APRIORI-SD: Adapting Association Rule Learning to Subgroup Discovery

鍾岳峰

元智大學

pchomekimojuf@gmail.com

May 12, 2015

Overview

Outline for section 1

INTRODUCTION:

- Classical rule learning algorithms are designed to **construct classification** and **prediction rules** [12,3,4,7].
- Some of the questions on how to adapt classical classification rule learning approaches to subgroup discovery have already been addressed in [10] and a well-known rule learning algorithm **CN2** was adapted to subgroup discovery.
- In this paper we take a rule learner **APRIORI-C** instead of CN2 and adapt it to **subgroup discovery**, following the guidelines from [10].

INTRODUCTION:

We have implemented the **new** subgroup discovery algorithm **APRIORI-SD** in C++ by **modifying the APRIORI-C algorithm**.
The proposed approach performs subgroup discovery through the following **modifications** of the rule learning algorithm APRIORI-C:

- using a **weighting scheme** in rule post-processing.
- using **weighted relative accuracy** as a **new measure of** the quality of the rules in the post-processing step **when the best rules are selected**.
- **probabilistic classification** based on the class distribution of **covered examples** by individual rules
- area under the **ROC(ROC: Receiver Operating Characteristic)** curve rule set evaluation.

INTRODUCTION:

- This paper **presents the APRIORI-SD subgroup discovery algorithm**, together with its experimental evaluation **in selected domains of** the **UCI Repository** of Machine Learning Databases [13].

INTRODUCTION:

These three factors are important for subgroup discovery:

- smaller size enables better understanding
- higher coverage means larger subgroups
- higher significance

Above factors means that rules describe subgroups whose **class distribution is significantly** different from the entire population by no loss in terms of the area under the ROC curve and accuracy.

INTRODUCTION:

This paper is organized as follows.

- In Section 2 **the background** for this work is explained: the APRIORI-C rule induction algorithm, including **the post-processing step of selecting the best rules**.
- Section 3 presents **the modified APRIORI-C algorithm**, called **APRIORI-SD**, adapting APRIORI-C for subgroup discovery together with **the weighted relative accuracy measure**, probabilistic classification and rule evaluation in the **ROC space**.
- Section 4 presents the **experimental** evaluation on selected **UCI domains**.
- Section 5 concludes by **summarizing the results** and presenting plans for **further work**.
- While in Section 4 we present **the summary results of the experiments**, the complete results on all **the UCI data sets** are presented in the **Appendix**.

Outline for section 2

Background: The APRIORI-C Algorithm:

- The main advantage of APRIORI-C over its predecessors is **lower memory consumption**, **decreased time complexity** and **improved understandability of results**.
- We describe here just **the parts of the APRIORI-C** that are essential for the reader to understand the derived **APRIORI-SD algorithm**.

Background: The APRIORI-C Algorithm:

An association rule has the following form:

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

- where X,Y I
- X and Y are itemsets
- I is the set of all items

Background: The APRIORI-C Algorithm:

The **quality** of each **association rule** is defined by its **confidence and support.**

- Confidence of a rule is an estimate of the conditional probability of Y given X: p(Y | X).
- Support of a rule is an estimate of the probability of itemset X ∪Y : p(XY).

Confidence and support are computed as follows:

$$Confidence = \frac{n(XY)}{n(X)} = \frac{p(XY)}{p(X)} = p(Y|X), Support = \frac{n(XY)}{N} = p(XY) \tag{2}$$

where n(X) is **the number of transactions** that are supersets of itemset X and N is the number of **all the transactions**.

Background: The APRIORI-C Algorithm:

The association rule learning algorithm APRIORI is then adopted for classification purposes (APRIORI-C).

Implementing the following steps:

- Discretize continuous attributes.
- Binarize all (discrete) attributes.
- Run the APRIORI algorithm by taking in consideration only rules whose **right-hand side** consists of a single item, **representing** the value of the class attribute (while running APRIORI).
- Post-process this set of rules, selecting **the best** among them and use this rules to classify unclassified examples.

Here we describe just **the last step**, **the post-processing of rules and classification of unclassified examples**, which are **the ones** we changed to obtain APRIORI-SD.

Background: The APRIORI-C Algorithm: Post-processing by rule subset selection.

Post-processing by rule subset selection.

- The APRIORI-C algorithm induces rules according to the parameters **minimal confidence and minimal support** of a rule [7].
- The setting of these two parameters is often such that the algorithm induces **a large number of rules**, which hinders the understandability and usability of induced rules.
- Moreover, there are **the problems of rule redundancy**, incapability of classifying examples and poor accuracy in domains with unbalanced class distribution.
- **A way to avoid these problems** is **to select just some best rules** among all the induced rules. APRIORI-C has **three ways of selecting such best rules：**

Background: The APRIORI-C Algorithm: Use N best rules:

Use N best rules:

- The algorithm first **selects the best rule** (rule having the highest support), then eliminates **all the covered examples**, **sorts the remaining rules** according to support and repeats the procedure.
- This procedure is **repeated until N rules are selected** or there are **no more rules to select** or there are **no more examples to cover.**
- The algorithm then **stops and returns** the classifier in the form of an IF-THEN-ELSE rule list.

Background: The APRIORI-C Algorithm: Use N best rules for each class:

Use N best rules for each class：

- The algorithm behaves in **a similar way** as the '**use N best rules**'case, selecting **N best rules for each class** (if so many rules exist for each class).
- This way the rules for **the minority class** will also find their way into the classifier.

Background: The APRIORI-C Algorithm: Use a weighting scheme to select the best rules:

Use a weighting scheme to select the best rules:

- The algorithm again behaves in a similar way as ‘use N best rules’.
- The **difference** is that **covered examples** are **not eliminated immediately**, but instead **their weight is decreased**.
- They are then **eliminated** when **the weight falls below a certain threshold** (e.g., when an example has been covered more than K times).

The details of the weighting scheme together with **the threshold** used are given in Section 3, describing APRIORI-SD.

Outline for section 3

APRIORI-SD:

APRIORI-SD

The main modifications of **the APRIORI-C algorithm**, making it appropriate for **subgroup discovery**, involve the implementation of **a new weighting scheme in post-processing**, **a different rule quality function** (the weighted relative accuracy), the probabilistic classification of unclassified examples and the area under **the ROC curve rule** set evaluation.

APRIORI-SD: Post-processing Procedure

The post-processing procedure is performed as follows:

repeat

- sort rules from best to worst in terms of the weighted relative accuracy quality measure (see Section 3.3)

- decrease the weights of covered examples (see Section 3.2)

until

all the examples have been covered or there are no more rules

APRIORI-SD: The Weighting Scheme Used in Best Rule Selection

- In the **'use a weighting scheme to select best rules'** post-processing method of APRIORI-C described in Section 2, **the examples covered by the 'currently'best rule** are **not eliminated but instead re-weighted**.
- This approach **is more suitable for the subgroup discovery process** which is, in general, aimed at **discovering interesting properties** of subgroups of the entire population.
- **The weighting scheme** allows this.

APRIORI-SD: The Weighting Scheme Used in Best Rule Selection

- **The weighting scheme treats examples** in such a way that covered positive examples **are not deleted** when the currently ‘best’rule is selected in the post-processing step of the algorithm.
- Instead, **each time a rule is selected**, the algorithm **stores with each example a count** that **shows how many times** (with how many rules selected so far) the example **has been covered so far**.
- Initial weights of all positive examples e_j equal 1, $w(e_j , 0) = 1$, which denotes that the example has not been covered by any rule, meaning **‘among the available rules select a rule which covers this example, as this example has not been covered by other rules’**, while lower weights mean **‘do not try too hard on this example’.**

APRIORI-SD: The Weighting Scheme Used in Best Rule Selection

- Weights of positive examples covered by the selected rule decrease according to the formula w(e_j,i) $= \dfrac{1}{i+1}$.
- In the first iteration all target class examples contribute the same weight w(e_j,0) $= 1$, while in the following iterations the contributions of examples are inverse proportional to their coverage by previously selected rules.
- In this way the examples already covered by **one or more** selected rules **decrease their weights** while rules covering many yet uncovered target class examples whose weights have not been decreased will have a greater chance to be covered in the following iterations.

Outline for section 4

Outline for section 5

Thank you for your attention