted by its ability to lower the support threshold, ensuring a balance between pattern capture and rule significance. Likewise, in the LikertPlay dataset, the Sad algorithm exhibited adaptiveness by adjusting the initial minimum support threshold of 0.200 to an optimized value of 0.057. This significant reduction underscores the algorithm's capability to flexibly adapt to dataset nuances.

## 4. Conclusion

In this study, research steps have been outlined on how to extract data characteristics in transaction dataset to determine an optimal minimum support threshold for association rule mining. However, the overall results underscore the efficacy of the adaptive minimum support threshold algorithm in fine-tuning support thresholds according to dataset characteristics. By optimizing these thresholds, the algorithm facilitates the discovery of more meaningful association rules, thereby enriching the analytical process with actionable insights.

## References:

[1] R. Agarwal, & M. Mittal, “Inventory classification using multi-level association rule mining”. Int. J. of Dec. Supt. Syst. Tech., vol. 11, no. 2, pp1-12. 2019. DOI: 10.4018/IJDSST.2019040101

[2] C.S.K Selvi, & A. Tamilarasi, “Association Rule Mining with Dynamic Adaptive Support Thresholds for Associative Classification”, International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007). 2007 doi:10.1109 /iccima.2007.233

[3] Q.H. Duong, B. Liao, P. Fournier-Viger & T.L. Dam, “An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies”. Knowledge-Based Sys, vol. 104, pp. 106-122, 2016. https://doi.org/10.1016/j.knosys.2016.04.016

[4] A. Dahbi, S. Jabri, Y. Balouki & T. Gadi, “Finding Suitable Threshold for Support in Apriori Algorithm Using Statistical Measures”. In Enabl. Mach. Learng. Appl. in Data Sci: Proceedings of Arab Conference for Emerging Technologies pp. 89-101, 2021 Singapore: Springer Singapore. https://doi.org/10.1007/978-981-33-6129-4_7

[5] Z. Kohzadi, A.M. Nickfarjam, L.S. Arani, Z. Kohzadi & M. Mahdian, “Extraction frequent patterns in trauma dataset based on automatic generation of minimum support and feature weighting”, BMC med. Res. Meth., vol. 24 no. 1, pp.40, 2024. https://doi.org/10.1186/s12874-024-02154-0

[6] E. Hikmawati, N.U. Maulidevi & K. Surendro, "Pruning Strategy on Adaptive Rule Model by Sorting Utility Items". IEEE Access, vol. 10, pp. 91650-91662, 2022. DOI: 10.1109/ACCESS.2022.3202307

[7] A. Dahbi, Y. Balouki, & T. Gadi, "Using multiple minimum support to auto-adjust the threshold of support in apriori algorithm" International Conference on Soft Computing and Pattern Recognition, pp. 111-119. 2018. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-76357-6_11

[8] B. Huynh, C. Trinh, V. Dang & B. Vo, "A parallel method for mining frequent patterns with multiple minimum support thresholds". Int. J. of Innov. Comp., Info. and Cont., vol. 15, no. 2, pp. 479-488, 2019. DOI: 10.24507/ijicic.15.02.479

[9] E. Hikmawati, N.U. Maulidevi, & K. Surendro, "Minimum Threshold Determination Method Based on Dataset Characteristics in Association Rule Mining". J. of Big Data, vol. 8, pp. 1-17, 2021. https://doi.org/10.1186/s40537-021-00538-3

[10] F. Liu, & G. Wu, "An improved Apriori algorithm based on compressed matrix". J. of Shan. Univ. (Eng. Edition), vol. 48, no. 6, pp. 82-88, 2018. DOI:10.1109/ISCID.2018.34

[11] A. Wu, & A. Liu, "Improvement of apriori algorithm based on Boolean matrix reduction". Comp. Eng. and sci., vol. 9., 2019. https://doi.org/10.1155/2022/3900094

[12] S. Rana, & M.N.I. Mondal, "A Seasonal and Multilevel Association Based Approach for Market Basket Analysis in Retail Supermarket", Euro. J. of Info. Tech. and Comp. Sci., vol. 1, no. 4, pp. 9-15, 2021. DOI:10.24018/compute.2021.1.4.31

[13] R.T. Agrawal, Imieli_nski, & A. Swami, "Mining Association Rules Between Sets of Items in Large Databases". ACM SIGMOD Record vol. 22, no. 2, pp.207, 1993. https://doi.org/10.1145/%20170036.1 70072.

[14] S. Darrab & B. Ergenç, "Vertical pattern mining for multiple support thresholds". Proc. Comp. Sci., vol. 112, pp. 417- 426, 2017. DOI:10.1016/j.procs.2017.08.051

[15] E. Hikmawati, & K. Surendro, "How to determine minimum support in association rule", In Proc. of the 2020 9th International Conference on Software and Computer Applications. Langkawi Malaysia: ACM, pp. 6-10, 2020. https://doi.org/10.1145/3384544.3 384563

[16] K.S. Sadhasivam, & T. Angamuthu, "Mining rare itemset with automated support thresholds". J. of Comp. Sci., vol. 7, no. 3, pp. 394, 2011. doi:10.3844/jcssp.2011.394.399

[17] A. Salam, & M.S.H Khayal, "Mining top−k frequent patterns without minimum support threshold". Knowl. and Info. Sys., vol. 30, no. 1, pp. 57–86. 2011. doi:10.1007/s10115-010-0363-3

[18] C.K. Selvi, & A. Tamilarasi, "An automated association rule mining technique with cumulative support thresholds". Int. J. Open Probl. in Compt. Math, vol 2, no 3, pp. 12, 2009.