Learning Semantic Association Rules from Internet of Things Data

Journal Title
XX(X):1–16

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Erkan Karabulut[1], Paul Groth[1] and Victoria Degeler[1]

Abstract

Association Rule Mining (ARM) is the task of discovering commonalities in data in the form of logical implications. ARM is used in the Internet of Things (IoT) for different tasks including monitoring and decision-making. However, existing methods give limited consideration to IoT-specific requirements such as heterogeneity and volume. Furthermore, they do not utilize important static domain-specific description data about IoT systems, which is increasingly represented as knowledge graphs. In this paper, we propose a novel ARM pipeline for IoT data that utilizes both dynamic sensor data and static IoT system metadata. Furthermore, we propose an Autoencoder-based Neurosymbolic ARM method (Aerial) as part of the pipeline to address the high volume of IoT data and reduce the total number of rules that are resource-intensive to process. Aerial learns a neural representation of a given data and extracts association rules from this representation by exploiting the reconstruction (decoding) mechanism of an autoencoder. Extensive evaluations on 3 IoT datasets from 2 domains show that ARM on both static and dynamic IoT data results in more generically applicable rules while Aerial can learn a more concise set of high-quality association rules than the state-of-the-art with full coverage over the datasets.

Keywords

association rule mining, neurosymbolic AI, semantic web, autoencoder, internet of things, sensor data

Introduction

Association Rule Mining (ARM) is a common data mining task that aims to discover associations between features of a given dataset in the form of logical implications (Agrawal et al. 1994). In Internet of Things (IoT) systems, ARM methods are utilized for various tasks including monitoring, decision-making, and optimization, for example, of a system's resources (Sunhare et al. 2022). Some IoT application domains in which ARM has been successfully utilized include agriculture (Fan et al. 2021), smart buildings (Degeler et al. 2014) and energy (Dolores et al. 2023). However, most applications of ARM in IoT give limited considerations to characteristics of IoT data such as heterogeneity and volume (Ma et al. 2013) as they are mere adaptations of rule mining methods not specifically tailored to IoT requirements.

IoT systems can produce or use data from diverse sources which can be categorized as static and dynamic. Static data refers to data that is not subject to frequent changes such as system models while dynamic data is subject to frequent changes, for instance, sensor data. The static part of IoT systems is increasingly represented as knowledge graphs (Rhayem et al. 2020; Karabulut et al. 2024), large databases of structured semantic information (Hogan et al. 2021). ARM algorithms are often run on the dynamic part of IoT data, not utilizing the valuable information in knowledge graphs. In addition, ARM algorithms can generate a high number of rules as the input dimension increases (Kaushik et al. 2023; Telikani et al. 2020), which is time-consuming to process and maintain. Generating a high number of rules can be the case for large-scale IoT environments, as each sensor is treated as a different data dimension.

To address these two issues, this paper presents two new contributions. The first contribution is a novel ARM pipeline for IoT data that combines knowledge graphs and sensor data to learn association rules with semantic properties, *semantic association rules* (Section Problem Statement), that represent IoT data as a whole (Section Pipeline). We hypothesize that semantic association rules are more generically applicable than association rules based on sensor data only, requiring fewer rules to have full data coverage. As an example, an association rule based on sensor data only looks as follows: *'if sensor1 measures a value in range R, then sensor2 must measure a value in range R2'*. This rule can only be applied to *sensor1* and *sensor2*. In contrast, semantic association rules are more contextual as seen in the following example in the Water Distribution Network (WDN) domain: *'if a water flow sensor placed in a pipe P1 with diameter* $\geq$ *A1 measures a value in range R, then a water pressure sensor placed in a junction J1 connected to P1 measures a value in range R2'*. The semantic association rule is no longer about individual sensors. Instead, it describes a certain context that the sensor is placed in and therefore is more generically applicable and explainable.

[1]University of Amsterdam, The Netherlands

Corresponding author:
Erkan Karabulut, University of Amsterdam, Science Park 904, 1098 XH, The Netherlands
Email: e.karabulut@uva.nl

However, enriching sensor data with semantics from a knowledge graph increases input size and may result in a high number of rules. Hence, the second contribution of this paper is an Autoencoder-based (Vincent et al. 2008) Neurosymbolic ARM method (Aerial) as part of the proposed pipeline that can learn a concise set of high-quality rules with full data coverage (Section Rule Extraction from Autoencoders). Aerial learns a neural representation of a given input data and then extracts association rules from the neural representation. This approach can be supplemented by and is fully compatible with other ARM variations that aim to mine a smaller subset of high-quality rules such as top-k rules mining (Fournier-Viger et al. 2012), and ARM with item constraints (Baralis et al. 2012; Srikant et al. 1997). An extensive set of experiments (Section Evaluation) is performed and the results show that ARM on knowledge graphs and sensor data together results in more generically applicable rules with high support and data coverage in comparison to ARM on sensor data only (Section Discussion). Furthermore, the results show that the proposed Aerial approach is capable of learning a concise set of high-quality rules with full coverage over the entire data.

In summary, the two contributions of this paper are: (1) a pipeline of operations to learn contextual and more generically applicable semantic association rules from IoT data compared to existing methods; and (2) an Autoencoder-based ARM approach for learning a more concise set of high-quality semantic association rules than the state-of-the-art, with full data coverage. This approach is orthogonal and can be used with other ARM variations.

Related Work

This section introduces the related work and background concepts.

Association Rule Mining

ARM is the problem of learning commonalities in data in the form of logical implications, e.g., $X \rightarrow Y$, which is read as *'if X then Y'*. Initial ARM algorithms such as Apriori (Agrawal et al. 1994) and HMine (Pei et al. 2001) focused on mining rules from categorical datasets. The initial methods needed pre-discretization for numerical data, struggled with scaling on big high-dimensional data, and produced a high number of rules that are costly to post-process. FP-Growth (Han et al. 2000), a widely used ARM algorithm, has many variations to tackle some of the aforementioned issues. ARM with item constraints (Srikant et al. 1997) is an ARM variation that focuses on mining rules for the items of interest rather than all, which reduces the number of rules and execution time (Baralis et al. 2012). Guided FP-Growth (Shabtay et al. 2021) is an FP-Growth variation for ARM with item constraints. Other variations include Parallel FP-Growth (Li et al. 2008) and FP-Growth on GPU (Jiang and Meng 2017) for better execution times.

Recently, a few Deep Learning (DL)-based ARM algorithms have been proposed. Patel et al. (2022) proposed to use Autoencoders (Chen and Guo 2023) to learn frequent patterns in a grocery dataset, however, no source code or pseudo-code was given. Berteloot et al. (2023) also utilized Autoencoders (ARM-AE) to learn association rules directly from categorical tabular datasets. However, ARM-AE has fundamental issues while extracting association rules from an Autoencoder, which we elaborate on in Section Setting 2: Aerial vs state-of-the-art.

Numerical Association Rule Mining (NARM) aims to identify intervals for numerical variables to generate high-quality association rules based on specific quality criteria. Following the recent systematic literature reviews (Telikani et al. 2020; Kaushik et al. 2023), the state-of-the-art in NARM is nature-inspired optimization-based algorithms which include evolutionary, differential evolution, swarm intelligence, and physics-based approaches. They employ heuristic search processes to find association rules that optimize one or more rule quality criteria and are used for both numerical and categorical datasets (Fister et al. 2018). However, optimization-based ARM methods too suffer from handling big high-dimensional data, together with other broader issues in NARM such as having a large number of rules, and explainability as also mentioned by Kaushik et al. and other works (Telikani et al. 2020; Berteloot et al. 2023; Kishore et al. 2021).

Association Rule Mining in Internet of Things

In IoT, both exhaustive ARM, such as Apriori and FP-Growth, and the optimization-based NARM methods are used for various tasks. Shang et al. (2021) utilized the Apriori algorithm for big data mining in IoT in the enterprise finance domain for financial risk detection. Sarker and Kayes (2020) utilized an exhaustive ARM approach with item constraints on phone usage data to learn user behaviors. Khedr et al. (2020) proposed a distributed exhaustive ARM approach that can run on a wireless sensor network. Fister Jr et al. (2023) proposed TS-NARM, an optimization-based NARM approach, and evaluated it on a smart agriculture use case with 5 optimization-based methods.

Sequential or temporal ARM is another ARM variant used in IoT (Wedashwara et al. 2019). The goal is to learn patterns between subsequent events, rather than events that happen in the same time frame, concurrent events. In this paper, we focus on mining association rules for concurrent events, rather than sequential events which is a different task.

Based on recent surveys (Karabulut et al. 2024; Listl et al. 2024), semantic web technologies such as ontologies (Gruber 1993) and knowledge graphs (Hogan et al. 2021) have been used for knowledge representation in IoT, providing valuable knowledge related to IoT systems and its components. Naive SemRL (Karabulut et al. 2023) is the only ARM method that utilizes semantics when learning rules from pre-discretized sensor data. It is based on FP-Growth, however, the paper does not provide a complete evaluation. We adopt a similar semantic enrichment approach but develop a completely new DL-based pipeline, and provide an extensive evaluation.

Note that the term *semantic association rules* is also used when mining rules from knowledge graphs (Barati et al. 2017) only, which is a different task than rule learning from sensor data presented in this paper. To the best of our knowledge, there has been no fully DL-based ARM algorithm for learning association rules from concurrent events in IoT data.

Table 1. Input notation, explanations, and examples from water networks domain.

Notation	Explanation	Example
C	Classes in an Ontology/Data schema	Pipe, Junction
R, r	Relations (R) in between the classes (C) mapped with (r)	(Pipe)_connectedTo_(Junction)
A, a	Properties for the classes and relations	(Junction).elevation: elevation property of the class Junction
V	Node IDs in the knowledge graph	P1, J2
E, e	IDs of the edges (E) in between nodes (V) in the knowledge graph mapped with (e)	(P1)_(e1)_(J2), P1 and J2 are node IDs, e1 is an edge ID
L, l	Labels for the nodes (V) and edges (E) in the knowledge graph mapped with (l)	(P1:Pipe)_(e1:connected_to) _(J2:Junction)
P, U, p	Property (P) and value (U) pairs for nodes and edges mapped with (p)	(P1:Pipe).elevation=v1, the elevation of pipe P1 is v1
M, S, F, s	each timestamp (F) and sensor ID (S) pair is mapped to a value (M) with (s)	a water flow sensor with the ID s1, measures u1 at a time t1
V, S, b	each sensor (S) is mapped (b) to a node (V) in the knowledge graph	(S1:Sensor)_(:has_type)_(:WaterFlow), a water flow sensor

Our approach. In contrast to existing work, we utilize both static knowledge graphs and dynamic sensor data that represent IoT data as a whole and propose a novel neurosymbolic ARM approach for learning *semantic* rules from IoT data, for concurrent events. Our approach leads to a more concise set of high-quality association rules that are more generically applicable than sensor-only rules with full coverage over the data. In addition, semantic association rules facilitate domain knowledge integration as domain knowledge can also be represented as semantic rules, e.g., as part of a domain ontology underlying the knowledge graph.

Problem Definition

This research problem relates to learning association rules from sensor data in IoT systems with semantic properties from a knowledge graph describing the system and its components, properties, and the relations between them. We formulate the problems as follows:

Given a sensor dataset T with sensors mapped to nodes in knowledge graph G with binding B, produce a set of association rules with clauses based on T and G.

Association rules are formal logical formulas in the form of implications, e.g. $X \rightarrow Y$, where $X \rightarrow Y$ is a horn clause with $|Y| = 1$ referring to a single literal and $|X| \geq 1$ referring to a set of literals. X is referred to as the antecedent, and Y is the consequent. A horn clause is defined as *a disjunction of literals with at most one positive literal*. Note that $p \rightarrow q \wedge r$ can be re-written as $p \rightarrow q$ and $p \rightarrow r$, hence $|Y| = 1$.

Note that the T is converted to a set of transactions before the learning process, e.g., by grouping sensor data based on time frames. G is in the form of a directed property graph which contains semantic information of the items in T, e.g., where a sensor is placed, and binding B maps sensors in T to a corresponding node in G, assuming that each sensor has a representation in G. Output rules can express conditions on the sensor measurements and its context.

Input

This section presents input notation. To help readers understand easier, Table 1 lists symbols used in the notation, high-level explanations, and examples from WDN domain.

Knowledge graph. The knowledge graph described in this section is a property graph with an ontology or data schema as the underlying structure (Tamašauskaitė and Groth 2023). We adapt the definition for a *property graph*, given in the next paragraph, from (Hogan et al. 2021).

Property Graph. Let Con be a countably infinite set of constants. A property graph is a tuple $G = (V, E, L, P, U, e, l, p)$, where $V \subseteq Con$ is a set of node IDs, $E \subseteq Con$ is a set of edge IDs, $L \subseteq Con$ is a set of labels, $P \subseteq Con$ is a set of properties, $U \subseteq Con$ is a set of values, $e : E \rightarrow V \times V$ maps an edge ID to a pair of node IDs, $l : V \cup E \rightarrow 2^L$ maps a node or edge ID to a set of labels, and $p : V \cup E \rightarrow 2^{P \times U}$ maps a node or edge ID to a set of property–value pairs.

Ontology/Data Schema. Let $O = (C, R, A, r, a)$ be an ontology or data schema, where $C \subseteq Con$ is a set of classes, $R \subseteq Con$ is a set of relations, $A \subseteq Con$ be a set of properties, $r : R \rightarrow C \times C$ maps a relation to a pair of classes, and $a : C \cup R \rightarrow 2^P$ maps a class or a relation to a set of properties.

To express that G has O as its underlying structure, we define; i) $L \subseteq C \cup R$, meaning that the labels in G can only be one of the classes or relations defined in O, ii) $P \subseteq A$, meaning that the properties of V and E in G, can only be one of the properties in A.

Sensor data. We define sensor data generically as a tuple $T = (M, S, F, s)$, where $M \subseteq (\mathbb{R} \cup Con)$ is either real numbers representing numerical sensor measurements or constants representing categorical sensor values (states, e.g., a valve is open or closed), $S \subseteq Con$ is a set of sensor IDs, F is an ordered numerical sequence of timestamps and $s : (S, F) \rightarrow M$ maps every sensor ID and timestamp to a value. **Note**, further in this approach, the order of timestamps is considered only to aggregate sensor measurements into transactions (of time frames) to enable generalizable rule learning, since the task is not to learn temporal rules.

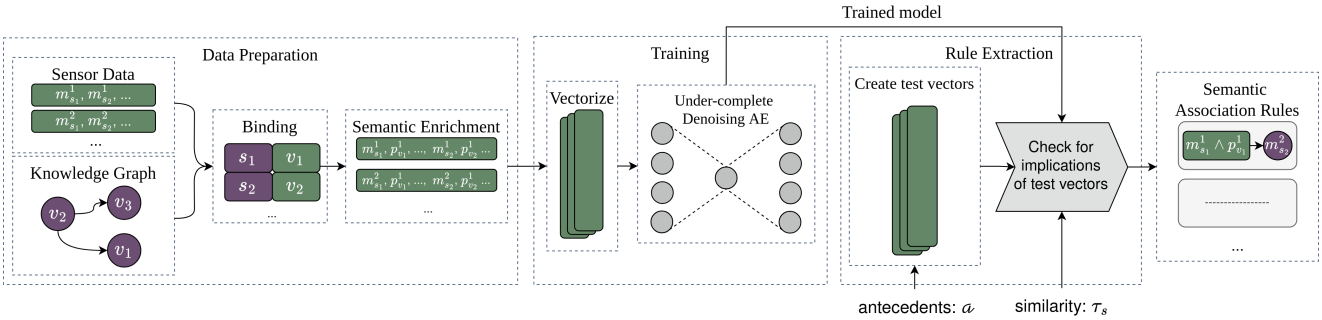

Figure 1. Proposed ARM pipeline for IoT data to learn semantic association rules from sensor data and knowledge graphs.

Table 2. Output item forms, explanations, and examples from the water network domain (#: Equation 1, and the symbols are described in Table 1).

Form	Example	Explanation
$p'\#z'$	p1.length > 100	Node p1 has length bigger than 100
$m'\#z'$	(s1:Sensor).value < 10	Sensor s1 measures a value smaller than 10
$v'_l = l'$	$p1 : Pipe$	Node p1 has the label 'Pipe'
$e'_l = l'$	$e1 : Junction$	Edge e1 has the label 'Junction'
$v' \rightarrow v'' = e'$	$p1 \rightarrow p2 = e1$	Node p1 connects to p2 via edge e1

Binding. It is a tuple $B = (V, S, b)$, where V is the set of node IDs from G, and S is the set of sensors IDs from T, $b : S \rightarrow V$ maps each sensor ID to a node in G, and $b(S) \subseteq V$ meaning that there is a node ID for each sensor ID, and there can be node IDs for more e.g., instances of classes in C.

Output

The output is a set of rules of the form described below.

Let I be a set of items, with the following basic comparison operations defined for each item: $\forall i' \in I\,(i' \in \{p'\#z', m'\#z', v'_l = l', e'_l = l', v' \rightarrow v'' = e'\})$, with $p' \in P$, $m' \in M$, $v', v'' \in V$, $e' \in E$, $l', v'_l, e'_l \in L$ where v'_l refers to a label mapped to a node with the ID v', and e'_l refers to a label mapped to an edge with the ID e'. z' refers to a value that is either *categorical* or *numerical*, $\#$ refers to one of the comparison operations with a truth value defined below:

$$\begin{aligned} \#_{\text{categorical}}(p, g) &::= (p = g) \mid (p \neq g) \mid (p \in \{g\}) \mid (p \notin \{g\}) \\ \#_{\text{numerical}}(p, g) &::= (p = g) \mid (p \neq g) \mid (p > g) \mid (p < g) \\ &\quad \mid (p \leq g) \mid (p \geq g) \end{aligned} \tag{1}$$

$X \rightarrow Y$ is an association rule where $(X, Y \subseteq I) \wedge (|Y| = 1)$. This means that items of the rule can only consist of properties of classes or relations defined in the ontology, and the consequent can only have one item. Examples and explanations for item forms are given in Table 2. The item forms consist of comparisons over $m \in M$ or $p \in P$, labels $l \in L$, and whether an edge $e \in E$ exists for a pair of $v \in V$. We call rules in this form **semantic association rules**.

Semantic Association Rules from IoT Data

This section introduces our proposed ARM pipeline for IoT data and an Autoencoder-based Neurosymbolic ARM approach (Aerial) as part of the pipeline. The goal is to learn a concise set of high-quality semantic association rules from sensor data and knowledge graphs with full data coverage.

Pipeline

Figure 1 depicts the proposed ARM pipeline for IoT data, which consists of three main stages: (i) data preparation, (ii) training, and (iii) rule extraction.

Data Preparation. This stage of the pipeline pre-processes a given sensor dataset and a knowledge graph describing the IoT system and its components to form transactions, making the data ready for ARM. First, sensor data is aggregated into time frames (e.g., average measurements per minute for all sensors). Each row in the Sensor Data depiction in Figure 1 refers to a time frame that contains aggregated sensor measurements, representing the state of the IoT system within that time frame. Second, the numerical sensor measurements and the numerical properties in the knowledge graph are discretized (e.g., by applying equal-frequency discretization (Foorthuis 2020)). This step is optional when using a Numerical ARM method as part of the pipeline. Note that the method of sensor data aggregation and discretization is domain-dependent, and our approach is independent of the preferred aggregation and discretization method.

As the last step, binding B is utilized to enrich sensor data with semantics from the knowledge graph. Given a set of transactions derived from sensor data T, and the binding B, the semantic enrichment process augments each transaction with semantic properties from the knowledge graph G. For each sensor $s_i \in T$, we retrieve its corresponding node $v_i = b(s_i)$ in G. We then determine the location of v_i by identifying adjacent nodes v_j such that there exists $e_k \in E$ with $e(e_k) = (v_i, v_j)$ or (v_j, v_i) and $l(v_j) \cap C \neq \emptyset$, i.e., v_j is an instance of a class in the ontology. For both v_i and its location node v_j, we extract their associated properties, $p(v_i)$ and $p(v_j)$. This process is repeated for all $s_i \in T$, and the retrieved property–value pairs, together with the sensor measurements s in the current transaction, are combined into a semantically enriched transaction. Property values from neighbors of node v_j and the relations can also be in the

transaction set, depending on the application. Note that the granularity of semantic modeling of knowledge graphs can impact the quality of the rules learned.

Example: To continue our WDN example, Figure 2 shows part of a WDN knowledge graph that we constructed for the LeakDB (Vrachimis et al. 2018) dataset. The figure contains 3 Pipe nodes in orange, 3 junction nodes in blue, 4 sensor nodes in beige, the properties of `Junction_5`, and the connection between the nodes. The figure shows that `Junction_5` has a water pressure and demand sensor, `Junction_4` has a pressure sensor, and `Junction_6` has a demand sensor. Sensor measurements are stored separately and not on the graph. Assume that the sensor measurements are aggregated into time frames by applying an average function per minute of measurements for each sensor, and the numerical values are discretized into intervals using equal-frequency discretization applied to each sensor type (e.g., all water flow sensors). To apply the semantic enrichment on the measurements from the pressure sensor placed in `Junction_5`, we first retrieve this node from the graph, its location node `Junction_5`, and all the properties of `Junction_5` that are shown on the right side of the figure, and aggregate them with the sensor measurements. We repeat this process for all sensors and obtain a single semantically enriched transaction. This process is then executed for each transaction. Thus, as the output of the data preparation stage, we obtain a semantically enriched transaction dataset.

Training and Autoencoder Architecture This step of the pipeline vectorizes the semantically enriched transactions and trains an under-complete denoising Autoencoder (Vincent et al. 2008) for ARM. Input transactions in standard ARM literature consist of binary attributes indicating the presence or absence of feature classes (Agrawal et al. 1994; Kaushik et al. 2023). In line with the literature, we apply one-hot encoding to the semantically enriched transactions and obtain vectors of 0 and 1. The autoencoder is only aware of the binary vectors, and our rule extraction algorithm (Algorithm 1) keeps track of which feature value is fed into which neuron of the autoencoder. Let j be the number of sensors in T, i be the number of semantic property values in U mapped to each $s_{1..j}$, z be the number of classes per input feature for simplicity, and n be the number of transactions. In practice, i and z usually are different per $s_{1..j}$, and property values $p \in U$ can be different per transaction if G changes over time. Input transactions to the Autoencoder look as follows:

$$\begin{aligned}\Big[&\{m1^1_{s_1},\dots,m1^z_{s_j},\\ &\quad p1^1_{s_{11}},\dots,p1^z_{s_{11}},\dots,p1^z_{s_{1_i}},\dots,p1^z_{s_{j_i}}\},\\ &\dots\\ &\{mn^1_{s_1},\dots,mn^z_{s_j},\\ &\quad pn^1_{s_{11}},\dots,pn^z_{s_{11}},\dots,pn^z_{s_{1_i}},\dots,pn^z_{s_{j_i}}\}\Big]\end{aligned} \tag{2}$$

We employ an under-complete denoising Autoencoder which creates a lower-dimensional representation of the noisy variant of its input (encoder) and then reconstructs the noise-free input from the dimensionally reduced version (decoder). In this way, the model learns a neural representation of the input data and becomes more robust to noise. After the one-hot encoding, a random noise $N \sim [-0.5, 0.5]$ is added to each item in (2). An under-complete Autoencoder, by creating a lower-dimensional representation of the input data (code layer) and then reconstructing the initial input from the code layer, learns the most prominent features of the input data rather than all. In the scope of ARM, we argue (and later show empirically) that this results in learning rules with high association strength, rather than obvious rules, hence reducing the number of rules learned.

Our under-complete denoising autoencoder has 3 layers for encoding and decoding units. During training, $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ is preferred in the hidden layers as activation function. After the encoding and decoding, the softmax function σ is applied per feature, $\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$, such that the values for classes of a feature sums up to 1 (100%). Each feature f refers to either sensor measurements (m) or each of associated semantic property values (p) from G, which was exemplified in (2):

$$\sum_{j=1}^{c_i} \sigma(f_i)_j = \sum_{j=1}^{c_i} \frac{e^{f_{i,j}}}{\sum_{k=1}^{c_i} e^{f_{i,k}}} = \frac{\sum_{j=1}^{c_i} e^{f_{i,j}}}{\sum_{k=1}^{c_i} e^{f_{i,k}}} = 1 \tag{3}$$

This leads to having one probability distribution per feature in the output layer. We will later exploit this to infer which class(es) of a feature are associated with other features' class values. As the lost function, binary cross entropy (BCE) loss is applied per feature and the results are aggregated:

$$\begin{aligned}BCE(F) &= \sum_{i=1}^{k} BCE(f_i)\\ &= \sum_{i=1}^{k} \frac{1}{c_i} \sum_{j=1}^{c_i} \Big[- y_{i,j} \log(p_{i,j})\\ &\qquad - (1 - y_{i,j}) \log(1 - p_{i,j}) \Big]\end{aligned} \tag{4}$$

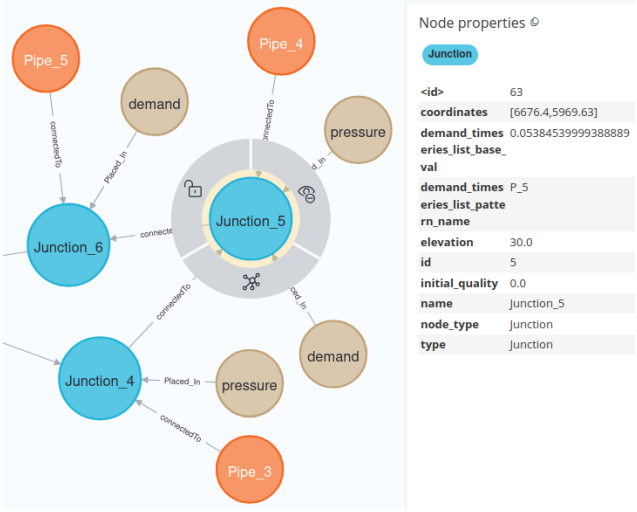

Figure 2. Part of a WDN knowledge graph for LeakDB (Vrachimis et al. 2018) using Neo4j*.

*https://neo4j.com/

where $p_{i,j}$ refers to $\sigma(f_i)_j$ and $y_{i,j}$ refers to initial noise-free version of f_i^j. We then calculate the loss between Autoencoder reconstruction and the initial noise-free input and propagate the loss backward. When learning to reconstruct a given input, we hypothesize that the autoencoder also learns the associations between input feature classes.

Example: Returning to our WDN example (Figure 2), consider the pressure sensor located at `Junction_5`. For simplicity, assume that its measurements are discretized into three ordinal classes: $\{r_1, r_2, r_3\}$, representing low, medium, and high pressure levels. The sensor is linked to the `Junction_5` node in the knowledge graph, which has the label *Junction*—one of five possible types: $\{Junction, Pipe, Tank, Pump, Reservoir\}$. In addition to the label, `Junction_5` has six other semantic properties shown on the right side of the figure (all except name and id, e.g., elevation), each discretized into three classes (e.g., $\{r_1^{p_1}, r_2^{p_1}, r_3^{p_1}\}$ for property p_1). As a result, this sensor contributes eight categorical features to the autoencoder input: one for the measurement, node label, and six for the node's properties. This step is repeated for each of the sensors, forming a transaction. After one-hot encoding, features become binary vectors. The autoencoder processes the vectors and outputs probability distributions via softmax per feature (e.g., eight probability distributions for the pressure sensor placed at `Junction_5`). Equation (4) calculates the difference between the initial noise-free input and the final output, and propagates the loss backward.

Other parameters used in the training, such as the learning rate, are described in Section Training and Execution. Note that some parts of the architecture are kept flexible as they may vary depending on the downstream task to which the proposed approach is applied, such as the type of discretization and sensor data aggregation.

Rule Extraction The last step of our pipeline is to extract association rules from a trained Autoencoder using Algorithm 1. Aerial is a Neurosymbolic approach to rule mining as it combines a neural network (an autoencoder) with an algorithm that can extract associations in the form of logical rules from a neural representation of input data created by training the autoencoder. **Note** that any other ARM algorithm can be used within the pipeline after the semantic enrichment step.

Intuition: Aerial exploits the reconstruction loss of a trained Autoencoder to learn associations. If reconstruction for an input vector with marked features, *test vector*, is more successful than a *similarity threshold*, then we say

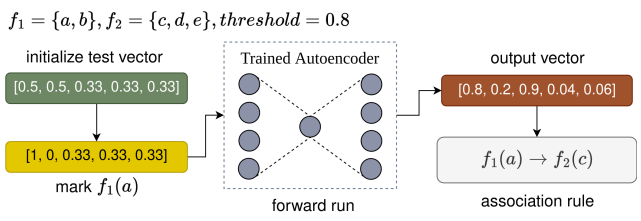

Figure 3. An illustration of association rule extraction from a trained Autoencoder with our Aerial approach.

Algorithm 1 Aerial rule extraction algorithm.

1: **procedure** AERIAL(X, AE, τ_s, a)
X: input transactions, AE: trained autoencoder, τ_s: similarity threshold, a: max antecedent size
2: $\mathcal{R} \leftarrow \emptyset$
3: $\mathcal{C} \leftarrow$ COMBINATIONS(X.features, a)
4: **for all** $A \in \mathcal{C}$ **do**
5: $v_0 \leftarrow$ UNIFORMVECTOR(X.features)
6: $V \leftarrow$ MARKFEATURES(v_0, A)
7: **for all** $v \in V$ **do**
8: $\hat{y} \leftarrow AE$.FORWARD(v)
9: **if** SIM($\hat{y}_A, v_A$) $< \tau_s$ **then**
10: **continue**
11: **for all** $f \in$ X.features $\setminus A$ **do**
12: **if** $\hat{y}_f > \tau_s$ **then**
13: $\mathcal{R} \leftarrow \mathcal{R} \cup \{(A \Rightarrow f)\}$
14: **return** $\mathcal{R}$

that the marked features imply the successfully reconstructed features. Marking features is done by assigning 1 (100%) probability to a certain class value for a feature, 0 to the other classes for the same feature, and assigning equal probabilities to the rest of the features in an input vector. In the case of IoT data, the input test vector represents a partially defined environment via the marked features, and the autoencoder reconstructs the co-occurrences with the rest of the environment. Therefore, we hypothesize that the marked feature classes in the test vector represent the antecedents of a rule, while the reconstructed feature classes represent the consequents.

Example: Figure 3 depicts an example rule extraction process. Assume that there are only two features in the input vector with 2 and 3 possible class values, namely $f_1 = \{a, b\}$ and $f_2 = \{c, d, e\}$. One-hot encoded versions of f_1 and f_2 can be represented with 5 digits. Assume we want to test whether $f_1(a)$ implies a certain class value of f_2. First, we initialize a vector of equal probability distributions per feature, $[0.5, 0.5, 0.33, 0.33, 0.33]$, where the first two digits correspond to the classes of f_1 and the last three to the classes of f_2 respectively. Next, we mark $f_1(a)$ by assigning a probability of 1 (100%) to the first digit in the vector and 0 to the rest of the classes of f_1, $[1, 0, 0.33, 0.33, 0.33]$. We call this a $test\_vector$. We perform a forward run on the trained Autoencoder with this vector. Assume that the output is $[0.8, 0.2, 0.9, 0.04, 0, 06]$. The third output digit that corresponds to $f_2(c)$ is bigger than the threshold, 0.8. Therefore, we say that $f_1(a) \rightarrow f_2(c)$.

Algorithm: The rule extraction algorithm is given in Algorithm 1. The parameters are the set of input vectors (X), a trained Autoencoder (AE), a similarity threshold (τ_s), and a maximum number of antecedents (a) that the rules will contain. Note that the input vectors X keep track of which sensor value or semantic property value is fed into which neuron, with a dictionary of key value pairs in $X.features$. Based on the maximum number of antecedents a, in line 3, the algorithm creates combinations of features as candidate antecedents ($\mathcal{C}$). For instance, to test whether values of features f_1 and f_2 are associated with other features, a tuple of (f_1, f_2) is created. Lines 4-13 go through each feature

tuple $A \in \mathcal{C}$ and first create an initial test vector with all equal probabilities per feature (line 5). This corresponds to the $[0.5, 0.5, 0.33, 0.33, 0.33]$ vector given in the example above. Line 6 marks feature values in the $\mathcal{C}$ with a probability of 1, and returns a list of test vectors V. This corresponds to the $[1, 0, 0.33, 0.33, 0.33]$ vector in the example. Lines 7-13 perform a forward run per test vector and; i) check whether output probabilities for the marked features are higher than the given threshold (lines 9-10), ii) find features (other than marked features, no self-implications) that have higher probability than the given threshold, which are added to the rule list as consequences together with the marked features which are the antecedents (lines 11-13). The algorithm's **time complexity** in big O notation is $O(|F|^{a+1})$ since it iterates over all combinations of a antecedents from $|F|$ features and performs operations linear in the total number of feature classes. Since a is typically small (usually less than 10 for most practical applications), the complexity is polynomial in $|F|$. For the extended analysis of Algorithm 1, please see Appendix **Time Complexity Analysis of Aerial**.

Evaluation

Two different experimental settings are used to evaluate the two main contributions of this paper.

First, to evaluate the impact of utilizing semantics in rule mining from IoT data (our proposed pipeline), we designed the Experimental Setting 1 where we run multiple ARM methods (including ours) with and without semantics. And we compare the results before and after adding semantics based on various rule quality criteria and execution times. Second, in Experimental Setting 2, we evaluate our proposed Neurosymbolic ARM method Aerial across eight baselines of different types, including exhaustive, optimization-based, and DL-based ARM algorithms. We made our best effort to compare different types of approaches fairly, based on the number of rules, rule quality, and execution time. Furthermore, we investigated the impact of the similarity threshold hyperparameter of Aerial in a separate experiment (Experiment 3).

This section first describes common elements across both settings such as datasets, and then describes setting-specific points including baselines. Additional experiments that are not directly relevant to the two settings are given in Appendix Additional Experiments.

Open source. The source codes of Aerial, baselines, and knowledge graph construction are written in Python and are available online together with all the datasets: https://github.com/DiTEC-project/semantic-association-rule-learning.

Hardware. All experiments ran on an AMD EPYC 7H12 64-core CPU with 256 GiB memory. No GPUs were used.

Setup

This section describes the **common** elements for both of the evaluation settings.

Datasets. 3 open-source IoT datasets from two different domains, water networks and energy, are used for all the experiments. A knowledge graph is created per dataset by mapping metadata about each component to domain-specific data structures. **LeakDB** (Vrachimis et al. 2018) is an artificially generated realistic dataset in water distribution networks. It contains sensor data from 96 sensors of various types, and semantic information such as the formation of the network, sensor placement, and properties of components. **L-Town** (Vrachimis et al. 2020) is another dataset in the water distribution networks domain with the same characteristics. It has 118 sensors. **LBNL** Fault Detection and Diagnostics Dataset (Granderson et al. 2022) contains sensor data from 29 sensors and semantics for Heating, Ventilation, and Air Conditioning (HVAC) systems. As semantic properties, it only includes a *type* property.

Training and Execution. The Aerial Autoencoder is trained for each dataset. The training parameters found via grid search are as follows: learning rate is set to $5e^{-3}$, the models are trained for 5 epochs, Adam (Kingma and Ba 2014) optimizer is used for gradient optimization with a weight decay of $2e^{-8}$, and the noise factor for the denoising Autoencoder is 0.5. All experiments are repeated 20 times over 20 randomly selected sensors for each dataset, and the average results are presented unless otherwise specified. The random selection is done by picking a random sensor node on the knowledge graph, and traversing through the first, second, etc. neighbors until reaching 20 sensor nodes. Equal-frequency discretization (Foorthuis 2020) with 10 intervals is used for numerical features for the methods that require pre-discretization (Table 3).

Evaluation Metrics. The most common way of evaluating ARM algorithms is to measure the quality of the rules from different aspects as there is **no single criterion** that fits all cases. In the evaluation, we used the standard metrics in ARM literature which are support, confidence, data coverage, number of rules, and execution time (Kaushik et al. 2023; Telikani et al. 2020). In addition, we selected Zhang's metric (Yan et al. 2009) to evaluate the association strength of the rules, commonly used in many open-source libraries including MLxtend (Raschka 2018) and NiaARM (Stupan and Fister 2022). The definitions are given below:

- **Support**: Percentage of transactions with a certain item or rule, among all transactions (D): $support(X \rightarrow Y) = \frac{|X \cup Y|}{|D|}$.
- **Confidence**: Conditional probability of a rule, e.g., given the transactions with the antecedent X in, the probability of having the consequent Y in the same transaction set: $confidence(X \rightarrow Y) = \frac{|X \cup Y|}{|X|}$.
- **Rule Coverage**: Percentage of transactions that contains antecedent(s) of a rule: $coverage(X \rightarrow Y) = support(X)$.
- **Data Coverage**: It refers to the percentage of transactions to which the learned rules are applicable.
- **Zhang's Metric**: This metric also considers the case in which the consequent appears alone in the transaction set, besides their co-occurrence, and therefore measures dissociation as well. A score of > 0 indicates an association, 0 indicates independence and < 0 indicates dissociation: $zm(X \rightarrow Y) = \frac{confidence(X \rightarrow Y) - confidence(X' \rightarrow C)}{max(confidence(X \rightarrow Y), confidence(X' \rightarrow Y))}$ in which X' refers to the absent of X in the transaction set.

Hyperparameters. There are two parameters to our Aerial approach: similarity threshold and number of

Table 3. Overall comparison of evaluated ARM approaches.

	Exhaustive	DL-based	Optimization
Semantic Assoc. Rules	Supports	Supports	Does not directly support
Rule Constraints	Supports constraints	Supports constraints	Does not support
Number of Rules	Very high	Low with full data coverage	Medium to High
Rule Length	Controllable	Controllable	Uncontrollable
Rule Quality	Controllable	Partially controllable	Partially controllable
Discretization	Required	Required	Not required

antecedents. The effect of similarity threshold on rule quality is investigated in Experiment 3. The effect of the number of antecedents on execution time and the number of rules learned is investigated in Experiment 2.1.

Experimental Settings

This section describes the two core experimental settings together with baselines in each setting. Please refer to Table 3 for baseline methods described in the settings below.

Setting 1: Semantics vs without Semantics. To show that semantics can enable learning more generically applicable rules, two different ARM algorithms, our Aerial approach and a popular exhaustive method FP-Growth (Han et al. 2000), are run with and without semantically enriched sensor data. Two algorithms are used to show that including semantics is beneficial regardless of the ARM method applied. The results are compared based on the number of rules, average rule support, confidence and coverage, and execution time. FP-Growth is implemented using MLxtend (Raschka 2018). This experimental setting does not aim to evaluate the capability of each ARM algorithm to learn high-quality rules, but only focuses on the impact of utilizing semantics. The prior is performed as part of the second experimental setting.

Setting 2: Aerial vs state-of-the-art. The goal is to evaluate the proposed Aerial method for IoT data, and the experiments are run on sensor data with semantics. The only existing semantic ARM approach Naive SemRL (Karabulut et al. 2023) is chosen as a baseline and executed with the exhaustive FP-Growth (as in the original paper) and HMine algorithms. In addition, the optimization-based NARM method TS-NARM (Fister Jr et al. 2023) with standard confidence metric as optimization goal is run with 5 algorithms (as in the original paper), Differential Evolution (DE) (Storn and Price 1997), Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995), Genetic Algorithm (GA) (Goldberg 2013), jDE (Brest et al. 2006), and LSHADE (Viktorin et al. 2016)). TS-NARM is implemented using NiaPy (Vrbančič et al. 2018) and NiaARM (Stupan and Fister 2022), and FP-Growth and HMine are implemented using Mlxtend (Raschka 2018). All rule quality criteria described earlier are used in the comparison.

Table 4. Aerial, baselines, and their parameters (Optimization refers to TS-NARM and Exhaustive to Naive SemRL. See Experiment 6 in Appendices for the evaluation of ARM-AE).

Algorithm	Type	Parameters
Aerial	DL-based	antecedents=2, similarity=0.8
ARM-AE	DL-based	antecedents=2, likeness=0.8
DE	Optimization	$F = 0.5, CR = 0.9$
GA	Optimization	$p_m = 0.01, p_c = 0.8$
PSO	Optimization	$c_1 = 0.1, c_2 = 0.1, w = 0.8$
LSHADE	Optimization	$NP_{max} = 18.NP, NP_{min} = 4.NP, H = 5, p = 0.1, r^{arc} = 2$
jDE	Optimization	$F^{(0)} = 0.5, CR^{(0)} = 0.9, \tau = 0.1$
FP-Growth	Exhaustive	(both) antecedents=2, min_support=(Aerial.rules. avg_support/2), min_confidence=0.8.
HMine	Exhaustive	

ARM-AE (Berteloot et al. 2023), another Autoencoder-based ARM method, uses an Autoencoder with equal-sized layers (no dimensionality reduction), does not distinguish between features (e.g., by applying softmax per feature as in our approach), and assumes that input to the trained Autoencoder represents consequent while the output represents an antecedent. We argue that this assumption does not hold, and the evaluation of ARM-AE resulted in exceptionally low rule quality both in their paper (33% confidence on the Nursery dataset and 50% confidence on the chess dataset) and also based on our results. Therefore, we opted not to include it in the core Evaluation section. Please refer to Experiment 6 in Appendices for the evaluation of ARM-AE.

Challenges in comparison. The distinct nature of different types of algorithms makes comparability a challenge. The exhaustive algorithms can find all rules with a given support and confidence threshold. The execution time of the 5 optimization-based approaches (TS-NARM) is directly controlled by the pre-set maximum evaluation parameter. And running them longer leads to better results up to a certain point (Section Aerial vs state-of-the-art). The quality of the rules learned by the DL-based ARM approaches depends on the given similarity threshold parameter (or likeness for ARM-AE). Given these differences, we made our best effort to compare algorithms fairly and showed the **trade-offs under different conditions**. Table 4 lists the parameters of each algorithm for both of the settings, unless otherwise specified. For TS-NARM, the population size is set to 200 which represents an initial set of solutions, and the maximum evaluation is set to 50,000 which represents the number of fitness function evaluations before convergence. The parameters of the 5 optimization-based methods, population size, and maximum evaluation count are the same as in the original paper. The antecedent length of both exhaustive and DL-based ARM methods is set to 2 for fairness unless otherwise specified.

Table 5. Comparison of ARM on sensor data with semantics (w-s, our pipeline) and without (wo-s), showing a significant increase in support and rule coverage (cov.) with semantics (FP-G = FP-Growth, Conf = Confidence).

	# Rules	Support	Cov.	Conf.
	w-s|wo-s	w-s|wo-s	w-s|wo-s	w-s|wo-s
		LeakDB		
FP-G	103K|9K	**0.41**|0.19	**0.43**|0.2	0.95|0.97
Aerial	554|2.5K	**0.54**|0.25	**0.59**|0.3	0.91|0.87
		L-Town		
FP-G	25K|5K	**0.86**|0.36	**0.9**|0.38	0.96|0.96
Aerial	1K|2.5K	**0.59**|0.39	**0.65**|0.45	0.91|0.88
		LBNL		
FP-G	7K|2K	**0.84**|0.73	**0.85**|0.75	0.98|0.99
Aerial	73|258	**0.74**|0.65	**0.74**|0.66	1.0|0.99

The minimum support threshold of the exhaustive methods is set to half of the average support of the rules learned by our Aerial method so that both approaches will result in a similar average support value for fairness.

Experimental Results

This section presents the experimental results for both settings.

Setting 1: Semantics vs without Semantics.

Experiment 1.1: Rule Quality. Table 5 shows the results for running Aerial and FP-Growth with (w-s) and without (wo-s) semantic properties. Average support and rule coverage for both algorithms on all datasets increased significantly upon including semantics. The rule count is increased for FP-Growth with semantics, while it decreased with our Aerial approach. The confidence values did not change significantly.

The results indicate that association rules learned from sensor data and semantics are more generically applicable than rules learned from sensor data only, as the support and rule coverage values are significantly higher. Furthermore, this experiment is repeated with varying numbers of sensors, and the results (Experiment 4 in Appendices) show that a higher number of sensors results in more generically applicable rules. The comparison of rule count and confidence for different approaches will be investigated in Experimental Setting 2.

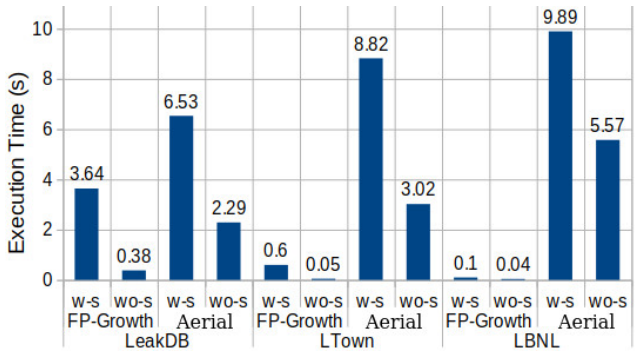

Figure 4. Effect of using semantics (indicated as w-s, and wo-s for without semantics) on execution time.

Table 6. Association rule examples with (top) and without (bottom) semantics learned from LeakDB dataset.

Association Rule	Support	Coverage
if a water flow sensor s1 is inside a Pipe with length 843-895, and a water demand sensor s2 inside a Junction measures 13-17, then s1 must measure between 23-31.	0.5	0.54
if the water flow sensor inside Pipe_28 measures between 23-31, then the water flow sensor inside Pipe_18 must measure between -767–471.	0.43	0.52

Experiment 1.2: Execution Time. Figure 4 shows the effect of including semantics in the execution time of FP-Growth and Aerial (training + rule extraction time). The increase in the execution time of FP-Growth is 3-12 times while it is 2-3 times in Aerial and is more stable. However, since the semantic association rules have higher support and data coverage, a smaller number of them can have full data coverage (which is the case for Aerial and will be investigated in Experimental Setting 2). Therefore, we argue that the increment in the execution time is acceptable. Note that despite FP-Growth running faster with the parameters given in Table 4, it is strictly dependent on the preset minimum support threshold value and it runs slower for lower thresholds. This is investigated in Experiment 2.1.

Illustration. Table 6 shows two example association rules learned from the LeakDB dataset. The first rule is based on the semantics and sensor data and has higher support and coverage than the second rule, which is only about two specific water flow sensors.

Setting 2: Aerial vs state-of-the-art

Experiment 2.1: Execution Time and Number of Rules Analysis. This experiment investigates how execution time and the number of rules change for the proposed Aerial approach and baselines depending on their relevant parameters.

The exhaustive methods' execution time and number of rules they mine are strictly dependent on the preset minimum support threshold and the number of antecedents. Figure 5 shows how the number of rules and execution time change based on antecedents (for 1, 2, 3, and 4 antecedents) and minimum support thresholds (for 0.05, 0.1, 0.2 and 0.3). The results show that the execution time increases as the support threshold decreases and the number of rules increases above 10 million for LeakDB while it reaches 1-2 million for LBNL and L-Town datasets which are highly costly to post-process. Similarly, as the number of antecedents increases the number of rules reaches the levels of millions, while the execution time reaches minutes. The execution did not terminate for the LeakDB dataset when using 4 antecedents after 30 minutes.

Execution time, number of rules as well as the quality of the rules mined by the optimization-based methods (TS-NARM) strictly depend on the number of evaluations. Table 7 shows the effect of the maximum evaluations parameter on the execution time, number of rules, and confidence of the rules for the LeakDB dataset (the results are consistent

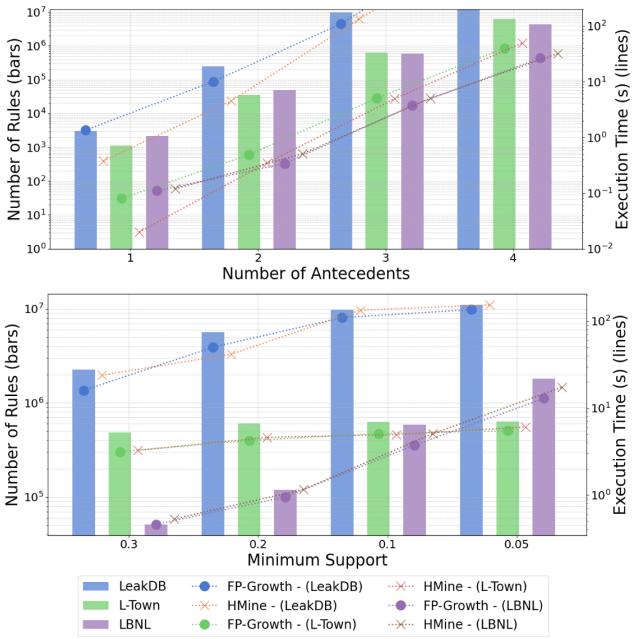

Figure 5. Exhaustive methods have higher execution times (dotted lines) and produce a larger number of rules (bars) as the number of antecedents (top chart, conf=0.8, sup=0.1) increase or min. support threshold (bottom chart, antecedents=3) decrease.

across datasets, see Experiment 5 in Appendices). The results show that longer executions lead to a higher number of rules with higher confidence for all 5 algorithms. 50,000 is chosen as the maximum evaluation for the rule quality experiment (Experiment 2.2) as this is also the case in the original paper.

Lastly, the rule extraction time of the proposed Aerial approach is affected by the number of antecedent parameters, as it increases the number of test vectors used in the algorithm. Figure 6 shows the effect of increasing the number of antecedents on the number of rules and execution time. The number of learned rules is 10-100 times lower than the exhaustive methods. Exhaustive methods run slower on datasets with low support rules, LeakDB (see Tables 5 and 8), while running faster on datasets with high support rules, L-Town and LBNL. Both Aerial and exhaustive methods run faster than the optimization-based methods for at least a low-to-medium-size antecedent (1-4).

Experiment 2.2: Rule Quality Analysis. The goal of this experiment is to assess the quality of rules found by

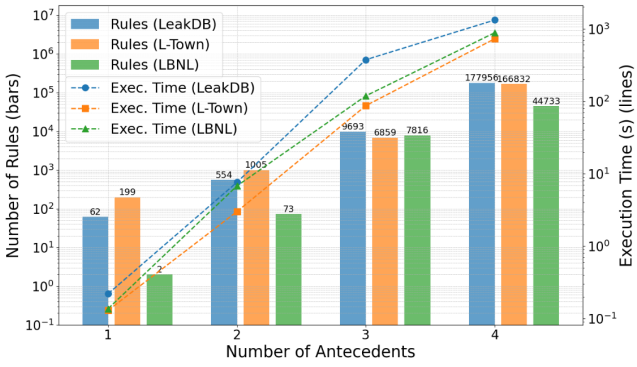

Figure 6. Execution time and the number of rules learned by Aerial depend on the number of antecedents.

Table 7. TS-NARM needs long evaluations (Evals.) for good performance (LeakDB, Conf=Confidence). The results are consistent across all datasets (Experiment 5 in Appendices).

Evals.	Algorithm	# Rules	Time(s)	Conf.
10000	DE	1388	109.24	0.69
	GA	106	120.58	0.47
	PSO	3281	115.39	0.81
	LSHADE	1786	133.01	0.77
	jDE	1578	88.48	0.75
30000	DE	6868	344.73	0.80
	GA	472	393.73	0.40
	PSO	10491	425.44	0.74
	LSHADE	9914	411.82	0.94
	jDE	5441	300.94	0.78
50000	DE	32525	782.72	0.81
	GA	11578	650.88	0.60
	PSO	32502	784.96	0.84
	LSHADE	34887	981.07	0.99
	jDE	24978	567.10	0.83

Aerial and baselines, highlighting the trade-offs between algorithms. **How to read the results?** The evaluation results are shown in Table 8 and the highest scores are intentionally **not** emphasized as ideal rule quality values can vary by task. As an example, high-support rules can be good at discovering trends in the data while low-support rules may be better at detecting anomalies. The focus is on understanding each algorithm's strengths under diverse conditions, therefore, results **should be interpreted together**.

Table 8. Rule qualities of all algorithms across all datasets (Exhaustive = FP-Growth and HMine, Sup = Support, Conf = Confidence, Cov = Data Coverage).

Algorithm	# Rules	Sup.	Conf.	Cov.	Zhang
LeakDB					
Exhaustive	103283	0.41	0.95	1.0	0.82
DE	11841	0.19	0.77	1.0	0.24
GA	663	0.08	0.46	1.0	0.15
PSO	12566	0.08	0.75	1.0	0.16
LSHADE	23605	0.4	0.98	1.0	0.41
jDE	10270	0.25	0.77	1.0	0.29
Aerial	554	0.54	0.91	1.0	0.9
L-Town					
Exhaustive	25421	0.86	0.96	1.0	-0.18
DE	15163	0.11	0.76	1.0	0.13
GA	1384	0.03	0.37	1.0	0.05
PSO	15651	0.03	0.75	1.0	0.04
LSHADE	22825	0.39	0.96	1.0	0.39
jDE	11255	0.19	0.78	1.0	0.21
Aerial	1005	0.59	0.91	1.0	0.4
LBNL					
Exhaustive	7220	0.84	0.98	1.0	0.01
DE	17393	0.22	0.79	1.0	0.23
GA	580	0.1	0.45	1.0	0.11
PSO	17944	0.06	0.8	1.0	0.07
LSHADE	30799	0.52	0.98	1.0	0.52
jDE	15594	0.28	0.77	1.0	0.29
Aerial	73	0.74	1.0	1.0	0.15

Aerial was able to find a concise set of rules that have full data coverage with 90%+ confidence, the highest association strength (Zhang's metric) in the LeakDB and L-Town datasets, and the second highest in the LBNL dataset. The FP-Growth and HMine algorithms yield the same results as they are *Exhaustive*. They have full data coverage, resulted in a high number of rules except for the LBNL dataset, and had very low association strength on L-Town and LBNL. The optimization-based methods had low confidence except for the LSHADE which had a high confidence score on all datasets, the highest association strength among other optimization-based methods, and the highest in LBNL among all algorithms.

These results show that Aerial was able to find prominent patterns in the datasets that have high association strength and achieved full data coverage with a concise number of rules in comparison to state-of-the-art, which was the initially stated goal. In addition, Experiment 3 shows that higher similarity thresholds in Aerial lead to even higher quality association rules.

Experiment 3: Effect of similarity threshold on rule quality in Aerial. The similarity threshold parameter of our Aerial method affects the quality of the rules learned. This experiment investigates the effect of the similarity threshold parameter of Aerial on all 3 datasets.

Table 9 presents the results for all 3 datasets. We observe that as the similarity threshold increases, the number of learned rules decreases, while the average support, confidence, and association strength (Zhang's metric) increase, with the exception when the similarity threshold is 0.9. In that case, we observe a decrease in the association strength except in the LeakDB dataset. We argue that this is due to both the relatively low number of rules (6 and 116) learned in comparison to a relatively higher number of rules in LeakDB (412), and LeakDB being a low-support dataset (see Table 5), meaning that the average rule support for association rules in the LeakDB dataset is significantly lower than the other two datasets.

These results imply that increasing the similarity threshold results in more prominent rules but fewer in number, acting similarly to the minimum confidence threshold of the exhaustive algorithms.

Table 9. Aerial learns a more concise set of higher quality rules as the similarity threshold (Sim.) increases (Conf = Confidence, Cov = Rule Coverage, Zhang = Zhang's Metric).

Sim.	# Rules	Support	Conf.	Cov.	Zhang
		LeakDB			
0.9	412	0.47	**0.92**	**1**	**0.91**
0.8	554.4	**0.54**	0.91	**1**	0.9
0.7	1845	0.3	0.88	**1**	0.83
0.6	3027	0.25	0.84	**1**	0.79
0.5	**9831**	0.28	0.73	**1**	0.58
		L-Town			
0.9	116	**0.7**	**0.98**	**1**	0.06
0.8	1005.2	0.59	0.91	**1**	**0.4**
0.7	1860	0.39	0.82	**1**	0.33
0.6	3851	0.32	0.76	**1**	0.32
0.5	**23017**	0.38	0.65	**1**	0.2
		LBNL			
0.9	6	**0.75**	**1**	0.71	0
0.8	73	0.74	**1**	**1**	**0.15**
0.7	826	0.66	0.86	**1**	0.13
0.6	1730	0.64	0.75	**1**	0.08
0.5	**2877**	0.63	0.7	**1**	0.06

Discussion

This section discusses and summarizes the experimental findings.

Semantics for generalizability. The results in Experimental Setting 1 showed that learning association rules from both static and dynamic data in IoT systems results in rules that have higher support and data coverage and, therefore, are more generically applicable than rules learned from sensor data only. The experiments also showed that including semantics is beneficial regardless of the ARM approach as the results were similar for both exhaustive FP-Growth and our proposed Aerial approach.

Neurosymbolic methods can help learning a concise set of high-quality rules. As semantic enrichment of sensor data increases data dimension, current ARM methods result in a higher number of rules which is already identified as a research problem in the ARM literature. As an alternative, our proposed Neurosymbolic Aerial rule mining approach can learn a concise number of rules with full data coverage, high confidence, and association strength, which is demonstrated in Experimental Setting 2. We argue that this is thanks to utilizing an under-complete autoencoder that can capture more prominent features of the input data in its code layer, rather than all features which may lead to obvious rules with low association strength. We believe that there is potential in the direction of neurosymbolic rule learning, and Aerial is a strong initial step.

Execution time. Semantic enrichment increases execution time by 2-3 times for Aerial and 3-12 times for exhaustive methods, as shown in Experiment 1.2. However, semantic association rules have higher support and rule coverage, and a substantially smaller number of them can have full data coverage, therefore we argue that the increment is acceptable. The exhaustive methods perform poorly on low-support (LeakDB) datasets with a low minimum support threshold and also perform poorly with a high number of antecedents, as demonstrated in Experiment 2.1. This experiment also showed that Aerial runs faster than the exhaustive methods on low-support datasets and Aerial's execution time does not depend on the datasets' support characteristics. Note that Aerial can be parallelized and run on a GPU (similar to the exhaustive methods). The optimization-based methods' execution time is directly controlled by the preset maximum evaluation parameter. Longer executions are required to obtain higher-quality rules and this also results in a high number of rules, which are costly to process and maintain. Aerial is faster than the optimization-based methods for learning rules with low-to-medium-size antecedents (1 to 4). Note that the number of antecedents for the optimization-based methods can not be controlled.

Variations of Aerial. Many existing ideas in ARM literature can be integrated into our Aerial approach. For instance, in ARM with item constraints, rules of interest are

described using a taxonomy or an ontology, and then ARM algorithms focus on those rules only, which speeds up the execution and leads to a smaller number of rules (Srikant et al. 1997; Baralis et al. 2012). A similar mechanism can be implemented in Aerial, simply by creating the test vectors in a way that only the items of interest are marked. This will reduce the number of test vectors, and thus reduce the execution time and the number of learned rules. Similarly, top-k rule mining focuses on mining top-k association rules with the highest quality (Fournier-Viger et al. 2012). An analogous process in Aerial is to find the top-k rules with the highest output probability. As shown in Experiment 3, higher output probabilities lead to higher quality rules.

Real-world scalability. Real-world large-scale IoT data differs from the tabular datasets that most state-of-the-art ARM methods focus on. Each sensor is treated as a different data dimension, hence resulting in potentially extremely high-dimensional data especially upon including semantic properties. Therefore, utilizing neural networks' capability to process high-dimensional data is essential. Both time complexity (given in Appendix Time Complexity Analysis of Aerial) and execution time analyses (Experiments 1.2 and 2.1) show that our Neurosymbolic rule mining approach is scalable on large-scale IoT data. Extrapolating the execution times (training + rule extraction) shown in Figure 6, Aerial can scale up to tens of thousands of sensors on a laptop (see Hardware) in a day. The training is linear over the number of features (sensor measurements and associated semantic properties) and the number of transactions, and the rule extraction stage is polynomial over the number of feature classes. Algorithm 1 is parallelizable as test vectors per feature subsets are created and processed independently. Another advantage of our Neurosymbolic ARM approach over the commonly used algorithmic approaches, such as FP-Growth, is the option to leverage neural network–specific optimizations that significantly speed up execution and improve scalability. Some examples are batch normalization, both in training and when performing forward runs with the test vectors, and quantization and pruning to reduce model size and inference time without compromising performance.

Practical implications. Besides knowledge discovery, ARM is a cornerstone of interpretable machine learning models such as rule list classifiers, which are the standard approach to high-stakes decision-making (Rudin 2019). Such models process a given set of association rules to find a small subset that can be used to explain a certain class label. One example of such high-stakes decision-making in the scope of IoT systems, continuing our WDN example, is leakage detection in WDNs. In this case, our proposed pipeline can be used to learn association rules with a class label (e.g., 0: no leakage, 1: leakage) on the consequent side (ARM with item constraints as explained in the Variations of Aerial discussion point). The rules are then passed to a rule-based classifier such as CORELS (Angelino et al. 2018) to build a classifier that can detect leakages. This example can easily be extended to other anomaly detection tasks, including digital twins of IoT systems. In addition, our proposed pipeline and rule mining method Aerial is integrated into a digital twin architecture of a WDN (Degeler et al. 2024) to detect such abnormalities.

Conclusion and Future Work

This paper introduced two contributions; i) a novel ARM pipeline for IoT systems, and ii) a Neurosymbolic ARM method (Aerial). In contrast to the state-of-the-art, our pipeline utilizes both dynamic sensor data and static knowledge graphs that describe the metadata of IoT systems. Aerial creates a neural representation of given input data using an Autoencoder and then extracts association rules from the neural representation. The experiments showed that the proposed pipeline can learn rules with 2-3 times higher support and coverage, which are more generically applicable than ARM on sensor data only. Moreover, the experiments further demonstrated that Aerial can learn a more concise set of high-quality association rules than the state-of-the-art with full data coverage. Aerial is also compatible with existing work on addressing the high number of rule problems in the ARM literature.

In future work, we first plan to investigate other neural network architectures for their capabilities of learning associations and develop new methods to extract rules from neural representations created using various architectures. Secondly, we plan to investigate the impact of semantic modeling in knowledge graphs on the rules learned, and measure the capability of our approach to capture the semantics of the underlying ontology, such as the symmetry or the transitivity of the relations. Finally, we plan to apply our methods to downstream tasks such as leakage detection in water networks or fault diagnosis in energy systems.

Funding

This work has received support from The Dutch Research Council (NWO), in the scope of the Digital Twin for Evolutionary Changes in water networks (DiTEC) project, file number 19454.

References

Agrawal R, Srikant R et al. (1994) Fast algorithms for mining association rules. In: *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215. Santiago, Chile, pp. 487–499.

Angelino E, Larus-Stone N, Alabi D, Seltzer M and Rudin C (2018) Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research* 18(234): 1–78.

Baralis E, Cagliero L, Cerquitelli T and Garza P (2012) Generalized association rule mining with constraints. *Information Sciences* 194: 68–84.

Barati M, Bai Q and Liu Q (2017) Mining semantic association rules from rdf data. *Knowledge-Based Systems* 133: 183–196.

Berteloot T, Khoury R and Durand A (2023) Association rules mining with auto-encoders. *arXiv preprint arXiv:2304.13717* .

Brest J, Zumer V and Maucec MS (2006) Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: *2006 IEEE international conference on evolutionary computation*. IEEE, pp. 215–222.

Chen S and Guo W (2023) Auto-encoders in deep learning—a review with new perspectives. *Mathematics* 11(8): 1777.

Degeler V, Hadadian M, Karabulut E, Lazovik A, van het Loo H, Tello A and Truong H (2024) Ditec: Digital twin for evolutionary changes in water distribution networks. In: *International Symposium on Leveraging Applications of Formal Methods*. Springer, pp. 62–82.

Degeler V, Lazovik A, Leotta F and Mecella M (2014) Itemset-based mining of constraints for enacting smart environments. In: *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. pp. 41–46. DOI:10.1109/PerComW.2014.6815162.

Dolores M, Fernandez-Basso C, Gómez-Romero J and Martin-Bautista MJ (2023) A big data association rule mining based approach for energy building behaviour analysis in an iot environment. *Scientific Reports* 13(1): 19810.

Fan J, Zhang Y, Wen W, Gu S, Lu X and Guo X (2021) The future of internet of things in agriculture: Plant high-throughput phenotypic platform. *Journal of Cleaner Production* 280: 123651.

Fister I, Iglesias A, Galvez A, Del Ser J, Osaba E and Fister I (2018) Differential evolution for association rule mining using categorical and numerical attributes. In: *Intelligent Data Engineering and Automated Learning–IDEAL 2018: 19th International Conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part I 19*. Springer, pp. 79–88.

Fister Jr I, Fister D, Fister I, Podgorelec V and Salcedo-Sanz S (2023) Time series numerical association rule mining variants in smart agriculture. *Journal of Ambient Intelligence and Humanized Computing* 14(12): 16853–16866.

Foorthuis R (2020) The impact of discretization method on the detection of six types of anomalies in datasets. *arXiv preprint arXiv:2008.12330* .

Fournier-Viger P, Wu CW and Tseng VS (2012) Mining top-k association rules. In: *Advances in Artificial Intelligence: 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012, Toronto, ON, Canada, May 28-30, 2012. Proceedings 25*. Springer, pp. 61–73.

Goldberg DE (2013) *Genetic algorithms*. pearson education India.

Granderson J, Lin G, Chen Y, Casillas A, Im P, Jung S, Benne K, Ling J, Gorthala R, Wen J, Chen Z, Huang S, and Vrabie D (2022) Lbnl fault detection and diagnostics datasets. DOI:10.25984/1881324. URL https://data.openei.org/submissions/5763.

Gruber T (1993) What is an ontology.

Han J, Pei J and Yin Y (2000) Mining frequent patterns without candidate generation. *ACM sigmod record* 29(2): 1–12.

He G, Dai L, Yu Z and Chen CLP (2024) Gan-based temporal association rule mining on multivariate time series data. *IEEE Transactions on Knowledge and Data Engineering* 36(10): 5168–5180. DOI:10.1109/TKDE.2023.3335049.

Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutiérrez C, Kirrane S, Labra Gayo JE, Navigli R, Neumaier S, Ngonga Ngomo AC, Polleres A, Rashid SM, Rula A, Schmelzeisen L, Sequeda JF, Staab S and Zimmermann A (2021) *Knowledge Graphs*. Number 22 in Synthesis Lectures on Data, Semantics, and Knowledge. Springer. ISBN 9783031007903. DOI:10.2200/S01125ED1V01Y202109DSK022. URL https://kgbook.org/.

Jiang H and Meng H (2017) A parallel fp-growth algorithm based on gpu. In: *2017 IEEE 14th International Conference on e-Business Engineering (ICEBE)*. IEEE, pp. 97–102.

Karabulut E, Degeler V and Groth P (2023) Semantic association rule learning from time series data and knowledge graphs. In: *Proceedings of the 2nd International Workshop on Semantic Industrial Information Modelling (SemIIM 2023) co-located with 22nd International Semantic Web Conference (ISWC 2023)*. pp. 1–7. URL https://ceur-ws.org/Vol-3647/SemIIM2023_paper_3.pdf.

Karabulut E, Pileggi SF, Groth P and Degeler V (2024) Ontologies in digital twins: A systematic literature review. *Future Generation Computer Systems* 153: 442–456. DOI:10.1016/j.future.2023.12.013.

Kaushik M, Sharma R, Fister Jr I and Draheim D (2023) Numerical association rule mining: A systematic literature review. *arXiv preprint arXiv:2307.00662* .

Kennedy J and Eberhart R (1995) Particle swarm optimization. In: *Proceedings of ICNN'95-international conference on neural networks*, volume 4. ieee, pp. 1942–1948.

Khedr AM, Osamy W, Salim A and Abbas S (2020) A novel association rule-based data mining approach for internet of things based wireless sensor networks. *Ieee Access* 8: 151574–151588.

Kingma DP and Ba J (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kishore S, Bhushan V and Suneetha K (2021) Applications of association rule mining algorithms in deep learning. In: *Computer Networks and Inventive Communication Technologies: Proceedings of Third ICCNCT 2020*. Springer, pp. 351–362.

Li H, Wang Y, Zhang D, Zhang M and Chang EY (2008) Pfp: parallel fp-growth for query recommendation. In: *Proceedings of the 2008 ACM conference on Recommender systems*. pp. 107–114.

Listl FG, Dittler D, Hildebrandt G, Stegmaier V, Jazdi N and Weyrich M (2024) Knowledge graphs in the digital twin: A systematic literature review about the combination of semantic technologies and simulation in industrial automation. *arXiv preprint arXiv:2406.09042* .

Ma M, Wang P and Chu CH (2013) Data management for internet of things: Challenges, approaches and opportunities. In: *2013 IEEE International conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing*. IEEE, pp. 1144–1151.

Patel HK et al. (2022) An innovative approach for association rule mining in grocery dataset based on non-negative matrix factorization and autoencoder. *Journal of Algebraic Statistics* 13(3): 2898–2905.

Pei J, Han J, Lu H, Nishio S, Tang S and Yang D (2001) H-mine: Hyper-structure mining of frequent patterns in large databases. In: *proceedings 2001 IEEE international conference on data mining*. IEEE, pp. 441–448.

Raschka S (2018) Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software* 3(24). DOI:10.21105/joss.00638. URL https://joss.theoj.org/papers/10.21105/joss.00638.

Rhayem A, Mhiri MBA and Gargouri F (2020) Semantic web technologies for the internet of things: Systematic literature review. *Internet of Things* 11: 100206.

Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence* 1(5): 206–215.

Sarker IH and Kayes A (2020) Abc-ruleminer: User behavioral rule-based machine learning method for context-aware intelligent services. *Journal of Network and Computer*

Applications 168: 102762.

Shabtay L, Fournier-Viger P, Yaari R and Dattner I (2021) A guided fp-growth algorithm for mining multitude-targeted item-sets and class association rules in imbalanced data. *Information Sciences* 553: 353–375.

Shang H, Lu D and Zhou Q (2021) Early warning of enterprise finance risk of big data mining in internet of things based on fuzzy association rules. *Neural Computing and Applications* 33(9): 3901–3909.

Srikant R, Vu Q and Agrawal R (1997) Mining association rules with item constraints. In: *Kdd*, volume 97. pp. 67–73.

Storn R and Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11: 341–359.

Stupan Ž and Fister I (2022) Niaarm: a minimalistic framework for numerical association rule mining. *Journal of Open Source Software* 7(77): 4448.

Sunhare P, Chowdhary RR and Chattopadhyay MK (2022) Internet of things and data mining: An application oriented survey. *Journal of King Saud University-Computer and Information Sciences* 34(6): 3569–3590.

Tamašauskaitė G and Groth P (2023) Defining a knowledge graph development process through a systematic review. *ACM Transactions on Software Engineering and Methodology* 32(1): 1–40.

Telikani A, Gandomi AH and Shahbahrami A (2020) A survey of evolutionary computation for association rule mining. *Information Sciences* 524: 318–352.

Viktorin A, Pluhacek M and Senkerik R (2016) Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on cec2014 benchmark set. In: *2016 IEEE congress on evolutionary computation (CEC)*. IEEE, pp. 4797–4803.

Vincent P, Larochelle H, Bengio Y and Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: *Proceedings of the 25th international conference on Machine learning*. pp. 1096–1103.

Vrachimis S, Eliades D, Taormina R, Ostfeld A, Kapelan Z, Liu S, Kyriakou M, Pavlou P, Qiu M and Polycarpou M (2020) Dataset of battledim: Battle of the leakage detection and isolation methods. In: *Proc., 2nd Int CCWI/WDSA Joint Conf. Kingston, ON, Canada: Queen's Univ*.

Vrachimis SG, Kyriakou MS et al. (2018) Leakdb: a benchmark dataset for leakage diagnosis in water distribution networks:(146). In: *WDSA/CCWI Joint Conference Proceedings*, volume 1.

Vrbančič G, Brezočnik L, Mlakar U, Fister D and Fister I (2018) Niapy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software* 3(23): 613.

Wedashwara W, Ahmadi C and Arimbawa I (2019) Sequential fuzzy association rule mining algorithm for plants environment classification using internet of things. In: *AIP Conference Proceedings*, volume 2199. AIP Publishing.

Yan X, Zhang C and Zhang S (2009) Confidence metrics for association rule mining. *Applied Artificial Intelligence* 23(8): 713–737.

Time Complexity Analysis of Aerial

This section provides a time complexity analysis of our Aerial approach, Algorithm 1, in big O notation. We analyze each line in the algorithm and aggregate the results at the end.

Line 2 initializes an empty rule set $\mathcal{R}$ in O(1).

Line 3 is a combination operation over the input features, *X.features*, taken antecedent a at a time, and creates a set of candidate antecedents $\mathcal{C}$. Let's assume F is the features ($X.features$) for short, and a is the maximum number of antecedents parameter, then the complexity is $O(\binom{|F|}{a})$.

Line 4 iterates over $\mathcal{C}$. Therefore, the operations inside this outer loop are repeated $\binom{|F|}{a}$ times.

Line 5 initiates a vector with equal probabilities per feature class value. It is linear over the feature class count. $\forall f \in F$, $F_c = \sum_{i=1}^{|F|} |f_i^c|$ where F_c is the total number of classes across all features F, $O(F_c)$.

Line 6 creates a set of vectors in which class values of the features $A \in \mathcal{C}$ are marked with 1. In the worst-case scenario, this step is linear over feature classes when the $\mathcal{C}$ is equal to all of the features in the input dataset, hence, $O(F_c)$.

Line 7 iterates $|V|$ times over the generated vectors in the previous line.

Line 8 performs a forward pass with the given test vector. Since each forward pass performs $softmax$ operations over the class values of features, this operation is linear over the number of feature classes, $O(F_c)$, assuming that softmax is performed in $O(1)$.

Lines 9 and 10 perform a comparison operation to check whether probabilities inside the $\hat{y}$ array that corresponds to the marked features A are higher than a threshold or not. Assuming the worst-case scenario, this operation is repeated for each feature class in the input data, $O(F_c)$.

Lines 11-13 iterate over the features F that are not among the marked features A (line 11), and check whether the probability of each feature class is higher than the threshold (line 12). High-probability features are then stored in line 13 as rules together with A. Since the loop is repeated at most F_c for each feature class, it is linear over the number of feature classes, $O(F_c)$.

Aggregation of the results:

1. The outer loop runs $\binom{|F|}{a}$ times.
2. For each iteration of the outer loop, lines 5 and 6 create an initial vector with equal probabilities and mark some of the feature classes in $O(F_c)$ time.
3. The middle loop (line 7) runs over the V test vectors. A forward pass and the probability check in lines 8-10 are performed in $O(F_c)$ time.
4. The inner-most loop (line 11) runs in $O(F_c)$ time.

Therefore, the complexity is $(O(\binom{|F|}{a}) \times O(F_c)) + (|V| \times 2 \times O(F_c))$. Assuming that $|V|$ is linear over the number of features $|F|$ (as the number of test vectors per antecedent is bounded due to limited class combinations, and the number of antecedents scales with $|F|$), and that the algorithm performs operations linear in the total number of feature classes F_c, which is assumed to scale linearly with $|F|$ (as each feature typically has a small, fixed number of class values in practice), the first part of the equation becomes the dominant term. Thus, the time complexity in the worst case is $(O(\binom{|F|}{a}) \times O(F_c))$.

Rewriting $O(\binom{|F|}{a})$ as $O(|F|^a)$, and assuming that the feature class count is linear over the number of features $|F|$ (again as each feature typically has a small, fixed number of class values in practice), and the number of antecedents a is a constant (usually less than 10, and between 2-5 in most practical applications) the complexity of Algorithm 1 is $O(|F|^{a+1})$.

Additional Experiments

This section contains auxiliary experiments that were not included in the core part of the paper. The experimental setups described in Section Experimental Settings are followed for these additional experiments as well unless otherwise specified.

Experiment 4: Effect of sensor count on rule generalizability. This experiment follows Experimental Setting 1 and investigates whether the effect of semantic enrichment of the sensor data is dependent on the number of sensors in terms of the generalizability of the rules learned. Note that we define the generalizability of rules as having high support and high coverage over the data. This is an extension of the Experiments in Section Semantics vs without Semantics.

Table 10. Comparison of ARM on sensor data with semantics (indicated as w-s) and without semantics (wo-s) for 10, 15, and 20 sensors (Cov=Coverage, Conf=Confidence, FP = FP-Growth, AE = Aerial).

	# Rules	Support	Cov.	Conf.
	w-s|wo-s	w-s|wo-s	w-s|wo-s	w-s|wo-s
		LeakDB		
FP(10)	43K|472	**0.23**|0.22	**0.24**|0.23	0.96|0.96
FP(15)	130K|7321	**0.27**|0.14	**0.28**|0.15	0.96|0.97
FP(20)	103K|8974	**0.41**|0.19	**0.43**|0.2	0.95|0.97
AE(10)	123|109	**0.31**|0.24	**0.33**|0.27	0.94|0.91
AE(15)	547|940	**0.31**|0.27	**0.37**|0.31	0.88|0.89
AE(20)	554|2521	**0.54**|0.25	**0.59**|0.3	0.91|0.87
		L-Town		
FP(10)	11489|578	**0.58**|0.35	**0.62**|0.37	0.94|0.95
FP(15)	19447|2055	**0.76**|0.33	**0.8**|0.35	0.95|0.95
FP(20)	25421|5047	**0.86**|0.36	**0.9**|0.38	0.96|0.96
AE(10)	72|381	**0.61**|0.34	**0.67**|0.39	0.92|0.88
AE(15)	264|1300	**0.54**|0.35	**0.6**|0.42	0.9|0.87
AE(20)	1005|2551	**0.59**|0.39	**0.65**|0.45	0.91|0.88
		LBNL		
FP(10)	25|764	**0.94**|0.24	**0.94**|0.24	1|0.99
FP(15)	280|181	**0.75**|0.35	**0.75**|0.35	1|0.99
FP(20)	7220|2883	**0.84**|0.73	**0.85**|0.75	0.98|0.99
AE(10)	422|14	**0.73**|0.28	**0.73**|0.29	1|0.97
AE(15)	832|61	**0.78**|0.42	**0.78**|0.43	1|0.99
AE(20)	73|258	**0.74**|0.65	**0.74**|0.66	1|0.99

Table 10 shows the average rule count, support, rule coverage, and confidence of the rules mined by FP-Growth and our Aerial algorithms with varying numbers of sensors (10, 15, and 20) with (w-s) and without (wo-s) the semantic enrichment. On all 3 datasets, regardless of the number of sensors used, the average support and coverage of the rules increased upon semantic enrichment of the sensor data.

Table 11. TS-NARM needs high numbers of evaluations (Evals.) for good performance (L-Town, Conf = Confidence).

Evals.	Algorithm	# Rules	Time(s)	Conf.
1000	DE	55.5	5.82	0.39
	GA	46	5.53	0.26
	PSO	67	6.52	0.4
	LSHADE	76	7.08	0.36
	jDE	76	3.81	0.5
10000	DE	2595	158.4	0.68
	GA	270	127.64	0.38
	PSO	2215.5	148.22	0.58
	LSHADE	2369	131.73	0.75
	jDE	2038.5	110.89	0.75
30000	DE	7686.5	610.8	0.75
	GA	823.5	576.75	0.36
	PSO	11026.5	572.44	0.82
	LSHADE	12071.5	616.04	0.94
	jDE	6297.5	403.73	0.76
50000	DE	31673.6	778.46	0.81
	GA	11239	591.89	0.51
	PSO	30570.2	828.5	0.75
	LSHADE	35559.4	871.7	**0.98**
	jDE	24245	433.69	0.78

This is consistent with the results presented in Semantics vs without Semantics section. In addition, the results show that increasing the number of sensors leads to even higher support and rule coverage values on average. The FP-Growth algorithm mined significantly more rules upon semantic enrichment across all datasets, while the number of learned rules decreased for our Aerial approach after semantic enrichment. We argue that due to the static semantic properties in the knowledge graph, the FP-Growth generates a high number of association rules in between those static properties, while this is not the case for Aerial. Lastly, the confidence values did not change significantly.

Experiment 5: Effect of maximum evaluations on the execution time and rule quality of optimization-based ARM. This section contains the experiments for evaluating the effect of maximum evaluation parameters of the optimization-based methods (TS-NARM) on execution time and rule quality. The experiment results for the LeakDB dataset are already given in Experiment 2.1. Therefore, this section only contains the results for the L-Town and the LBNL datasets and the results are consistent across all datasets.

Tables 11 and 12 show the results for the L-Town and LBNL datasets respectively. Similar to the results for the LeakDB dataset, as the maximum number of evaluation parameters increases, the number of rules, execution time as well as average confidence of the rules increase. The increment in the confidence values decreases as the maximum evaluations increase. These experiments show that optimization-based methods require longer execution times in order to obtain higher-quality rules.

Experiment 6: Extracting Association Rules with ARM-AE. ARM-AE (Berteloot et al. 2023), an Autoencoder-based ARM approach, is different than Aerial

Table 12. TS-NARM needs high numbers of evaluations (Evals.) for good performance (LBNL, Conf = Confidence).

Evals.	Algorithm	# Rules	Time(s)	Conf.
1000	DE	90	6.91	0.49
	GA	58.5	5.26	0.51
	PSO	104.5	6.93	0.48
	LSHADE	132.5	6.15	0.47
	jDE	172.5	4.83	0.65
10000	DE	3037.5	115.96	0.72
	GA	370.5	107.77	0.51
	PSO	2825.5	108.82	0.78
	LSHADE	3372.5	95.99	0.82
	jDE	2919	51.2	0.73
30000	DE	9751	170.16	0.74
	GA	419	185.84	0.5
	PSO	8933	188.72	0.96
	LSHADE	17624	182.64	**0.97**
	jDE	7958	111.05	0.77
50000	DE	27778.8	479.6	0.77
	GA	7945.4	501.37	0.47
	PSO	25453.8	530.74	0.79
	LSHADE	26864.4	787.79	**0.97**
	jDE	20243.2	421.82	0.77

both in terms of autoencoder architecture and rule extraction methodology. Its autoencoder is not under-complete but has equal dimensions in each layer, has a different loss function (MSE), and the loss function is not applied per feature, but for the entire output. In the rule extraction stage, it does not assign equal probabilities to the unmarked features, but leaves them as 0, and assumes that the output is the antecedent while the input is the consequent. We argue that due to these differences, ARM-AE resulted in low rule quality with a higher number of rules, both in their paper (33% confidence on the Nursery dataset and 50% confidence on the chess dataset) and also based on our results presented in this section. Therefore, we opted not to include it in the core evaluation section.

In the original paper, ARM-AE is tested on categorical tabular data only, and to the best of our knowledge, ARM-AE is the only fully DL-based ARM approach, besides our approach, at the time of writing this paper. Note that there are DL-based approaches to sequential ARM (He et al. 2024), however, that is a different task than the one we tackle in this paper. We adapted ARM-AE to work with sensor data as part of our pipeline and used it as a baseline for Experimental Setting 2. It expects a number of antecedents, a number of rules per consequent, and a likeness (similarity threshold) parameter. The number of antecedents is set to 2, number of rules per consequent is set to the number of rules learned by our Aerial approach divided by the number of features (of the dataset subject to evaluation), and the likeness is set to 80%, similar to our approach for fairness.

The evaluation results, given in Table 13, show that ARM-AE resulted in exceptionally low rule quality values on all 3 datasets. Therefore, the results were not included in the core part of the paper, however, for the purpose of having complete novel baselines, we included them in this section.

Table 13. Evaluation of ARM-AE on all 3 datasets for experimental setting 2 (Conf = Confidence, Cov = Coverage).

Dataset	Rule Count	Support	Conf.	Data Cov.	Zhang's Metric
LeakDB	4400	0.08	0.13	0.05	-0.79
L-Town	5600	0.08	0.11	0.1	-0.89
LBNL	3440	0.36	0.46	0.08	-0.41