TinyNARM: Simplifying Numerical Association Rule Mining for Running on Microcontrollers

Iztok Fister Jr.[1(✉)], Iztok Fister[1], Akemi Galvez[2,3], and Andres Iglesias[2,3]

[1] Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia
iztok.fister1@um.si
[2] University of Cantabria, Avenida de los Castros, s/n, 39005 Santander, Spain
[3] Toho University, 2-2-1 Miyama, 274-8510 Funabashi, Japan

Abstract. This paper presents a tinyNARM which is an experimental effort in approaching/tailoring the classical Numerical Association Rule Mining to limited hardware devices, precisely on ESP32 microcontrollers so that devices do not need to depend on in-cloud remote servers. The tinyNARM reduces the number of attributes in the transaction database by discretizing the continuous numeric attributes, and replacing the stochastic evolutionary algorithm for association rule mining with its deterministic counterpart. The preliminary results of the comparative study, in which the in-cloud NiaARM and the on-device tinyNARM were included by mining several UCI ML datasets, revealed that the quality of mined association rules and the time complexity were good enough for continuing the research in the direction of the tinyML.

Keywords: numerical association rule mining · NiaARM · tinyML

1 Introduction

The problem of global warming is, now more than ever, one of the most frequent topics in mainstream media. The same topic is also daily on the tables of many world politicians who are responsible for taking action to mitigate the effect of the warming Earth [18]. Although fuel is the primary pollutant in terms of emissions that have a significant impact on global warming Artificial Intelligence (AI) is nowadays also responsible for producing the carbon footprint [20], especially, when running complex AI models with a high computational cost [4,15,17].

In today's world, we are witnesses of the minimization of computer hardware on the one hand, and increasing their processing power on the other. Even today, smartphones can run complex computational models [7]. With the advent of the Tiny Machine Learning (tinyML) research area, researchers have been solving challenges, in how to run all phases of Machine Learning (ML) on the same miniaturized device, i.e., smartphones, Raspberry Pi computers, different microcontrollers, etc. The tinyML is defined broadly as a fast growing field of

P. García Bringas et al. (Eds.): SOCO 2023, LNNS 749, pp. 122–131, 2023.
https://doi.org/10.1007/978-3-031-42529-5_12

ML technologies and applications, including hardware (dedicated integrated circuits), algorithms, and software capable of performing on-device sensor (vision, audio, IMU, biomedical, etc.) data analytics at extremely low power, typically in the mW range and below, and hence enabling a variety of always-on use-cases and targeting battery operated devices [13].

The motivation of the current study lies primarily in the practical utilization of Numerical Association Rule Mining (NARM) in smart agriculture. In our previous study [8], we developed a hardware system based on an ESP32 microcontroller consisting of several sensors for capturing the data of plants. The whole system was intended for capturing data and transferring them to the web server. However, the data mining part, which involved NARM, was conducted on an in-cloud remote computer since the ESP32 microcontroller may not be able to run computationally expensive algorithms.

Having the possibility to conduct data mining also on the same microcontroller is very practical, since it typically does not need access to the internet, which is a bottleneck in smart agriculture in some remote areas. Secondly, having direct access to data mining results, which can lead to potential actions, may be more efficient.

To satisfy the previously-mentioned goals, this paper presents a novel approach for NARM that aligns with tinyML for running on limited hardware devices. The proposed deterministic algorithm is also computationally less expensive than the original variants based on the stochastic Evolutionary Algorithm (EA), and, thus, it may be dramatically closer to the green AI.

This paper defines a new algorithm called tinyNARM, and presents a detailed experimental comparison with the competitive approach to evaluate its efficiency. At the moment, we do not offer an implementation for the ESP32 microcontroller, but give a theoretical framework for implementing the tinyNARM.

In summary, the main contributions of this study are:

- a novel tinyNARM algorithm is developed for NARM,
- the tinyNARM is evaluated practically on several benchmark datasets,
- the benefit of using tinyNARM is outlined in smart agriculture.

The structure of the remainder of the paper is as follows: Sect. 2 illustrates basic information needed by a potential reader to understand the topics that follow. In Sect. 3, the design and implementation of tinyNARM is discussed in detail. The experiments and the results are the subjects of Sect. 4, while the paper is concluded with Sect. 5, where a short overview of the performed work is summarized and the potential directions are outlined for the future work.

2 Basic Information

2.1 Numerical Association Rule Mining

The ARM problem is defined formally as follows: Let us suppose a set of objects $O = \{o_1, \ldots, o_m\}$ and transaction database Db are given, where each object o_i

for $i = 1, \ldots, m$ denotes either categorical attributes $A_i^{(cat)} \in \{ca_{i,1}, \ldots, ca_{i,m_i}\}$ with a set of discrete values $\{ca_{i,k}\}$ for $k = 1, \ldots, m_i$ or numerical attributes $A_i^{(num)} \in \{cn_i\}$ with a feasible variable represented as an interval $[lb_i, ub_i]$, and each transaction T is a subset of objects $T \subseteq O$. Thus, the variable m designates the number of objects, m_i the number of attributes of the i-th object, $lb_i \in \mathcal{R}$ is the lower and $ub_i \in \mathcal{R}$ the upper bound of the numeric interval. Then, an association rule can be defined as an implication [1]:

$$X \Rightarrow Y, \tag{1}$$

where $X \subset O$, $Y \subset O$, in $X \cap Y = \emptyset$. The following two measures are defined for evaluating the quality of the association rule [1]:

$$supp(X \Rightarrow Y) = \frac{n(X \cup Y)}{N}, \tag{2}$$

and

$$conf(X \Rightarrow Y) = \frac{n(X \cup Y)}{n(X)}, \tag{3}$$

where $supp(X \Rightarrow Y) \geq S_{min}$ denotes the support and $conf(X \Rightarrow Y) \geq C_{min}$ the confidence of the association rule $X \Rightarrow Y$. There, N in Eq. (2) represents the number of transactions in the transaction database Db, and $n(.)$ is the number of repetitions of the particular rule $X \Rightarrow Y$ within Db. Additionally, C_{min} denotes minimum confidence and S_{min} minimum support, determining that only those association rules with confidence and support higher than C_{min} and S_{min} are taken into consideration, respectively.

Let us notice that each numerical attribute $A_i^{(num)}$ for $i = 1, \ldots, m$ is identified by an interval of feasible values limited by their lower and upper bounds. The broader the interval, the more association rules mined. The narrower the interval, the more specific relations are discovered between attributes. Introducing intervals of feasible values has almost two effects on the optimization: To change the existing discrete search space to continuous, and to adapt these continuous intervals to suit the problem of interest better.

2.2 Classical NARM Using Evolutionary Approaches

Typically, the NARM was solved using Swarm Intelligence (SI) [2] or Evolutionary Algorithms (EAs) [6] due to the complexity of the problem. Both types belong to a family of stochastic nature-inspired population-based algorithms and differ from each other according to the principle of exploring the problem search space. While the Darwian evolution operators (i.e., crossover and mutation) [3] are applied by EAs for variation of individuals, the variation operators in SI-based algorithms are usually guided by some natural phenomenon (i.e., the behavior of ants or bees living in colonies). Indeed, the NiaARM framework [16], used in our study, supports both types of NARM algorithms.

Each individual in the NiaARM algorithm is encoded as a real-valued vector of size $D + 1$:

$$\mathbf{x}_i = \{\underbrace{x_{i,1}, x_{i,2}, x_{i,3}}_{Attr_{i,1}^{(cat)}}, \underbrace{x_{i,4}, x_{i,5}, x_{i,6}, x_{i,7}}_{Attr_{i,2}^{(num)}}, \ldots, \underbrace{\ldots, x_{i,D}}_{Attr_{i,m}^{(.)}}, \underbrace{x_{i,D+1}}_{cp}\}, \tag{4}$$

where the categorical attributes are encoded by three and numerical ones by four sequential elements. For instance, the first three elements $\langle x_{i,1}, \ldots, x_{i,3}\rangle$ are decoded to the categorical attribute $Attr_{i,1}^{(cat)} = \langle \pi_{i,j}, A_{i,j}^{(cat)}, Th_{i,j}\rangle$, while the next four $\langle x_{i,4}, \ldots, x_{i,7}\rangle$ to the numerical attribute $Attr_{i,1}^{(num)} = \langle \pi_{i,j}, A_{i,j}^{(num)}, Th_{i,j}\rangle$. Thus, the variable $\pi_{i,j}$ denotes the order in the permutation of objects, $A_{i,j}^{(.)}$ the corresponding attribute value, and $Th_{i,j}$ is a threshold determining if the corresponding attribute is present in the rule. The last element in the vector encodes the so-called cut point cp calculated as $cp_i = \lfloor x_{i,D+1} \cdot (D-1)\rfloor + 1$ that determines which attributes belong to the antecedent and which to the consequent.

Obviously, the size of the vector $\mathbf{x}_i$ is calculated as:

$$D = \sum_{j=1}^{m} L\left(Attr_j^{(.)}\right), \tag{5}$$

where the function $L(.)$ identifies the length of either the categorical or numerical attribute.

After decoding, the association rule $X \Rightarrow Y$ is obtained from the real-valued vector, where the quality of the association rule is evaluated using a fitness function. The fitness function is calculated according to the following equation:

$$f(\mathbf{x}_i^{(t)}) = \frac{\alpha \cdot supp(X \Rightarrow Y) + \beta \cdot conf(X \Rightarrow Y)}{\alpha + \beta}, \tag{6}$$

where α, and β denote weights, $supp(X \Rightarrow Y)$ and $conf(X \Rightarrow Y)$ represent the support and confidence of the observed association rule, respectively.

2.3 TinyML

Modern AI-oriented applications in computer vision, natural language processing, and big data rely on Machine Learning (ML) methods that demand large-scale datasets to model training in the cloud environment [10]. Obviously, these environments support the so-called in-cloud learning that is connected with a great demand for computing resources. On the one hand, these systems ensure privacy and security of data, while they allow no personalization at all due to the model's inflexibility, suffer from a high latency, and run on expensive hardware [11].

Recently, we have witnessed the advent of very powerful computer devices (e.g., mobile devices, IoT, single-board computers) that change the view of the traditional computing [14]. Consequently, the increasing efficiency of hardware,

minimizing computational overhead and reducing energy costs have resulted in the emergence of the new computational paradigm, i.e., tinyML. The tinyML proposes the so-called on-device learning that prefers the use of ML applications and model training on the device itself.

The advantages of the on-device learning are summarized as follows [10]:

- customized model,
- resource adaptation,
- end-to-end learning,
- on-line applicability.

The customized ML model running on limited hardware demands a large-scale personalization. On the other hand, the personalized data are produced by a specific person, and therefore can be stored in reduced, small-sized datasets. Finally, the processing speed of the new hardware has also been increased dramatically recent years. As a result, it seems that a bright future is predicted for tinyML.

The criteria for successful implementation of the tinyML methods on-device are as follows [19]:

- memory footprint,
- processing speed,
- prediction accuracy.

The first criterion refers to the amount of data needed for on-device learning. The second one estimates the time in which solutions are obtained. Usually, the on-device learning operates in real-time conditions. The last criterion indicates the quality of solution that must be comparable to the results of the in-cloud learning.

3 TinyNARM

The purpose of the tinyNARM is to reduce the continuous intervals of numeric attributes into the transaction database by introducing their discretization. In place of the intervals, the most interesting numerical interval is determined according to a corresponding support measure. The interval bears the characteristics of the whole numeric attribute, and it is calculated in the beginning.

In the continuation, all the combinations C_r^M of n possible attributes by r selected ones are varied from $r = 2, \ldots, n$, where the implication sign is moved from the first to the last position in the combination of association rule in steps of one. Thus, the best association rule is searched for regarding the support. Furthermore, each of the observed combination of attributes and their corresponding rules are also saved in an archive of the mined rules.

A pseudo-code of the TinyNARM association rule mining algorithm is illustrated in Algorithm 1, from which it can be seen that the transaction database Db, consisting of categorical $A_i^{(cat)}$ and numerical attributes $A_i^{(num)}$, is entered

Algorithm 1. The tinyNARM for mining the reduced number of ARs.

Require: $T = \langle o_1, \ldots, o_m \rangle$ ▷ Load transaction database
Ensure: $arch\_c, best\_c^*$ ▷ An archive and the best combination
1: **for all** $A^{(num)} \in D$ **do**
2: Discretize $A_i^{(num)} = \{da_{i,j}\}$, where $d_{i,j} = \lfloor \frac{\max\_o_i - \min\_o_i}{N} \rfloor$
3: Calculate frequencies $supp(da_{i,j}) = \frac{Freq(da_{i,j})}{N}$ for $i = 1, \ldots, M_i$
4: Find the maximum element $A_i^{(dis)} = \max_{j=1,\ldots,M_i} supp(da_{i,j})$
5: $D' = (D - A_i^{(num)}) \cup A^{(disc)_i}$
6: **end for**
7: **for all** $r \in [2, M-1]$ **do**
8: **for all** $c \in C_r^M$ **do**
9: $arch\_c \cup c_{cp}$ for $cp \in [1, r-1]$
10: $best\_c = \max_{cp \in [1,r-1]} supp(c_{cp})$
11: **end for**
12: $best\_c^* = \max(best\_c^*, best\_c)$
13: **end for**

into the data mining process. Thus, all the numeric attributes $A_i^{(num)}$ are discretized as $A_i^{(dis)}$, while the original database:

$$Db = \{A_1^{(\cdot)}, \ldots, A_i^{(num)}, \ldots, A_j^{(cat)}, \ldots, A_M^{(\cdot)}\}$$

is transformed into a modified transaction database:

$$Db' = \{A_1^{(\cdot)}, \ldots, A_i^{(dis)}, \ldots, A_j^{(cat)}, \ldots, A_M^{(\cdot)}\},$$

where each numerical attribute $A_j^{(num)}$ is replaced with its representative $A_i^{(dis)}$ discrete attribute, which is calculated according to the maximum support $\max supp(da_{i,j}) = \frac{Freq(da_{i,j})}{M}$, where the function $Freq(da_{i,j})$ denotes the number of attributes' occurrences in the transaction database.

Let us expose that the mining algorithm is deterministic, and therefore its time complexity should be tractable [9].

4 Experiments and Results

The main goal of the experiments was to evaluate how close the results of the competitive NiaARM can approach the results of the proposed tinyNARM. The tinyNARM algorithm was coded in the Python programming language. Actually, the entire Python project is available on the following link[1]. As a part of the future work, we plan to run experiments on ESP32 microcontrollers, which will result in a new C language implementation of the tinyNARM based on the current Python prototype.

[1] https://gitlab.com/firefly-cpp/tinynarm.

4.1 Datasets

We have utilized four public datasets from the UCI ML library [5]. Table 1 presents the observed datasets with the number of corresponding features, their types, and number of instances. All datasets are intended primarily for classification tasks, where a class of each transaction in a dataset is counted as a standard feature. Three types of features were used in the experiments, i.e., pure categorical, pure numerical and mixed. In this case, the influence of the attributes' types can be evaluated on the results of the NARM.

Table 1. Datasets utilized in our study.

Dataset	Type	Features	Instances
Abalone	Categorical/Numerical	9	4,177
Breast_Cancer	Categorical	10	286
Nursery	Categorical	9	12,960
Wine	Numerical	14	178

The experiments were conducted as follows: Firstly, numerical datasets were discretized into five classes. Then, the discretized datasets were used for data mining in both methods. Thus, fair comparisons were ensured for both association rule mining algorithms. Finally, a detailed analysis of the obtained results was performed according to the various discretized datasets.

4.2 Experimental Environment

Similar to the NiaARM, the tinyNARM algorithm is also written entirely in Python. The experiments were conducted on a Fedora Linux PC with 8 GB RAM. The tinyNARM algorithm does not need any specific control parameters, and, as an input, requires only the name of a discretized dataset. On the other hand, the NiaARM was run using the Particle Swarm Optimization (PSO) as an ARM population-based metaheuristic algorithm [12]. Indeed, the algorithm's parameters were set as follows during the experiments: The population size was set to 50 individuals, while the termination condition to 10,000 fitness function evaluations. The other PSO parameters were assigned to the default values as suggested by the NiaARM framework.

4.3 Results

The results of the comparative analysis, in which the in-cloud NiaARM and on-device tinyNARM were incorporated, are collected in Table 2. Both methods discovered a lot of rules, but we took only the 20 best association rules into account and then applied descriptive statistics of these. Table 2 aggregates the

Table 2. Results of experiments

Algorithm	Measures	Abalone	Breast_Cancer	Nursery	Wine
tinyNARM	max	0.743	0.823	0.667	0.511
	min	0.558	0.554	0.222	0.503
	mean	0.623	0.649	0.274	0.505
	median	0.609	0.631	0.236	0.506
	std	0.059	0.081	0.101	0.003
	time	2.26	5.344	2.44	157.72
	no. rules	1,320	2,504	1,320	10,027
NiaARM	max	0.688	0.760	0.510	0.534
	min	0.549	0.579	0.503	0.511
	mean	0.605	0.638	0.505	0.519
	median	0.601	0.611	0.505	0.518
	std	0.041	0.057	0.002	0.007
	time	14.020	13.895	16.420	15.649
	no. rules	603	912	1,503	632

corresponding results according to the statistical metrics obtained by each ML algorithm mining the four UCI ML datasets. The results were compared according to: (1) the quality measured by the fitness (i.e., minimum, maximum, mean, standard deviation, and median), (2) the time complexity, and (3) the number of mined rules.

According to the quality measures, the results of both algorithms were comparable by mining all the four observed datasets, due to the means achieved by both not distinguishing by more than 1 %, except for the Nursery dataset, where the mean of the tinyNARM is for 48 % worse than by the NiaARM. On the other hand, the results according to the time complexity showed that the tinyNARM demanded approximately 10× more time for obtaining the comparable results. The third comparison showed that the tinyNARM mined on average more association rules.

4.4 Discussion

The experiments showed that the results of the tinyNARM are strongly dependent on the type of attributes in the transaction datasets. Indeed, this algorithm is dedicated for dealing with numerical attributes. Therefore, the results can be deteriorated when this algorithm is applied to the transaction datasets consisting of discrete attributes only.

On the other hand, the tinyNARM caused an increase in the time complexity when the mined transaction datasets consisted of numerical attributes. Obviously, the exhaustive search governing this algorithm is the main reason for this undesirable behavior. Finally, the number of association rules mined by the

observed algorithms is a consequence of the way of handling the archive, and can be changed easily.

5 Conclusion

The paper is a preliminary experimental study that presents how to move the in-cloud ML method to on-device tinyML, where the NARM was taken into the account as an ML method. Although the study showed that the tinyNARM performed well on average, some drawbacks of the approach can also be indicated: For instance, in order to minimize the time complexity, the stochastic population-based algorithm was replaced by the deterministic algorithm. Unfortunately, it seems that the proposed exhaustive search can be even more time complex in situations, where more numeric attributes exists in the transaction datasets. As a result, a new heuristic algorithm should be found in the future to avoid the problem.

Acknowledgements. Iztok Fister Jr. is grateful the Slovenian Research Agency for the financial support under Research Core Funding No. P2-0057. Iztok Fister thanks the Slovenian Research Agency for the financial support under Research Core Funding No. P2-0042 - Digital twin. This research is also supported by the PDE-GIR project from the European Union Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No 778035, and the project PID2021-127073OB-I00 of the MCIN/AEI/10.13039/501100011033/FEDER, EU.

References

1. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, VLDB, vol. 1215, pp. 487–499 (1994)
2. Blum, C., Merkle, D.: Swarm intelligence: Introduction and applications. In: Swarm Intelligence (2008)
3. Darwin, C., Carroll, J.: On the Origin of Species. Broadview Editions Series. Broadview Press (2003)
4. Dhar, P.: The carbon impact of artificial intelligence. Nat. Mach. Intell. **2**(8), 423–425 (2020)
5. Dua, D., Graff, C.: UCI machine learning repository (2017)
6. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series. Springer, Berlin Heidelberg (2015). https://doi.org/10.1007/978-3-662-44874-8
7. Fister, I., Deb, S.: Near real-time performance of population-based nature-inspired algorithms on cheaper and older smartphones. In: 2018 5th International Conference on Soft Computing & Machine Intelligence (ISCMI), pp. 12–16. IEEE (2018)
8. Fister Jr., I., Fister, D., Fister, I., Podgorelec, V., Salcedo-Sanz, S.: Time series numerical association rule mining variants in smart agriculture. arXiv:2212.03669 (2022)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)

10. Guo, S., Zhou, Q.: Machine Learning on Commodity Tiny Devices: Theory and Practice. CRC Press (2022)
11. Iodice, G.M.: TinyML Cookbook: Combine artificial intelligence and ultra-low-power embedded devices to make the world smarter. Packt Publishing (2022)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
13. Njor, E., Madsen, J., Fafoutis, X.: A primer for tinyml predictive maintenance: Input and model optimisation. In Maglogiannis, I., Iliadis, L., Macintyre, J., Cortez, P. (eds.) Artificial Intelligence Applications and Innovations, pp. 67–78. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-08337-2_6
14. Ren, H., Anicic, D., Runkler, T.A.: How to manage tiny machine learning at scale: An industrial perspective (2022)
15. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. arXiv:1906.02243 (2019)
16. Stupan, Ž, Fister, I., Jr.: Niaarm: a minimalistic framework for numerical association rule mining. J. Open Source Softw. **77**(7), 44–48 (2022)
17. Verdecchia, R., Sallou, J., Cruz, L.: A systematic review of green ai. arXiv:2301.11047 (2023)
18. Wallace-Wells, D.: The uninhabitable earth. In: The Best American Magazine Writing 2018, pp. 271–294. Columbia University Press (2019)
19. Warden, P., Situnayake, D.: TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media (2019)
20. Wiedmann, T., Minx, J.: A definition of 'carbon footprint'. Ecolog. Econ. Res. Trends **1**(2008), 1–11 (2008)