Data & Knowledge Engineering 118 (2018) 92–106

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Mining range associations for classification and characterization

Jianhua Shao [*], Achilleas Tziatzios

School of Computer Science & Informatics, Cardiff University, CF24 3AA, UK

ARTICLE INFO

Keywords:
Classification
Characterization
Numerical ranges
Classification association rule mining

ABSTRACT

In this paper, we propose a method that is able to derive rules involving range associations from numerical attributes, and to use such rules to build comprehensible classification and characterization (data summary) models. Our approach follows the classification association rule mining paradigm, where rules are generated in a way similar to association rule mining, but search is guided by rule consequents. This allows many credible rules, not just some dominant rules, to be mined from the data to build models. In so doing, we propose several sub-range analysis and rule formation heuristics to deal with numerical attributes. Our experiments show that our method is able to derive range-based rules that offer both accurate classification and comprehensible characterization for numerical data.

1. Introduction

In many practical applications, it is desirable that we are able to extract the following type of rule from numerical data:

$$age \in [25, 30] \wedge loan \in [2000, 3000] \Rightarrow repay = yes$$

That is, we derive rules that contain *ranges* in their antecedents and a categorical value as a consequent. These rules capture knowledge from numerical data naturally and allow comprehensible classification and characterization (data summary) models to be built. For example, in the process industry, performance data is often analyzed to help determine how engineering processes may be optimized. Such data typically contains a large number of numerical attributes and it is useful that we are able to extract range-based rules to describe the relationships among various variables, so that causality can be understood naturally and processes tuned accordingly.

Extracting ranges from numerical attributes has been considered in classification rule mining. Existing methods typically follow a "cover and remove" strategy [1]. That is, a rule is heuristically formed to cover a subset of the data as well as possible, and this subset is then removed from the data. This is repeated on the remaining data until all the data is covered this way. This strategy works well with categorical data, but is not effective when dealing with numerical attributes, because there is a potentially very large number of ways to form ranges and to cover the data. Consequently, these methods resort to discretization or point-based split. However, these mechanisms may not capture some relevant ranges and do not help understand discovered rules [2].

The extraction of ranges from numerical data has also been considered in association rule mining. For example, Srikant and Agrawal proposed a method that partitions numerical data into initial ranges first and then allows neighboring ranges to be combined [3]; Fukuda et al. developed an efficient method that allows rectangular regions to be found from two dimensional numerical data directly [4]; Autmann and Lindell suggested a statistical model which focuses on discovering ranges that are statistically significant [2]; and Salleb-Aouissi et al. used genetic algorithms to derive ranges heuristically from numerical attributes [5]. All these methods aim to derive ranges with some kind of optimality, but they are either limited to dealing with no more than two numerical attributes or they do not attempt to find range-based rules that can support classification as well as characterization.

* Corresponding author.
E-mail address: ShaoJ@cardiff.ac.uk (J. Shao).

https://doi.org/10.1016/j.datak.2018.10.001
Received 12 September 2017; Received in revised form 5 October 2018; Accepted 10 October 2018
Available online 24 October 2018

In this paper, we propose a method that is able to derive range-based rules from multiple numerical attributes, and to use such rules to build accurate classification and characterization models. Our approach is inspired by the classification association rule mining (CARM) methodology [6], where instead of searching for all large itemsets and then generating rules from them, as performed in conventional association rule mining, search is guided by rule consequents and only those itemsets that are relevant to the given consequents will be generated. This, in contrast to the conventional "cover and remove" methodology for classification rule mining, allows many credible rules, rather than only some dominant rules, to be generated. As such, it enables an ensemble type of classifier to be built or detailed data characterization to be obtained. This is especially useful for applications where data may support multiple hypotheses. We adapt this approach to discovering range-based rules from numerical data. That is, we use rule consequents to guide the formation of ranges (e.g. $age \in [25, 30]$) and to create associated ranges (e.g. $age \in [25, 30] \wedge loan \in [2000, 3000]$). More specifically, we make the following contributions:

- We attempt to derive rules involving range associations from numerical attributes, and to use such rules to build comprehensible classification and characterization models. In so doing, we adapt CARM to mining associated ranges from numerical data. This is significantly more challenging than mining large itemsets from categorical data as there are many ways to form ranges from a numerical attribute. We introduce *consequent bounded ranges* which are an association of ranges whose boundaries are determined by rule consequents. We also propose a top-down split strategy to partition a large range into sub-ranges. These strategies allow many credible, not just some dominant rules to be discovered from numerical data efficiently.
- We propose heuristics for splitting ranges and for rule formation. Given a range, there can be many ways to split it into sub-ranges. Our heuristics are greedy in nature and split ranges in such a way that the support, confidence or a tradeoff between the two is optimized in the resulting sub-ranges. These heuristics allow rules with specific qualities to be extracted from data, and as such different application needs can be catered for by our method.
- We introduce a new measure, *density*, for assessing the quality of extracted rules. This is in addition to the standard support and confidence measures that are commonly used in mining association rules, and is used to measure the concentration of an extracted rule. This is necessary as unlike dealing with categorical itemsets, associated ranges can contain "dangling" cases (i.e. tuples contained in one range but not another of the same rule). Such dangling cases can dilute the quality of a range-based rule, and the density measure is designed to assess this.
- We evaluate extracted rules in both classification and characterization settings, rather than just their classification quality as the majority of exiting classification rule extraction works consider. Note that while criteria for evaluating rules for classification are well established, how to assess rules for their data characterization quality is less clear. We introduce measures to study the characterization power of extracted rules.

To the best of our knowledge our proposed method is the only one that is able to derive associated ranges from any number of numerical attributes through a top-down range split, and to build an ensemble type of range-based rule set for classification as well as characterization. Our experiments show that our method is able to derive range-based rules that can offer not only good classification results, but also good characterization for numerical data.

The rest of the paper is organized as follows. In Section 2, we give the necessary definitions that we use in the paper. In Section 3, we introduce our CARM-based method. The top-down range partitioning and a number of heuristics for rule formation are given in Section 4. Section 5 discusses experimental results and related work is analyzed in Section 6. We conclude the paper in Section 7.

2. Preliminaries

Without loss of generality, we assume that data is contained within a single table $T(A_1, A_2, \ldots, A_m, C)$, where each A_j, $1 \le j \le m$, is a numerical attribute and C is a categorical class attribute. We denote the kth tuple of T by $t_k = \langle v_{k,1}, v_{k,2}, \ldots, v_{k,m}, c_k \rangle$, where $v_{k,j} \in A_j$, $1 \le j \le m$. We may drop c_k from t_k when it is not needed in the discussion. Also, for conciseness of discussion in this paper, we consider numerical attributes only here. But our proposed method can be extended to work with a table that consists of both numerical and categorical attributes.

Definition 1 (*Range*)**.** Let a and b be two values in the domain of attribute A such that $a \le b$. A *range* over A, denoted by $[a, b]_A$, is a set of values in A that fall between a and b.

Definition 2 (*Cover*)**.** Let $r = [a, b]_{A_j}$ be a range over attribute A_j. r is said to *cover* tuple $t_k = \langle v_{k,1}, v_{k,2}, \ldots, v_{k,m} \rangle$ if $a \le v_{k,j} \le b$. We denote the set of tuples covered by r by $\tau(r)$.

Definition 3 (*Associated Ranges*)**.** Let $r_1 = [a_1, b_1]_{A_1}, r_2 = [a_2, b_2]_{A_2}, \ldots, r_h = [a_h, b_h]_{A_h}$ be a set of ranges over attributes $A_1, A_2, \ldots, A_h$ respectively. $r_1, r_2, \ldots, r_h$ are *associated ranges* if $\tau(r_1) \cap \tau(r_2) \cap \cdots \cap \tau(r_h) \neq \emptyset$.

Definition 4 (*Range-based Classification Rule*)**.** Let c be a class value and $r_1, r_2, \ldots, r_h$ be a set of associated ranges. $r_1, r_2, \ldots, r_h \Rightarrow c$ is a *range-based classification rule* or *range-based rule* for short.

Many range-based rules can be formed from a given table T. For example, each tuple of T can be such a rule. Clearly, such rules will be too specific to be useful. To find rules with some desired quality, we introduce three measures below.

Table 1
An example table where TID shows the tuple identifiers and C is the class attribute.

TID	A_1	A_2	A_3	C
t_1	0.23	0.20	0.21	c_1
t_2	0.17	0.13	0.10	c_1
t_3	0.82	0.52	0.05	c_2
t_4	0.11	0.10	0.33	c_1
t_5	0.05	0.44	0.52	c_2
t_6	0.49	0.06	0.06	c_2
t_7	0.57	0.47	0.15	c_2
t_8	0.27	0.09	0.13	c_1
t_9	0.34	0.19	0.45	c_1
t_{10}	0.39	0.32	0.72	c_2

Definition 5 (*Support*)**.** Let T be a table and $\lambda : r_1, r_2, \ldots, r_h \Rightarrow c$ be a range-based rule derived from T. The *support* of λ in T is

$$\sigma(\lambda) = \frac{|\tau(r_1) \cap \tau(r_2) \cap \cdots \cap \tau(r_h)|}{|T|}$$

where $|\cdot|$ denotes the size of a set.

Definition 6 (*Confidence*)**.** Let T be a table and $\lambda : r_1, r_2, \ldots, r_h \Rightarrow c$ be a range-based rule derived from T. The *confidence* of λ in T is

$$\delta(\lambda) = \frac{|\tau(r_1) \cap \tau(r_2) \cap \cdots \cap \tau(r_h) \cap \tau(c)|}{|\tau(r_1) \cap \tau(r_2) \cap \cdots \cap \tau(r_h)|}$$

where $\tau(c)$ denotes the set of tuples that have class value c in T.

Definition 7 (*Density*)**.** Let T be a table and $\lambda : r_1, r_2, \ldots, r_h \Rightarrow c$ be a range-based rule derived from T. The *density* of λ in T is

$$\gamma(\lambda) = \frac{|\tau(r_1) \cap \tau(r_2) \cap \cdots \cap \tau(r_h)|}{|\tau(r_1) \cup \tau(r_2) \cup \cdots \cup \tau(r_h)|}$$

The support and confidence measures follow those used in [7] for numerical association rule mining: support for λ indicates its *strength* (i.e. how many cases in T are included in the associated ranges) and confidence indicates its *credibility* (i.e. how often the rule is actually valid given the associated ranges). Density is a new measure we introduce to assess a rule's *concentration* (i.e. how many dangling tuples covered by the rule in T). The following example explains these definitions.

Example 1. Suppose that we have the data in Table 1 and we have a rule $\lambda : [0.11, 0.39]_{A_1} \wedge [0.13, 0.32]_{A_2} \Rightarrow c_1$. Then we have

$$\begin{aligned}
\sigma(\lambda) &= \frac{|\tau([0.11, 0.39]_{A_1}) \cap \tau([0.13, 0.32]_{A_2})|}{|T|} \\
&= \frac{|\{t_1, t_2, t_9, t_{10}\}|}{10} = 4/10 \\
\delta(\lambda) &= \frac{|\tau([0.11, 0.39]_{A_1}) \cap \tau([0.13, 0.32]_{A_2}) \cap \tau(c_1)|}{|\tau([0.11, 0.39]_{A_1}) \cap \tau([0.13, 0.32]_{A_2})|} \\
&= \frac{|\{t_1, t_2, t_9\}|}{|\{t_1, t_2, t_9, t_{10}\}|} = 3/4 \\
\gamma(\lambda) &= \frac{|\tau([0.11, 0.39]_{A_1}) \cap \tau([0.13, 0.32]_{A_2})|}{|\tau([0.11, 0.39]_{A_1}) \cup \tau([0.13, 0.32]_{A_2})|} \\
&= \frac{|\{t_1, t_2, t_9, t_{10}\}|}{|\{t_1, t_2, t_4, t_8, t_9, t_{10}\}|} = 2/3
\end{aligned}$$

Note that in Table 1, t_8 has a value of 0.27 in A_1 which is covered by $[0.11, 0.39]_{A_1}$, and a value 0.09 in A_2 which is not covered by $[0.13, 0.32]_{A_2}$. So t_8 is partially covered by λ given in Example 1, and therefore is a dangling tuple. Dangling tuples do not affect the support or confidence of a rule, but intuitively the fewer dangling tuples are covered by a rule, the better the rule is as we can be more certain about its range association. We introduce density to measure this.

One class of rules of interest is those whose support, confidence and density are above a minimum threshold.

Definition 8 (*Min-$\sigma\gamma\delta$ Rule*)**.** A range-based rule λ is a *min-$\sigma\gamma\delta$ rule* if it satisfies the following properties, where σ_{min}, γ_{min} and δ_{min} are user specified thresholds:

- $\sigma(\lambda) \geq \sigma_{min}$,
- $\gamma(\lambda) \geq \gamma_{min}$, and
- $\delta(\lambda) \geq \delta_{min}$.

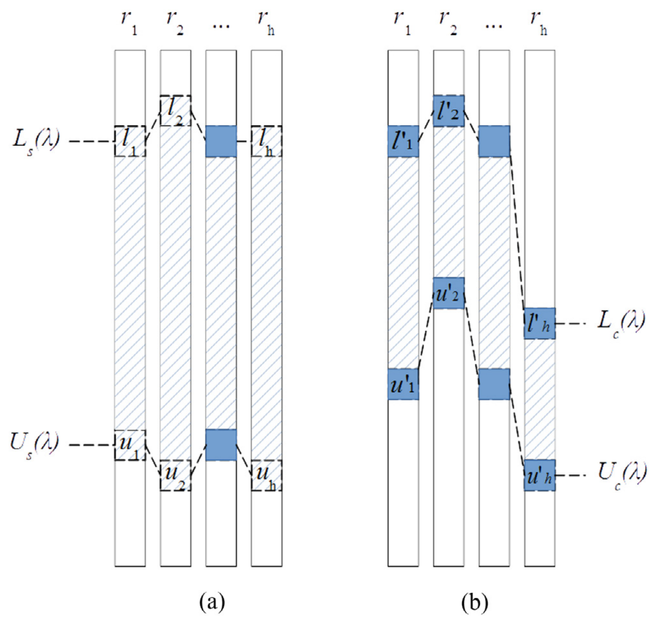

Fig. 1. For an arbitrary rule $\lambda : r_1, r_2, \dots, r_h \Rightarrow c$, (a) shows the removal of dangling tuples from the two ends of a range, resulting in narrowed boundaries ($[l_i, u_i]$) for each range; and (b) shows the further removal of tuples at the two ends of a range that do not have c as a consequent, resulting further contracted boundaries ($[l'_i, u'_i]$) for each range.

A brute-force solution to find all min-$\sigma\gamma\delta$ rules from a given table is to examine all possible combinations of ranges across all attributes. This is computationally infeasible. In the following sections, we describe our methods for finding such rules heuristically.

3. Deriving associated ranges

To derive range-based rules, we first introduce the concept of *consequent bounded ranges*, and then discuss how they may be obtained from data using an apriori type of association formation.

3.1. Consequent bounded ranges

Given an arbitrary range-based rule $\lambda : r_1, r_2, \dots, r_h \Rightarrow c$, we observe that the boundaries of its ranges may be revised (i.e. some ranges may be narrowed) without affecting its support or confidence. This is illustrated in Fig. 1, where each rectangular box represents the set of tuples covered by the corresponding range, and their values are sorted in ascending order.

Consider the first tuple covered by r_1 (i.e. the tuple has the smallest value in r_1), for example. If it is a dangling tuple (i.e. it is not covered by at least one range in $r_2, \dots, r_h$), we may move the boundary of r_1 downwards. Likewise, we can apply this boundary revision to the other end of the range and to all the ranges. As a result, we obtain two sets of new boundaries, $L_s(\lambda)$ and $U_s(\lambda)$ as shown in Fig. 1(a). $L_s(\lambda) = \{l_1, l_2, \dots, l_h\}$ contains the lowest value in each range whose corresponding tuples support the rule, and $U_s(\lambda)$ contains the highest such values. Note that $l_1, l_2, \dots, l_h$ are not necessarily from the same tuple. These two sets of values effectively form "support boundaries" within which λ is supported. Clearly, this revision of boundaries will only eliminate some dangling tuples, hence it will not affect the rule's support or confidence, but can increase its density. So this revision is always desirable.

We further observe that the tuples corresponding to the boundary values in $L_s(\lambda)$ and $U_s(\lambda)$ may not have c as a consequent. However, it makes sense that rules should start and end with a tuple whose consequent is c [8], so we can move boundaries further inwards towards the first (non-dangling) tuple that has c as a consequent. This will result in "c-boundaries" as shown in Fig. 1(b).

Definition 9 (*c-boundaries*). Let $\lambda : r_1, r_2, \dots, r_h \Rightarrow c$ be a range-based rule. Its lower and upper *consequent* boundaries (*c-boundaries*) are given by $L_c(\lambda)$ and $U_c(\lambda)$

- $L_c(\lambda) = \{a_1, a_2, \dots, a_h \mid a_i = \min_{\forall t \in \mathcal{S} \wedge C(t)=c} \phi(t, r_i)\}$
- $U_c(\lambda) = \{b_1, b_2, \dots, b_h \mid b_i = \max_{\forall t \in \mathcal{S} \wedge C(t)=c} \phi(t, r_i)\}$

where $\mathcal{S} = \tau(r_1) \cap \tau(r_2) \cap \dots \cap \tau(r_h)$, $C(t)$ is a function that returns the consequent of t, $\phi(t, r_i)$ is a function that returns the value of t in r_i, and $i = 1 \dots h$.

Note that a rule formed by ranges with c-boundaries is not necessarily "better than" those formed by ranges with support boundaries, since by deriving c-boundaries, support is sacrificed for confidence. However, c-boundaries are intuitively preferred,

Table 2
Table 1 in Column Store Format. Each column is represented by a set of triples $\langle tid, val, c\rangle$, where val records the values of A_i sorted in ascending order, tid records their corresponding tuple identifiers, and c records their consequents.

A_1			A_2			A_3		
tid	val	c	tid	val	c	tid	val	c
t_5	0.05	c_2	t_6	0.06	c_2	t_3	0.05	c_2
t_4	0.11	c_1	t_8	0.09	c_1	t_6	0.06	c_2
t_2	0.17	c_1	t_4	0.10	c_1	t_2	0.10	c_1
t_1	0.23	c_1	t_2	0.13	c_1	t_8	0.13	c_1
t_8	0.27	c_1	t_9	0.19	c_1	t_7	0.15	c_2
t_9	0.34	c_1	t_1	0.20	c_1	t_1	0.21	c_1
t_{10}	0.39	c_2	t_{10}	0.32	c_2	t_4	0.33	c_1
t_6	0.49	c_2	t_5	0.44	c_2	t_9	0.45	c_1
t_7	0.57	c_2	t_7	0.47	c_2	t_5	0.52	c_2
t_3	0.82	c_2	t_3	0.52	c_2	t_{10}	0.72	c_2

as the amount of support that is lost in the process is associated with the tuples that do not support c. We aim to find a set of rules whose ranges are given by c-boundaries, or c-bounded rules.

Example 2. Consider the rule given in Example 1 again:

$$\lambda : [0.11, 0.39]_{A_1} \wedge [0.13, 0.32]_{A_2} \Rightarrow c_1$$

Its support boundaries are $L_s(\lambda) = \{0.17, 0.13\}$ and $U_s(\lambda) = \{0.39, 0.32\}$, respectively. But for the boundary tuple t_{10}, its consequent is c_2. Therefore, we revise the corresponding boundaries, and obtain the following c-bounded rule (assuming that it also meets the min-$\sigma\gamma\delta$ condition):

$$\lambda' : [0.17, 0.34]_{A_1} \wedge [0.13, 0.20]_{A_2} \Rightarrow c_1$$

3.2. Finding c-bounded rules

Our approach to finding c-bounded rules follows the CARM methodology [6] and is shown in Algorithm 1. For convenience, an association of i ranges will be referred to as an i-range in the following discussion. For example, $[0.11, 0.39]_{A_1} \wedge [0.13, 0.32]_{A_2}$ is referred to as a 2-range.

Algorithm 1 *Finding c-bounded rules*

```
input: $T(A_1, A_2, \ldots, A_m, C)$, $\sigma_{min}$, $\gamma_{min}$ and $\delta_{min}$
output: $\mathcal{R}$

1.  $\mathcal{R} \leftarrow \emptyset$;
2.  for each $c_j$ in $C$ do
3.      $LR_1 \leftarrow \{[v_{a_1}, v_{b_1}]_{A_1}, \ldots, [v_{a_m}, v_{b_m}]_{A_m}\}$
4.      for ($i = 1,\ i \le m,\ i{+}{+}$) do
5.          for each $r_1, r_2, \ldots, r_i$ in $LR_i$ do
6.              $R \leftarrow$ ANALYZE($r_1, r_2, \ldots, r_i \Rightarrow c_j, \sigma_{min}, \gamma_{min}, \delta_{min}$)
7.              $\mathcal{R} \leftarrow \mathcal{R} \cup R$
8.          $LR_{i+1} \leftarrow \emptyset$
9.          for each $ar \in LR_i$ do
10              for each $r \in LR_1 \wedge r \notin ar$ do
11.                 $LR_{i+1} \leftarrow LR_{i+1} \cup (ar, r)$
12. return $\mathcal{R}$
```

Algorithm 1 works as follows. Each distinct consequent c_j is considered in turn (step 2), and a set of largest 1-ranges (LR_1) is obtained for it according to Definition 9 (step 3). For efficiency, we store T as a set of columns, for example, Table 2 shows Table 1 stored as a set of columns. This enables the set of largest 1-ranges to be obtained by simply removing tuples whose consequents are not c_j at the two ends of the columns. For example, for c_1 we have $LR_1 = \{[0.11, 0.34]_{A_1}, [0.09, 0.20]_{A_2}, [0.10, 0.45]_{A_3}\}$ for the data in Table 2.

The algorithm then goes into iteration. Each i-range in LR_i is analyzed to generate rules (steps 4–7). Note that when mining association rules from categorical items, rules that have enough support but not sufficient confidence are simply discarded. In mining range-based rules, however, we may decrease the size of a range to increase a rule's confidence or density. Thus, when a rule does not have sufficient confidence or density, we can consider replacing some of its ranges with their sub-ranges. Furthermore, even

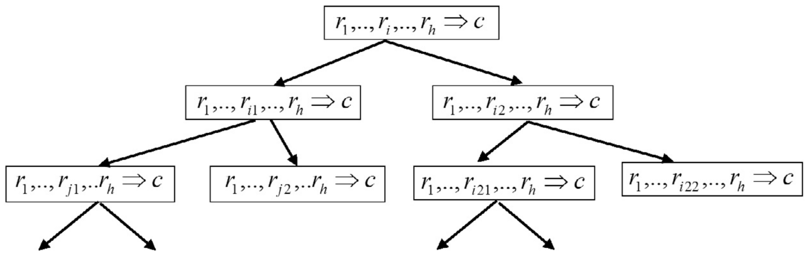
$r_1,..,r_i,..,r_h \Rightarrow c$
$r_1,..,r_{i1},..,r_h \Rightarrow c$
$r_1,..,r_{i2},..,r_h \Rightarrow c$
$r_1,..,r_{j1},..r_h \Rightarrow c$
$r_1,..,r_{j2},..r_h \Rightarrow c$
$r_1,..,r_{i21},..,r_h \Rightarrow c$
$r_1,..,r_{i22},..,r_h \Rightarrow c$

Fig. 2. Split Tree. Starting with a given rule (root), one range is selected heuristically to split, for example, r_i into r_{i1} and r_{i2}. This process is then repeated until no ranges may be split. The result is a binary split tree shown here. Each node represents a rule involving different ranges. Each has sufficient support, but may or may not have sufficient confidence or density, thus it may or may not be a valid rule.

when a rule has enough confidence and density, it is still worth considering forming additional rules using their sub-ranges, since these rules may offer better predictive power and data characterization. The Analyze function performs this analysis (its pseudo-code will be given in Section 4): for each i-range in LR_i and the consequent c_j, it derives relevant sub-ranges from it and uses them to form range-based rules heuristically. We will discuss different heuristics for sub-range derivation and rule formation in the following sections. A set of $(i+1)$-ranges is then generated by appending each large i-range with a largest 1-range (steps 8–11). Finally, the set of derived rules is returned (step 12).

4. Range analysis

A range can contain many sub-ranges, especially when overlapping sub-ranges are considered. For example, given

$$[0.05, 0.57]_{A_1} \wedge [0.09, 0.20]_{A_2} \Rightarrow c_1$$

we can derive $[0.11, 0.39]_{A_1}$ and $[0.17, 0.49]_{A_1}$ from $[0.05, 0.57]_{A_1}$ to form

$$[0.11, 0.39]_{A_1} \wedge [0.09, 0.20]_{A_2} \Rightarrow c_1$$

$$[0.17, 0.49]_{A_1} \wedge [0.09, 0.20]_{A_2} \Rightarrow c_1$$

This can create a large number of range associations.

To avoid generating excessive sub-ranges for association computation, we propose to iteratively partition a range, rather than considering all possible sub-ranges. That is, given a range-based rule $\lambda : r_1, r_2, \ldots, r_h \Rightarrow c$, we heuristically select one range, say r_i, from λ and split it into r_{i1} and r_{i2}, and use them to replace r_i to form two new rules: $\lambda_1 : r_1, r_2, \ldots, r_{i1}, \ldots, r_h \Rightarrow c$ and $\lambda_2 : r_1, r_2, \ldots, r_{i2}, \ldots, r_h \Rightarrow c$. Note that as r_i is partitioned into two sub-ranges, the resultant r_{i1} and r_{i2} will not overlap. This avoids generating excessive ranges. As support is a monotonic measure, we can repeat the process on λ_1 and λ_2 as long as they have sufficient support. Consequently, this process will create a *split tree*, as shown in Fig. 2.

To split a range, it intuitively makes sense that the splitting point in the range should be one that corresponds to a tuple whose consequent is not the same as the rule's consequence. Moreover, as all ranges in our rules are required to be c-bounded, it is easy to see that we need to remove an interval surrounding such a point when splitting a range. We call such an interval *non-consequent interval* (*NCI*).

Definition 10 (*Non-consequent Interval*)**.** Let c be a class value. If a range r covers a set of tuples that do not contain c as their class value and no super-range of r has this property, then r is said to be a *non-consequent interval* w.r.t. c.

In this paper we consider two such splitting heuristics: *Max-NCI* and *Min-NCI*. Given a rule $\lambda : r_1, r_2, \ldots, r_h \Rightarrow c$, *Max-NCI* scans all the ranges in λ, and chooses the largest *NCI* (an NCI that contains most tuples) to split the corresponding range, and *Min-NCI* chooses the smallest (containing least tuples). If there is a tie, one is selected at random. The following example illustrates these split heuristics.

Example 3. Suppose that we need to split rule $\lambda : [v_6, v_9]_{A_1} \wedge [u_4, u_7]_{A_2} \Rightarrow c$ whose ranges cover a set of data shown in Table 3. Note that both $[v_6, v_9]_{A_1}$ and $[u_4, u_7]_{A_2}$ are c-bounded ranges and their values are listed in Table 3 in ascending order.

To split the rule using *Max-NCI* given the data in Table 3, the Analyze function locates the interval covering $\{u_3, u_5, u_8, u_{15}\}$ in A_2 (shown in bold), and splits λ into two new rules:

$$\lambda_1 : [v_6, v_{16}]_{A_1} \wedge [u_4, u_{16}]_{A_2} \Rightarrow c \text{ and } \lambda_2 : [v_7, v_9]_{A_1} \wedge [u_{11}, u_7]_{A_2} \Rightarrow c$$

Note that the ranges in A_1 are revised following the split in order to satisfy the c-bounded property. For example, following the split, u_9 is no longer covered by $[u_4, u_{16}]_{A_2}$, hence v_9 in $[v_6, v_9]_{A_1}$ becomes dangling and is removed. All such dangling tuples are removed as part of our split process. Similarly, to split λ using *Min-NCI*, the Analyze function will locate v_1 in A_1 (shown in bold), and split λ into

$$\lambda'_1 : [v_6, v_{10}]_{A_1} \wedge [u_6, u_{10}]_{A_2} \Rightarrow c \text{ and } \lambda'_2 : [v_7, v_9]_{A_1} \wedge [u_4, u_7]_{A_2} \Rightarrow c$$

as a result.

Table 3
Data covered by $\lambda : [v_6, v_9]_{A_1} \wedge [u_4, u_7]_{A_2} \Rightarrow c$. For convenience of discussion, we refer to the values by their tuple IDs, e.g. v_6 represents the value that the 6th tuple in the original table has in A_1 and u_6 represents the value of the same tuple in A_2. Also, tuples having c as a consequent are represented by 1 and otherwise 0.

A_1		A_2	
val	*c*	*val*	*c*
v_6	1	u_4	1
v_{10}	1	u_{11}	1
v_1	**0**	u_6	1
v_{14}	1	u_{10}	1
v_7	1	u_{16}	1
v_{11}	1	u_3	**0**
v_2	1	u_5	**0**
v_{12}	1	u_8	**0**
v_5	0	u_{15}	**0**
v_{13}	0	u_{11}	1
v_3	0	u_9	1
v_4	1	u_7	1
v_{16}	1		
v_{15}	0		
v_8	0		
v_9	1		

We now describe how range split is done. Given a range-based rule $\lambda : r_1, r_2, \ldots, r_h \Rightarrow c$, a direct implementation of our range split function is to scan each range to find the largest or smallest *NCI* to split the rule. This, however, can lead to some redundant work. For example, when processing $A_1, A_2 \Rightarrow c$ and $A_1, A_3 \Rightarrow c$, A_1 will be scanned twice. To avoid repeated scan of the same range, we materialize all split intervals in all the attributes in a pre-processing step, and record them as an index:

NCI Index : $\{NCI_1, NCI_2, \ldots, NCI_p\}$

These *NCI*'s are ordered in their interval sizes. For instance, in Table 3 we obtain the following NCI index for A_1 and A_2:

$$\{[u_3, u_{15}]_{A_2}, [v_5, v_3]_{A_1}, [v_{15}, v_8]_{A_1}, [v_{10}, v_{10}]_{A_1}\}$$

To check how $A_1, A_2 \Rightarrow c$ should be split, only a scan of this index is needed. If *Max-NCI* is used to split the ranges, then this index is searched from left to right to find the first applicable *NCI*. Note that the splits following the current split (i.e. those splits at lower nodes of the tree) will never involve a larger *NCI* than the current split, so we do not need to repeatedly scan the index from the start to end, but only need to follow from the current position onwards as we carry on to split the ranges. *Min-NCI* is performed similarly except that we search the index from right to left.

Lemma 1. *Given a range-based rule $\lambda : r_1, r_2, \ldots, r_h \Rightarrow c$, the worst case cost of creating its split tree is $O(\frac{h}{\sigma_{min}})$, where σ_{min} is the minimum support requirement.*

Proof. Observe that the splitting process creates a binary tree and in the worse case it can have a maximum of $2^{\lceil \log_2 \frac{1}{\sigma_m} \rceil}$ nodes, as each sub-range must have a minimum support of $\frac{1}{\sigma_{min}}$ tuples. Thus, in the worst case $2^{\lceil \log_2 \frac{1}{\sigma_m} \rceil}$ splits will be performed on λ. We only need to scan through the NCI index once to complete all the splits on λ, and the size of the index in the worse case is $\frac{h}{\sigma_{min}}$. Hence the total cost is $O(\frac{h}{\sigma_{min}} + 2^{\lceil \log_2 \frac{1}{\sigma_m} \rceil}) = O(\frac{h}{\sigma_{min}})$.

5. Rule formation

Different strategies may be used to form rules by selecting certain sub-ranges from the split tree. In the following sections we describe four such heuristics for rule formation, each implementing the Analyze function in Algorithm 1.

5.1. Maximum confidence rule (MaxConf)

We first describe a strategy which returns a single rule from the splitting process that has the highest confidence. We call this strategy *maximum confidence rule* (MaxConf). This requires a simple tracking of the rule in the splitting process that has the required minimum density and the highest confidence. Note that all the rules in the split tree have sufficient support.

The MaxConf function works as follows. If a rule under consideration has the minimum support (step 4), then we perform two tasks. First, if it also has the minimum density and the highest confidence so far, then it is potentially returnable, so we retain it (steps 5–6). Second, we use the *Max-NCI* heuristic to split the rule (step 7). The Split function simply splits a rule into two with one range

Algorithm 2 *MaxConf*

```
input: $\lambda$, $\sigma_{min}$, $\gamma_{min}$ and $\delta_{min}$
output: A ruleset $R$

1.  $r_{max} \leftarrow \emptyset, Q \leftarrow Q.enqueue(\lambda)$
2.  while $Q \neq \emptyset$ do
3.      $q \leftarrow Q.dequeue()$
4.      if $\sigma(q) \geq \sigma_{min}$ then
5.          if $\gamma(q) \geq \gamma_{min} \wedge \delta(q) > \delta(r_{max})$ then
6.              $r_{max} \leftarrow q$
7.          $< \lambda_1, \lambda_2 > \leftarrow$ SPLIT($q$, Max-NCI)
8.          $Q \leftarrow Q.enqueue(\lambda_1, \lambda_2)$
9.  return $r_{max}$
```

replaced by two sub-ranges using the NCI index to search for the required splitting interval. The resultant two new rules λ_1 and λ_2 are put on the queue to be considered in the next iteration to see if they can be further split (step 8). This is repeated for the new rules created from the splitting process until the support requirement is no longer met.

5.2. Maximum gain rule (MaxGain)

The maximum confidence rule strategy can sacrifice substantial support for confidence. This may not be desirable as, for example, very confident rules may represent some known knowledge. In this section, we consider how we might strike a balance between support and confidence, and optimize both jointly. To achieve this, we use the *gain* measure [4].

Definition 11 (*Gain*)**.** Let T be a table and $\lambda : r_1, r_2, \ldots, r_h \Rightarrow c$ be a range-based rule derived from T. The *gain* for λ in T is

$$\xi(\lambda) = |\tau(r_1) \cap \tau(r_2) \cap \cdots \cap \tau(r_h) \cap \tau(c)| -$$

$$\delta_{min} \times |\tau(r_1) \cap \tau(r_2) \cap \cdots \cap \tau(r_h)|$$

Example 4. Suppose we have Table 1, $\lambda : [0.11, 0.34]_{A_1} \wedge [0.06, 0.20]_{A_2} \wedge [0.10, 0.33]_{A_3} \Rightarrow c_1$ and $\delta_{min} = 0.5$. Then we have

$$\begin{aligned}
\xi(\lambda) &= |\tau([0.11, 0.34]_{A_1}) \cap \tau([0.06, 0.20]_{A_2}) \cap \\
&\quad \tau([0.10, 0.33]_{A_3}) \cap \tau(c_1)| - \\
&\quad \delta_{min} \times |\tau([0.11, 0.34]_{A_1}) \cap \tau([0.06, 0.20]_{A_2}) \\
&\quad \cap \tau([0.10, 0.33]_{A_3})| \\
&= |\{t_1, t_2, t_4, t_8, t_9\} \cap \{t_1, t_2, t_4, t_6, t_8, t_9\} \cap \\
&\quad \{t_1, t_2, t_4, t_7, t_8\} \cap \{t_1, t_2, t_4, t_8, t_9\}| - \\
&\quad \delta_{min} \times |\{t_1, t_2, t_4, t_8, t_9\} \cap \{t_1, t_2, t_4, t_6, t_8, t_9\} \\
&\quad \cap \{t_1, t_2, t_4, t_7, t_8\}| \\
&= 4 - 0.5 \times 4 = 2
\end{aligned}$$

So gain is a measure that attempts to balance between support and confidence, and our MaxGain strategy is to return a single rule from the split tree that has the largest gain. The method is similar to Algorithm 2, except for two differences. First, in order not to bias towards confidence, *Min-NCI* is used to split a range. Second, the selection will be based on the maximum gain measure, not on confidence, hence minimum confidence requirement is also checked in order to guarantee that we return min-$\sigma\gamma\delta$ rules. The method is shown in Algorithm 3. Note that this algorithm follows from Algorithm 2 with only steps 4–7 changed, so for conciseness of presentation, we omitted the same steps from Algorithm 2, and only presented the steps that are different from Algorithm 2 here.

5.3. All supported rules (AllSupp)

The previous two strategies attempt to find one optimal rule to return, which either has the highest confidence (hence likely to be reliable) or has the largest gain (hence likely to be most characteristic) in the split tree. In this section, we consider returning multiple rules from a split tree. This is useful when the dataset contains multiple, perhaps even conflicting knowledge patterns, and returning just one rule may be too limited, especially for data characterization.

One obvious approach is to create the full tree and then return every internal node that has sufficient confidence and density. This is, however, not desirable as we may return some rules that are contained or subsumed by other rules. These rules can lead to a bias, for example, when a majority vote is used to classify data using the derived rules, as a "strong" rule may contain many sub-rules.

Algorithm 3 *MaxGain*

```
:   input, output and steps 1-3 are the same as Algorithm 2
4.      if $\sigma(q) \geq \sigma_{min}$ then
5.          if $\gamma(q) \geq \gamma_{min} \wedge \delta(q) \geq \delta_{min} \wedge \xi(q) > \xi(r_{max})$ then
6.              $r_{max} \leftarrow q$
7.          $< \lambda_1, \lambda_2 > \leftarrow$ Split($q$, Min-NCI)
:   steps 8-9 are the same as Algorithm 2
```

Definition 12 (*Subrule*). Let $\lambda_1 : r_1, r_2, \ldots, r_h \Rightarrow c$ and $\lambda_2 : r'_1, r'_2, \ldots, r'_k \Rightarrow c$ be two range-based rules derived from table T. We say that λ_1 is a *subrule* of λ_2, denoted by $\lambda_1 \prec \lambda_2$, if for each r_i in the antecedent of λ_1 there exists an r'_j in the antecedent of λ_2 such that r_i is a subrange of r'_j.

Clearly, each branch of the split tree will form containment relation from the root to the leaf. That is, each rule at a node is contained in the rule at its predecessor nodes. Our AllSupp heuristic is to return one rule per branch whose support is as high as possible, as long as they have minimum confidence and density. The method is described in Algorithm 4, which returns a set of rules from a split tree that do not have containment relationships.

Algorithm 4 *AllSupp*

input: λ, σ_{min}, γ_{min} and δ_{min}
output: A ruleset R

```
1.  $R \leftarrow \emptyset$; $Q \leftarrow Q.enqueue(\lambda)$
2.  while $Q \neq \emptyset$ do
3.      $q \leftarrow Q.dequeue()$
4.      if $\sigma(q) \geq \sigma_{min}$ then
5.          if $\gamma(q) \geq \gamma_{min} \wedge \delta(q) \geq \delta_{min}$ then
6.              $R \leftarrow R \cup q$
7.          else
8.              $< \lambda_1, \lambda_2 > \leftarrow$ Split($g$, Min-NCI)
9.              $Q \leftarrow Q.enqueue(\lambda_1, \lambda_2)$
10. return $R$
```

The methods is similar to Algorithm 2 with one main difference. The split of rule stops as soon as a rule is found with minimum support, confidence and density (step 4–6). We will only split a rule that has enough support but not enough confidence or density (step 8). This is because any sub-rule obtained from this point on (i.e. any rules at a lower node in the split tree) will be contained in this rule as sub-rules and these sub-rules cannot have a higher support than the current rule due to the monotonic property. Hence this rule is returned. In order to maximize support, we use *Min-NCI* to split a range, thereby retaining as much support as possible following the split.

5.4. All confident rules (AllConf)

Our final heuristic mirrors AllSupp, except that instead of using *Min-NCI*, we use *Max-NCI* to split a range, and instead of returning one rule from each branch that has the highest support, we return all rules that have sufficient confidence. The method is similar to Algorithm 4, and is shown in Algorithm 5. Again we only show the steps that are different from Algorithm 4 here. As confidence is not monotonic, we need to continue splitting the ranges until they no longer have enough support.

Algorithm 5 *AllConf*

```
:   input, output and steps 1-3 are the same as Algorithm 4
4.      if $\sigma(q) \geq \sigma_{min}$ then
5.          if $\gamma(q) \geq \gamma_{min} \wedge \delta(q) \geq \delta_{min}$ then
6.              if $q$ not redundant in $R$ w.r.t. Def 13 then
7.                  $R \leftarrow R \cup q$
8.      $< \lambda_1, \lambda_2 > \leftarrow$ Split($q$, Max-NCI)
:   steps 9-10 are the same as Algorithm 4
```

Note that some of the rules passing steps 4–5 are redundant. That is, these rules have their confidence lower than their ancestors.

Table 4
Datasets used in the experiments.

Dataset	Tuples	Attributes	Types	Classes
Breast Cancer (Diagnostic)	569	30	*Real*	2
Ecoli	336	7	*Real*	8
Glass	214	9	*Real*	7
Image Seg	2310	19	*Real*	7
Iris	150	4	*Real*	3
Page Blocks	5473	10	*Int*, *Real*	5
Waveform	5000	21	*Real*	3
WineQ-Red	1599	11	*Real*	11
WineQ-White	4899	11	*Real*	11
Yeast	1484	8	*Real*	10

Definition 13 (*Rule Redundancy*)**.** Let r and r' be two range-based rules that have the same consequent c. If $r \prec r'$ and $\delta(r) \leq \delta(r')$, then r is *redundant*.

As they cannot have a greater support than their ancestors either, these rules are not useful. We therefore do not retain these rules (step 6), which is similar to the argument given in [2]. Note that according to Definition 13 only parent rules may render their children redundant in the split tree and therefore there is no need to remove a rule that has already been added to R. Due to the rule redundancy property each of the resulting rules represents a specialization of the same original rule, but with different support and confidence characteristics. This property is important in data characterization because they help describe possibly multiple characteristics of a given dataset.

6. Experimental results

In this section we report experimental results. We compare the proposed methods to *C4.5* and *RIPPER* in terms of their classification accuracy and their ability to characterize a given set of data. We also examine the effect of our density measure on the quality of rule induction. We compare our methods to *C4.5* and *RIPPER* because these two methods are widely used and studied, and they produce human-interpretable rules as we do in our work.

6.1. Experments setup

A number of datasets selected from the UCI repository [9] are used in the experiments. These datasets are among the most popular datasets used in the research community for studying classification and they vary in tuple and attribute size, the nature of their numerical attributes and the number of different class labels. Table 4 contains a summary of the characteristics of each dataset.

6.2. Classification experiments

This section presents the results of experiments on the performance of the proposed methods for classification. We first compare the classification accuracy of our methods to that of the Weka implementation of *RIPPER* algorithm and *C4.5* [10] and then report the effect of density on classification accuracy.

6.2.1. Classification accuracy

In this set of experiments, 50%, 60%, 70%, 80% and 90% of each dataset is used for training and the remaining data for testing. The training data is selected at random, and prediction is made by using a majority vote. That is, we use every rule in the result set to predict an unseen case, and the consequent with the most votes is deemed as the result of prediction.

The results are given in Fig. 3. In these experiments, σ_{min} and γ_{min} were set to 0.01 and δ_{min} varied in the range of 0.5 to 0.95. We ran experiments 5 times and report average results. Overall the *AllSupp* method did not perform well, suggesting that support alone is not a good criterion. The *MaxConf*, *MaxGain* and *AllConf* methods all performed well compared to JRip and J48, and outperformed them in many cases. This is largely the result of our CARM inspired approach: when it is difficult to obtain a set of non-overlapping rules to classify unseen cases accurately, it can be more reliable to derive as many credible rules from data as possible, and use them to classify unseen cases collectively. Our results show that this approach is more resilient than the more conventional cover and remove strategy.

Table 5 shows a summary of the average classification accuracy (over different training datasets) achieved in the experiments. The *AllSupp* method is omitted here, as its classification accuracy was not comparable to the other methods. As can be seen, our methods are comparable to JRip and J48 in terms of average classification accuracy with *AllConf* and *MaxGain* outperformed JRip and J48 on 6 of the 10 datasets.

It is also worth mentioning that the proposed methods are designed to achieve good data characterization as well as classification. While these experiments confirm that the new methods are able to achieve classification performance that is broadly comparable to two of the most popular rule mining solutions, they can achieve a superior performance in data characterization, as we will see in Section 6.3.

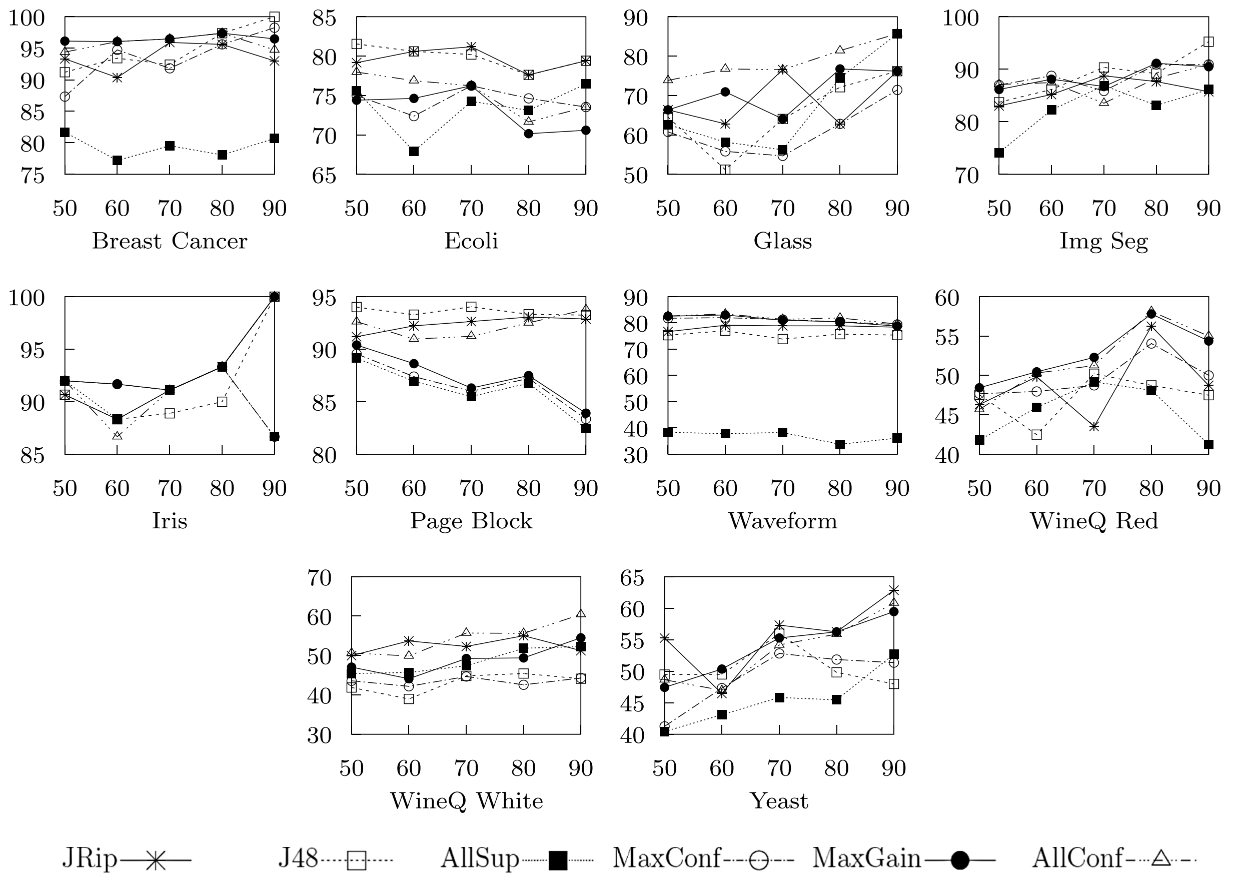

Fig. 3. Classification Accuracy. All charts have the same axis titles: *% of training data* for the *x*-axis and *classification accuracy* for the *y*-axis. Other than *AllSupp* our methods are comparable to JRip and J48 in classification accuracy.

Table 5
Average classification accuracy over different training datasets (%).

Dataset	Algorithm				
	JRip	J48	MaxConf	MaxGain	AllConf
Breast Cancer (Diagnostic)	93.63	94.88	93.55	**96.51**	95.80
Ecoli	79.60	**79.87**	74.36	73.20	75.25
Glass	68.94	65.60	61.09	70.86	**78.85**
Image Seg	86.06	**88.92**	88.69	88.52	87.46
Iris	92.69	91.58	90.96	**93.62**	92.62
Page Blocks	92.40	**93.58**	86.70	87.34	92.22
Waveform	78.40	75.47	80.97	81.18	**81.79**
WineQ-Red	48.94	47.30	49.69	**52.68**	52.07
WineQ-White	52.43	43.02	43.45	48.85	**54.44**
Yeast	**55.62**	50.54	48.91	53.75	53.30

6.2.2. *Effect of density*

To examine the effect of our new density parameter γ_{min} on classification accuracy, we set each dataset the same as in the previous section with the percentage of data used for training set at 70%, $\sigma_{min} = 0.01$ and $\delta_{min} = 0.8$. γ_{min} is however varied from 0 to 0.75. Note that the setting of particular σ and δ values are insignificant in these tests as our goal is to observe how classification accuracy changes when density varied. These experiments are performed using all four methods. The results of our experiments are presented in Fig. 4.

We observed that increasing the density threshold reduced the number of rules generated, and overall this has also led to the drop in classification accuracy. This is an interesting finding and is against our expectation. Close examination of the experimental results showed that high density settings resulted in many narrow ranges, and these ranges seemed to have caused some overfitting. Our results therefore suggest that the boundaries and association of ranges are more important than how many values are actually observed to be associated across the ranges. Our density measure is a conservative assessment that treats dangling cases as those that

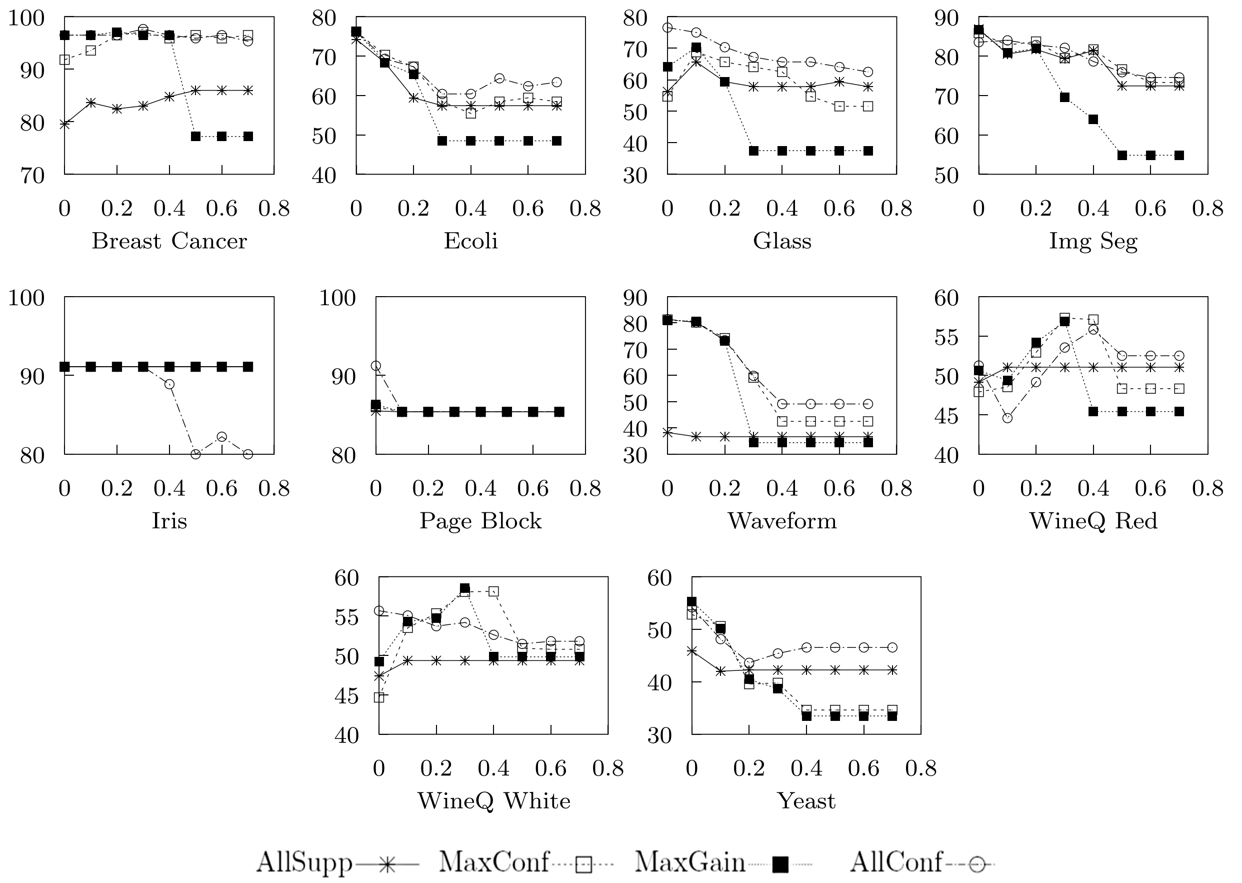

Fig. 4. Density effect on classification accuracy. All charts have the same axis titles: minimum confidence γ_{min} for the x-axis and *classification accuracy* for the y-axis. Observe that increasing density requirement tends to result in lower classification accuracy, suggesting that highly confident rules with narrow ranges may not always be desirable for classification.

are likely to lead to wrong predictions. The experiments results seem to suggest that such dangling cases would do little harm, and finding appropriate boundaries for the ranges involved in a rule has a more significant impact on classification accuracy.

We can however observe that large density seemed to produce more stable classification performance. That is, their classification performance does not vary significantly when the density is set high enough. This is largely expected as when the density is high, the discovered associated ranges would contain many actually observed cases in the dataset. As such, they tend to be more reliable when used to classify unseen cases.

6.3. *Characterization experiments*

The task of classification is well understood. In contrast, the task of data characterization and how the effectiveness of different rule sets may be compared for their data characterization power are less clear. In this section we introduce two measures to study the data characterization power of our proposed methods.

6.3.1. *Rule stability*

If discovered rules are relatively stable, then we may consider that the rules have characterized the data well. That is, if adding a small percentage of cases to a dataset does not change the resulting rules significantly, then a set of rules derived from data can be considered as having truly characterized data, since the key data characteristics should not have changed significantly when the data is slightly varied. We therefore use stability as a measure of data characterization quality.

Fig. 5 shows a comparison of differences in classification accuracy when changing the percentage of data used for training. That is, we measure the difference in classification accuracy when the amount of training data is changed from, say 50% to 60% or 60% to 70%. If the difference is small, then we consider our characterization of the data is stable, as adding 10% data to the dataset has not significantly changed the model derived from the data.

We compare our *MaxGain* and *AllConf* methods to JRip and J48 as *AllConf* gives best classification accuracy in our study and *MaxGain* is the only one that attempts to balance between support and confidence. For conciseness of presentation, we only report

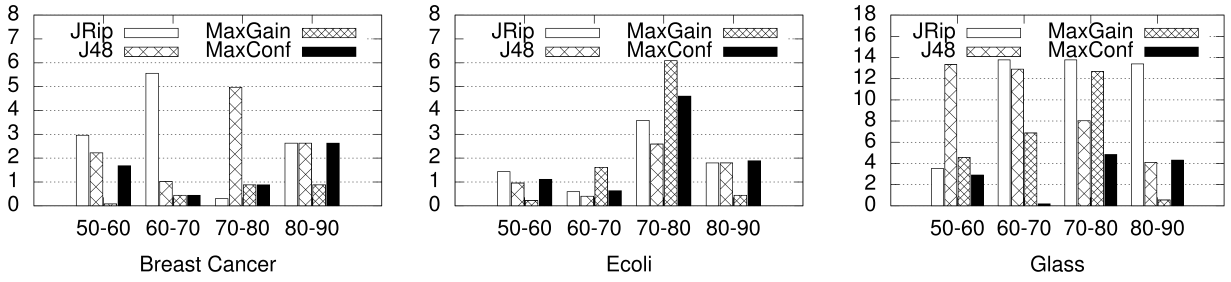

Fig. 5. Stability Test. All charts have the same axis titles: the *x*-axis shows the amount of data used in training, for example 50–60 indicates that we first used 50% data for training and then added 10% to the data and used the resultant 60% data for training. The *y*-axis shows the difference in classification accuracy between the two tests, for example, 3% for JRip when training data (Breast Cancer) is changed from 50% to 60%.

Table 6
Median density of 10 most confident and 10 most supported rules.

Dataset	10 most confident rules				10 most supported rules			
	J48	JRip	MaxGain	AllConf	J48	JRip	MaxGain	AllConf
Breast Cancer (Diagnostic)	0.015	0.398	**0.410**	**0.410**	0.019	0.398	0.596	**0.645**
Ecoli	0.013	0.126	0.058	**0.229**	0.042	0.126	0.515	**0.518**
Glass	0.026	0.089	**0.122**	**0.122**	0.056	0.089	0.227	**0.237**
Image Seg	0.021	0.147	**0.313**	**0.313**	0.094	**0.238**	0.161	0.161
Iris	0.320	**1.000**	0.625	0.625	0.320	**1.000**	0.625	0.749
Page Blocks	0.001	0.035	0.030	**0.076**	0.021	0.035	**0.999**	**0.999**
Waveform	0.011	**0.075**	0.019	0.035	0.025	0.163	0.191	**0.243**
WineQ-Red	0.009	0.018	0.020	**0.030**	0.024	0.064	0.174	**0.313**
WineQ-White	0.004	0.002	**0.016**	0.012	0.009	0.056	0.482	**0.684**
Yeast	0.007	**0.110**	0.026	0.015	0.038	0.132	0.160	**0.170**

the experimental results on three datasets here: the *Ecoli* and *Glass* datasets represent the cases where prediction using our methods is less and more accurate than the other two methods, respectively, whereas the *Breast Cancer* dataset is used as a representative case, where the average of classification accuracy by our methods outperformed the other two.

The figures demonstrate that both *AllConf* and *MaxGain* methods performed more consistently than JRip and J48 in most cases. We attribute this to the fact that CARM was used, and we are able to take inconsistent patterns within the data into account, producing more robust results. It is also interesting to observe that when the percentage of training data change from 80 to 90, the *MaxGain* method performed consistently well. Thus, it appears that in characterizing data, seeking balance between support and confidence is an effective strategy. These results show that our methods can result in stable solutions, which implies that they can capture key underlying patterns in data, hence a good solution for data characterization.

6.3.2. *Top k rule cohesion*

In this section we introduce another measure for data characterization: the median density of k most confident or most supported rules. The rationale for this measure is that intuitively if a rule covers a dense population of tuples and is highly supported and confident, then it characterizes a key pattern in a dataset. In other words, we want to test how "separate" the data covered by a set of rules is from the rest of the data, hence the rules have summarized a key pattern from the data.

In this set of experiments, we selected the 10 most confident and 10 most supported rules for each dataset and measured their median density. The results are presented in Table 6. In all cases 100% of datasets were used to mine the rules.

As we can see, J48 is completely outperformed by the other solutions in every single case. This is because J48 mines rules by performing point-based split, and in every experiment performed this resulted in rules with large ranges and a reduced density. *AllConf* performed better than *MaxGain* due to its rule formation heuristic: most confident rules tend to cover relatively fewer tuples, resulting in higher density. Comparing the two sets of results, it is useful to observe that for all methods selecting the rules with most support results in better density than the most confidence rules. Together with our analysis of density on classification accuracy in Section 6.2.2, this suggests that highly confident rules are better for classification whereas highly supported rules are better for data characterization.

Overall, our experiments show that the proposed methods offer better data characterization than the existing rule mining methods. We attribute this to the fact that the CARM methodology was used. Rather than following the cover and remove strategy, aiming to derive a single optimal model from the data, we allow many credible rules to be discovered and used collectively in classification and characterization. Consequently, a small change in data might affect the accuracy of some isolated rules that were learnt, but it would not affect the overall discovery. This is important to the derivation of stable, comprehensible characterization models from data.

7. Related work

One popular approach to obtaining ranges from numerical attributes is discretization and many techniques have been proposed to discretize numerical data as a pre-processing step in rule mining [11]. These methods rely on the discretization criteria used and there is no guarantee that relevant or optimal ranges will be captured during the pre-processing step. Some range merging techniques have also been considered during rule mining to refine the initial ranges obtained from discretization. For example, Srikant and Agrawal proposed to use equi-depth partitioning to group individual data items into initial ranges first, and then allow neighboring ranges to be combined based on a user specified threshold [3]. Wang et al. proposed a clustering based method which successively merges neighboring values into a range. Our method can also be broadly considered as discretization, except that we perform a top-down split as opposed to a bottom-up merge, and we perform this directly in the rule mining process without a pre-processing step. This allows more ranges to be explored during mining, resulting in more credible rules to be found.

Methods have also been proposed to extract optimal ranges directly from numerical data. Fukuda et al. [7] proposed an approach to mining ranges from a single numerical attribute that have maximum support using confidence as a constraint or mining ranges that have maximum confidence given the support threshold. The gain measure is used as an optimization to balance the two. This method was then extended to handling two numerical attributes [4]. The extended method was inspired by image segmentation and finds rectangular and admissible (connected x-monotone) regions from two dimensional data directly. Other extensions have been made to extract optimal gain regions from data [12,13], but they are all limited to handling a maximum of two numerical attributes. Aumann and Lindell proposed a statistical model to determine relevant ranges from data [2]. Their method focuses primarily on the rule consequent being a single numerical attribute and is able to find all ranges/sub-ranges from that attribute that statically deviate from the overall statistics in the entire attribute. Their method is again limited to dealing with one numerical attribute only in the rule antecedent. In contrast, our approach does not have this restriction and we allow multiple numerical attributes to form range associations as the antecedent of a rule.

Other works have considered extraction of ranges from numerical attributes as an optimization problem. For example, Mata et al. [14] proposed a solution that uses an evolutionary algorithm based on a fitness function that improves the generated ranges. This solution is able to mine overlapping ranges with high support but offers poor results in term of confidence. *Quantminer* [5] is a genetic algorithm based solution that delivers better results in terms of confidence and a reduced number of rules than the method given in [14]. Alatas et al. [15–17] proposed several evolution based algorithms for deriving associated ranges without needing to specify minimum support and confidence and are able to mine positive as well as negative ranges. These proposed methods are applicable to any number of numerical and categorical attributes, but they are designed to mine association rules, as such they offer no guarantees that their solutions will extract good ranges for classification and characterization. In contrast we use range split heuristics to extract rules for classification and characterization. The work reported in [18] is similar to the work presented here. However, their approach is based on the solution to the *Max Sum* problem [19], and derives an excessive number of min-$\sigma\gamma\delta$ rules. In comparison, we use a top-down split to find c-bounded ranges, and our methods are thus more scalable.

All range mining methods we discussed above are done in the context of either classification or association rule mining. In so doing they attempt to discover a single, optimal model from data. That is, they look for best ways of partitioning numerical data and deriving rules from such partitioning. In contrast, our work adopts the CARM methodology [6]. This allows effectively multiple models to be discovered from data, and to be used as a type of ensemble model for classification and characterization. While some methods have been proposed to mine rules using CARM, they are restricted to deal with categorical data only [20]. We, on the other hand, extend the approach to deal with numerical data in this paper.

8. Conclusions

In this paper, we proposed a method for finding range-based rules from numerical data in order to build classification and characterization models. Our method is inspired by the CARM approach: we search for associated ranges in a similar way to how associated items are searched for in conventional association rule mining, but we guide the range search with class values. To do so, we have introduced a number of heuristics for splitting ranges into sub-ranges and for range-based rule formation. This allows effectively multiple models to be discovered from data, and to be used as a type of ensemble model for classification and characterization. Our experimental results have shown that the new method is promising, and it outperformed popular rule mining methods such as C4.5 and RIPPLE in both data classification and characterization.

There are two issues to be addressed in future research. First, it is worth considering how rules with ranges in the consequent may be mined using the proposed approach, for example, discovering ranges in the consequent that are statistically significant [2] and use them to guide the formation of range-based rules. Second, extending our approach to mining rules from both numerical and categorical attributes can be considered. While it is possible to simply use additional categorical attributes to filter out some ranges and rules discovered by our method, it will be interesting to study if such categorical attributes can be used as an integral part of sub-range derivation. We plan to address these issues in our future work.

References

[1] P-N. Tan, M. Steibach, V. Kumar, Introduction to Data Mining, Addison Wesley, 2005.
[2] Y. Aumann, Y. Lindell, A statistical theory for quantitative assocation rules, J. Intell. Syst. 20 (3) (2003) 255–283.
[3] R. Srikant, R. Agrawal, Mining quantitative association rules in large relational tables, in: Proceedings of the ACMSIGMOD Conference on Management of Data, 1996, pp. 1–12.

[4] T. Fukuda, Y. Morimoto, S. Morishita, Data mining with optimized two-dimensional association rules, ACM Trans. Database Syst. 26 (2) (2001) 179–213.
[5] A. Salleb-Aouissi, C. Vrain, C. Nortet, Quantminer: a genetic algorithm for mining quantitative association rules. In Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI-07), 2007, pp. 1035–1040.
[6] B. Liu, W. Hsu, Y. Ma, Integrating classication and association rule mining, in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), 1998, pp. 80–86.
[7] T. Fukuda, Y. Morimoto, S. Morishita anf T. Yokuyama, Mining with optimized association rules for numeric attributes, in: Proceedings of the 15th ACM Symposium on Principles of Database Systems (PODS), 1996, pp. 182–191.
[8] U.M. Fayyad, K.B. Irani, Multi-interval discretisation of continuous-valued attributes for classification learning, in: Proceedings of the International Joint Conference on Uncertainty in AI, 1993, pp. 1022–1029.
[9] A. Frank, A. Asuncion, UCI Machine Learning Repository. SIGKDD Explorations, University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml, 2010.
[10] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, third ed., Morgan Kaufmann Publishers, 2011.
[11] S. García, J. Luengo, J. Antonion Sáez, V. López, F. Herrera, A survey of discretisation techniques: taxonomy and emperical analysis in supervised learning, IEEE Trans. Knowl. Discovery Data Eng. 25 (4) (2013) 734–750.
[12] S. Brin, R. Rastogi, K. Shim, Mining optimized gain rules for numeric attributes, in: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 135–144.
[13] W. Lian, D.W. Cheung, S.M. Yiu, An efficient algorithm for finding dense regions for mining quantitative association rules, Comput. Math. Appl. 50 (3–4) (2005) 471–490.
[14] J. Mata, J.L. Alvarez, J.C. Riquelme, An evolutionary algorithm to discover numeric association rules, in: Proceedings of the 2002 ACM symposium on Applied computing, 2002, pp. 590–594.
[15] B. Alatas, E. Akin, An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules, Soft Comput. 10 (3) (2006) 230–237.
[16] B. Alatas, E. Akin, A. Karci, MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules, Appl. Soft Comput. 8 (1) (2008) 646–656.
[17] U. Can, B. Alatas, Automatic mining of quantitative association rules with gravitational search algorithm, Int. J. Softw. Eng. Knowl. Eng. 27 (3) (2017) 343–372.
[18] A. Tziatzios, J. Shao, G. Loukides, A heuristic method for deriving range-based classification rules, in: Proceedings of the Eighth International Conference on Fuzzy Systems and Knowledge Discovery, 2011, pp. 925–929.
[19] J. Bentley, Programming pearls, Commun. ACM 27 (27) (1984) 865–871.
[20] F.A. Thabtah, P.I. Cowling, A greedy classification algorithm based on association rule, Appl. Soft Comput. 7 (3) (2007) 1102–1111.