Supervised and experiential learning - Work 1

A rule-based classifier

Natalia Jakubiak
Student number: 12
Algorithm: RULES

April 9, 2021

Master in Artificial Intelligence

Universitat Politècnica de Catalunya

Contents

1 Introduction

The goal of this work is to implement and validate a rule-based classifier. Rule-based classifiers are just another type of classifier which makes the class decision depending by using various IF-THEN rules. These rules are easily interpretable and thus these classifiers are generally used to generate descriptive models. The condition used with “if” is called the antecedent and the predicted class of each rule is called the consequent.
The algorithm assigned to my student number (12) is **RULES**.

2 Algorithm

2.1 Description

RULES (RULe Extraction System) works based on the concept of separate-and-conquer to directly induce rules from a given training set and build its knowledge repository. It is used to build a predictive model based on given observation. The resulting rules are stored in an ‘IF condition THEN conclusion’ structure; condition is constituted by a attribute-vale pair [1]. If the number of attributes is $\mathrm{n_a}$, a rule may contain between one and $\mathrm{n_a}$ conditions, each of which must be a different attribute-value pair. Only the conjunction of conditions is permitted in a rule, and therefore the attributes must all be different if the rule comprises more than one condition.

The rule-forming procedure starts with forming simple rules with only one condition in the first iteration and continues by building rules with two conditions (second iteration), three conditions (third iteration) and so on. The maximum number of iterations that algorithm may require to induce the rules covering all training instances is $\mathrm{n_a}$.

In the first iteration, each element of the array of attributes and values is examined to decide whether it can form a rule with that element as the condition. For the whole set of examples, if a given element applies to only one class, then it is a candidate for forming a rule. If it pertains to more than one class, it is passed over and the next element is examined. When all elements of the array have been looked at, the whole set of examples is checked for any example that cannot be classified by the candidate rules. If there are no unclassified examples, the procedure terminates. Otherwise, a new array is constructed that comprises attributes and values contained in all the unclassified examples. In the second iteration, elements of the array are examined in pairs to determine whether they apply to only one class in the whole set of examples. If an attribute value is not present any longer in the set of unclassified examples, it is not considered anymore as a candidate condition. By that the number of considered attribute values decreases with every iteration. The rule-creating procedure continues until all examples are correctly classified or the number of iterations is equal to $\mathrm{n_a}$. For each iteration after the first, candidate rules extracted in the current iteration are checked against previously obtained rules, i.e. if the new rule covers all examples that the previously induced rule and more (more examples), the old rule is deleted and the new one is added to the stock or if the training example can be classified by a simpler rule with less conditions.

Algorithm 1 General structure of RULES algorithm

$NumberCombinations \leftarrow 1$
$Rules \leftarrow \emptyset$
while $NumberUnclassifiedInstances > 0 and NumberComibnations \leq NumberAttributes$ **do**
    for each condition in Conditions **do**
        Find all selectors (pairs AttributeValue) from NONclassified instances
        Form Conditions as a combination of NumberCombinations selector
        if all instances satisfying the condition belong to the same class C **then**
            Create the rule NewRule based on Condition
            Check irrelevant conditions against all previously obtained rules
            if irrelevant conditions recognized **then**
                $Rules \leftarrow Rules - IrrelevantRule$
            end if
            $Rules \leftarrow Rules + R$
        end if
    end for
    $NumberCombinations \leftarrow NumberCombinations + 1$
end while
return $Rules$

2.2 Implementation

I decided to implement classifier algorithm as a class named RULES. This class has two main functions that can be called after initializing an object:

- *fit()*: this function allows to fit the model with a training dataset and the training target values. It is used to specify the rules.
- *predict()*: assignment of labels to a list of observations.

The fit function is iterative based, it repeatedly forms candidate conditions for a rules as a combination of selectors (pairs Attribute-Value). The number of selectors being used to create conditions is one for the first iteration and is increased by one with each successive iteration. Each iteration can be described in following steps which are repeated as long as there are unclassified examples and the number of attributes in the database is greater than number of iteration (NumberCombinations):

1. Find all selectors(pairs Attribute-Value) from NON-classified instances
2. Form conditions as a combination of NumberCombinations selectors

3. For each condition in conditions set check if there are instances which satisfies this condition

4. If there are unclassified instances which satisfy condition - check the relevance of the condition against previously obtained rules. If irrelevant rule is recognized - delet it from the rules' list.

5. Create new rule (if adequate conditions are fulfilled)

6. Delete all the examples that are covered by the new rule (if the new rule was generated)

When the rule-forming process is finished, the set of obtained rules are displayed in a interpretable way:
`IF condition THEN conclusion`
Also for each rule its coverage and precision are printed. Coverage is counted as the percentage of training set instances which satisfy the antecedent conditions of a particular rule. Later, when the model is fitted with training data, function *predict()* can be called to classify the given test examples according to the induced rules. The list of assigned labels is returned.

3 Databases

To analyze behavior of implemented algorithm I decided to use following datasets from from UCI ML: balance-scale dataset, car dataset and nursery dataset. Details about them are presented in the table 1.

Name of dataset	Size	Type of attributes	Number of instances	Number of Attributes	Number of classes	Missing values
Balance-scale	Small	Categorical	625	4	3	No
Car	Medium	Categorical	1728	6	4	No
Nursery	Large	Categorical	12960	8	5	No

Table 1: Details of selected datasets

3.1 Balance-scale dataset

This data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of (left-distance * left-weight) and (right-distance * right-weight). If they are equal, it is balanced.

Class Values: L, B, R
Attributes:

- buying: vhigh, high, med, low.
- Left-Weight: 1, 2, 3, 4, 5
- Left-Distance: 1, 2, 3, 4, 5
- Right-Weight: 1, 2, 3, 4, 5
- Right-Distance: 1, 2, 3, 4, 5

3.2 Car evaluation dataset

Car Evaluation Database was created to evaluate cars according to chosen parameters. Test subjects were asked about their opinion about certain cars. Simple features like safety, number of doors, etc. were extracted from these cars which can be used to predict the subjects opinion.
Class Values: unacc, acc, good, vgood
Attributes:

- buying: vhigh, high, med, low.
- maint: vhigh, high, med, low.
- doors: 2, 3, 4, 5more.
- persons: 2, 4, more.
- lug_boot: small, med, big.
- safety: low, med, high.

3.3 Nursery dataset

Nursery Database was derived from a hierarchical decision model originally developed to rank applications for nursery schools. The final decision depended on three subproblems: occupation of parents and child's nursery, family structure and financial standing, and social and health picture of the family.
Class Values:not_recom, recommend, very_recom, priority, spec_prior.
Attributes:

- parents: usual, pretentious, great_pret
- has_nurs: proper, less_proper, improper, critical, very_crit
- form: complete, completed, incomplete, foster
- children: 1, 2, 3, more
- housing: convenient, less_conv, critical

- finance: convenient, inconv
- social: non-prob, slightly_prob, problematic
- health: recommended, priority, not_recom

3.4 Data preparation methods

The aim of data preparation is transforming raw data into a representation that allows learning algorithms to get the most out of the data and make skillful rules. In the algorithm evaluation process I decided to manually specify the data preparation techniques for each dataset separately to use for the given algorithms based on the detailed knowledge of the dataset.

Because I have chosen databases with all attributes of categorical type, the missing values are replaced by the most frequent values within each column. The major drawback of this method is that it does not factor the correlations between features, because the imputed values are just estimates and will not be related to other values inherently. In the algorithm evaluation process, I tested various datasets where the missing value complement function was used. As a last resort, to present the evaluation of the algorithm in this report, I use sets that do not have the missing data.

In the preprocessing step the dataset is splitted into train and test sets. For all datasets 80% of the examples were used for training (inducing rules), so the remaining 20% could be used for testing.

4 Evaluation

Besides the datasets used to evaluate the prediction results of the implemented RULES algorithm, I also used two additional datasets for debugging purposes: Season dataset and Contact-lenses dataset. The Season Classification Problem presentend in the article [1], the training set for which is given in table 2.

Example	Weather	Trees	Temperature	Season (Class)
1	rainy	yellow	average	*autumn*
2	rainy	leafless	low	*winter*
3	snowy	leafless	low	*winter*
4	sunny	leafless	low	*winter*
5	rainy	leafless	average	*autumn*
6	rainy	green	high	*summer*
7	rainy	green	average	*spring*
8	sunny	green	average	*spring*
9	sunny	green	high	*summer*
10	sunny	yellow	average	*autumn*
11	snowy	green	low	*winter*

Table 2: Training set for Season Classification Problem. Source [1]

Each object in the training set is described in terms of the following attributes: Weather, with values {rainy, sunny, snowy}, Trees, with values {green, yellow, leafless}, and Temperature, with values {low, average, high}. Each object belongs to one of four classes, winter, summer, autumn, or spring.

The output obtained for this dataset is following:

```
R1: IF Trees = yellow THEN autumn
Coverage: 2 instances 18.18% of all instances
Precision: 100.00%
R2: IF Temperature = high THEN summer
Coverage: 2 instances 18.18% of all instances
Precision: 100.00%
R3: IF Temperature = low THEN winter
Coverage: 4 instances 36.36% of all instances
Precision: 100.00%
R4: IF Trees = green AND Temperature = average THEN spring
Coverage: 2 instances 18.18% of all instances
Precision: 100.00%
R5: IF Trees = leafless AND Temperature = average THEN autumn
Coverage: 1 instances 9.09% of all instances
Precision: 100.00%
```

The number of extracted rules is equal to 5 and equal to the number obtained in the original article [1]. The obtained rules covered all training instances. The precision of each rule is 100%. Because of the rule-checking procedure adopted, the rules obtained involve only relevant conditions. The details (the set of induced rules, the coverage and precision of each rule) for Contact-lenses dataset can be found in ./Documentation/contact-lenses.txt file.

4.1 Results for the tested problems

Name of dataset	Number of instances	Number of rules	Accuracy
Balance-scale	625	246	0.72
Car	1728	222	0.64
Nursery	12960	525	0.98

Table 3: Summary for the tested datasets

Table 3 shows the results of training and testing process for all datasets. The first thing that can be noticed is that the accuracy of predicitions made by the algorithm increases with the number of training examples. However, can be seen that number of extracted rules does not have the same property. In the case of chosen datasets, especialy for balance-scale dataset, the number of rules is related to imbalance in the dataset and the number of attribue-value pairs. Because RULES does not employ irrelevant condition, the number of conditions in extracted rules is likely to be fewer than for other algorithms. Overall can be seen that for all three datasets a high accuracy could be achieved. The cause of this might be that for every dataset it was possible to construct simple rules with only one condition that cover a lot of examples (tables 4, 5, 6). The full set of rules with precision and coverage is stored in separate files in Documentation folder.

Rule index	Coverage [%](training)	Precision(training)	Coverage[%](test)
R1	3.0	100%	8.0
R2	3.8	100%	4.8
R3	3.0	100%	4.0
R4	3.2	100%	2.4
Total	**13%**		**17.2%**

Table 4: Summary of 4 first rules for Balance-scale dataset

Rules related to table 4:

```
R1: IF LeftW = 5 AND LeftD = 5 THEN L
R2: IF LeftW = 1 AND RightD = 5 THEN R
R3: IF LeftD = 5 AND RightW = 1 THEN L
R4: IF RightW = 5 AND RightD = 5 THEN R
```

Rule index	Coverage [%](training)	Precision(training)	Coverage[%](test)
R1	33.2	100%	34.4
R2	22.3	100%	23.7
R3	1.7	100%	1.4
R4	1.8	100%	2.6
Total	59%		62.1%

Table 5: Summary of 4 first rules for Car dataset

Rules related to table 5:

```
R1: IF doors = 2 THEN unacc
R2: IF lug_boot = low THEN unacc
R3: IF buying = vhigh AND persons = small AND lug_boot = med THEN unacc
R4: IF maint = 2 AND doors = more AND persons = small THEN unacc
```

Rule index	Coverage [%](training)	Precision(training)	Coverage[%](test)
R1	33.4	100%	32.8
R2	2.1	100%	2.6
R3	2.2	100%	2.2
R4	2.1	100%	2.6
Total	39.8%		40.2%

Table 6: Summary of 4 first rules for Nursery datase

Rules related to table 6:

```
R1: IF recommended = not_recom THEN not_recom
R2: IF usual = pretentious AND proper = less_proper
    AND recommended = priority THEN priority
```

```
R3: IF usual = pretentious AND proper = proper
    AND recommended = priority THEN priority
R4: IF usual = usual AND proper = improper
    AND recommended = priority THEN priority
```

In the table 5 can be seen that with only four rules that each contain only one condition almost the 2. Both in the training and test dataset the two rules cover nearly half of the examples. The balance-scale and nursery datasets contain more complex rules, thus better accuracy is achieved.

5 Execution of the code

The implementation of this assignment was performed using Python 3.6. The included packages that were exploited and are required to execute the script:

- **Numpy:** mathematical operations
- **Pandas:** reading databases
- **Sklearn:** splitting the dataset into test and train sets; counting accuracy score

To execute the algorithm the size of the dataset must be specified as a command line argument. The options are: small, medium, large. More information can be found using the help command.
To reproduce the results from this report the following three commands should be executed:

```
python ./Source/main.py small
python ./Source/main.py medium
python ./Source/main.py large
```

6 Conclusion

In this document I have demonstrated the implementation and evaluation of the RULES algorithm. RULES has been applied to different problems, and the results obtained have shown that in all cases the number of generated rules is small and the achieved accuracy is relatively high. The experiments I have conducted show that, number of induced rules does not necessarily increase with the number of dataset instances. The important factor is how well dataset is balanced and how many attribute-value pairs have.

References

[1] D.T. Pham and M.S. Aksoy. Rules: A simple rule extraction system. *Expert Systems with Applications*, 8(1):59–65, 1995.