A Rule-Based classifier implementation: PRISM

Ángel Poc

4th of April, 2019

1 Introduction

In this work we are going to study the behavior of the rule-based classifier systems, concretely PRISM [1]. We will describe how PRISM selects the rules, our implementation and finally we will select some datasets to test the performance of the implementation by means of classification error and non-classified instances.

2 Datasets

In this section we will describe the datasets which will be used to evaluate the algorithm. All of them have been collected from the UCI repository [2] and they present different characteristics.

2.1 Lenses

This dataset will be used to test the correctness of the algorithm developed, as it appears in [1]. The low number of instances will make unfeasible to evaluate this dataset in terms of performance with cross validation.

2.2 Iris

This is one of the most known and used datasets in machine learning. It contains instances from 3 kinds of iris plant being described with 4 numerical attributes, related to, for instance, petal or sepal length. This dataset contains 150 instances equally distributed to each class.

This dataset is known for being easily classifiable and that is the reason why we are using it, to see if our algorithm is able to cope with easy problems. Using this dataset we are also testing if our discretization process for numerical variables performs correctly.

2.3 Breast cancer

On the contrary, we have Breast Cancer dataset, which is a dataset that contains only categorical attributes. It contains 9 features related to characteristics of women that have suffered from breast cancer and the objective is to classify whether the pacient has had recurrence events or not. It contains 286 instances with only 9 missing values.

2.4 Horse Colic

Finally, the Horse Colic dataset has been selected because it is known to be a difficult problem. It contains only 300 instances, representing horses, described with 27 attributes of continuous, discrete and nominal domain. The goal is to classify if a certain horse lived, died or was euthanized after a lesion. The 33% of the values are missing. We want to see if this PRISM is able to deal with really complex problems as this one.

3 PRISM

Rule-based classifier systems are born from the same idea of information gain from decision trees but disposing of the tree structures, making systems more interpretable for humans.

One of those rule-based systems is PRISM. It starts with general rules and modifies them making more specific rules. It tries to make rules that cover the highest number of instances possible mantaining 100% accuracy if it is possible (there are no several instances with the same class label).

The algorithm begins with the pair atribute-value α_x with highest a-posteriori probability, $p(\delta_n|\alpha_x)$, being δ_n the class label. Once selected, removes this attribute from the possible attributes, and iteratively selects the next one that maximizing the a-posteriori probability having in mind the ones previously selected.

Algorithm 1 PRISM algorithm

```
 1: procedure PRISM(data, labels)          ▷ Returns the set of rules produced by PRISM
 2:     rules ← empty list
 3:     for each class in $c_i$ do
 4:         instances_left ← instances of data with label $c_i$
 5:         while ¬isEmpty(instances_left) do          ▷ We follow until all the instances have a rule
 6:             all_pairs ← generate all possible attribute/value pairs according to instances_left
 7:             current_rule ← new rule(class c_i)
 8:             while ¬ isPerfect(current_rule) and length(all_pairs)>0 do
 9:                 best_pair_rule ← new rule(class c_i)          ▷ Dummy rule with 0 precision and coverage
10:                 for each pair in all_pairs do
11:                     test_rule ← current_rule
12:                     addToAntecedent(test_rule,pair)
13:                     precision, coverage ← calculate_precision(test_rule)
14:                     if precision > get_precision(best_pair_rule) then
15:                         best_pair_rule ← test_rule
16:                     else if precision == get_precision(best_pair_rule) ∧ coverage > get_coverage(best_pair_rule)
    then
17:                         best_pair_rule ← test_rule
18:                     end if
19:                 end for
20:                 current_rule ← best_pair_rule
21:                 Remove the new selected attribute in best_pair_rule from all_pairs
22:             end while          ▷ The rule is perfect or there are not more attributes available
23:             append(rules, current_rule)
24:         end while          ▷ All instances of a class have been classified
25:     end for          ▷ All classes have been classified
26:     Sort rules according to precision and coverage
27: end procedure
```

The work from Cendrowska [1], has some interpretation issues, as it could be understood that as you advance in the algorithm, you calculate the precision and coverage only for the instances you are focused on in that moment, however, I think that what they propose is to calculate it for the whole dataset, to achieve a sytesm which is capable of generalizing better. The pseudocode for my implementation is described in algorithm 1.

4 Implementation details

The algorithm has been implemented in Python. The data is first loaded, requiring the csv files to have a header indicating the name of the attributes to increment the interpretability of the rules. After that, the data is pre-processed, substituting the missing values by the next valid observation. This simplistic approach has been taken because the goal of this work is not to study missing values filling approaches, but to study PRISM's efficacy.

The rules have been implemented with a class with antecedent, consequent, precision and coverage, where the antecedent is a list of tuples (attribute,value). All the data has been stored in *Dataframes*, and the precision has been calculated making queries to the dataframe with the values of the antecedent of the rules.

The rules are first displayed in discovery order and later are shown ordered by precision and coverage.

5 Evaluation

In this section we are going to evaluate the performance of PRISM according to the accuracy and the ratio of non classified instances. Table 1 shows the results obtained for each dataset, showing the classification error, the non classified instances rate, the number of rules produced and the execution time mean for a 10 fold cross validation. The shown rules refer to the system trained with all the instances. In the following subsections we will comment specifically the results for each dataset.

Dataset	Error	Non Classified Rate	Rules	Time [s]
Lenses	0.29±0.26	0.10±0.17	9	0.50±0.07
Iris	0.066±0.073	0.0066±0.0199	14	0.96±0.15
Breast Cancer	0.38±0.10	0.090±0.051	172	75.50±4.71
Horse Colic	0.35±0.08	0.046±0.047	178	163.41±12.05

Table 1: PRISM statistics for the selected datasets

5.1 Lenses

In table 1 one we can observe the the results obtained by PRISM in this dataset. As the dataset only contains 24 instances, the statistics will not be representative. However, we are going to review it because it was used in [1], and therefore we can test if the produced rules are the same. The rules obtained for our implemenation are the following, showed by discovery order, as in the reference paper:

1. d='1' → 3 Prec: 1.0 Coverage: 0.5
2. a='3' and b='2' and c='2' → 3 Prec: 1.0 Coverage: 0.0833
3. a='3' and b='1' and c='1' → 3 Prec: 1.0 Coverage: 0.0833
4. b='2' and c='2' and a='2' → 3 Prec: 1.0 Coverage: 0.0833
5. c='1' and d='2' and b='2' → 2 Prec: 1.0 Coverage: 0.125
6. c='1' and d='2' and a='1' → 2 Prec: 1.0 Coverage: 0.0833
7. c='1' and d='2' and a='2' → 2 Prec: 1.0 Coverage: 0.0833
8. c='2' and d='2' and b='1' → 1 Prec: 1.0 Coverage: 0.125
9. c='2' and d='2' and a='1' → 1 Prec: 1.0 Coverage: 0.0833

We can see that the produced rules are exactly the same produced in the original implementation, the only changes are the order in which the rules are displayed and the order of some selectors.

5.2 Iris

Looking at table 1 we can see the performance of the system on the dataset. We can see how the number of rules is relatively small for 150 instances and how it is able to classify a high proportion of the instances.

Following, we can see the predicted rules ordered by precision and coverage:

1. petal_length='(0.994, 2.967]' → Iris-setosa Prec: 1.0 Coverage: 0.33
2. petal_width='(0.9, 1.7]' and petal_length='(2.967, 4.933]' and sepal_length='(5.5, 6.7]' → Iris-versicolor Prec: 1.0 Coverage: 0.22
3. petal_width='(0.9, 1.7]' and petal_length='(2.967, 4.933]' and sepal_width='(2.8, 3.6]' → Iris-versicolor Prec: 1.0 Coverage: 0.14
4. petal_width='(0.9, 1.7]' and petal_length='(2.967, 4.933]' and sepal_length='(6.7, 7.9]' → Iris-versicolor Prec: 1.0 Coverage: 0.02
5. petal_width='(0.9, 1.7]' and petal_length='(2.967, 4.933]' and sepal_width='(1.998, 2.8]' and sepal_length='(4.296, 5.5]' → Iris-versicolor Prec: 0.90 Coverage: 0.066
6. petal_width='(0.9, 1.7]' and sepal_width='(2.8, 3.6]' and sepal_length='(5.5, 6.7]' → Iris-versicolor Prec: 1.0 Coverage: 0.12
7. petal_width='(0.9, 1.7]' and sepal_length='(5.5, 6.7]' and sepal_width='(1.998, 2.8]' and petal_length='(4.933, 6.9]' → Iris-versicolor Prec: 0.25 Coverage: 0.0066
8. petal_length='(2.967, 4.933]' and sepal_width='(2.8, 3.6]' and sepal_length='(5.5, 6.7]' and petal_width='(1.7, 2.5]' → Iris-versicolor Prec: 0.33 Coverage: 0.0066
9. petal_width='(1.7, 2.5]' and petal_length='(4.933, 6.9]' → Iris-virginica Prec: 1.0 Coverage: 0.266
10. petal_width='(1.7, 2.5]' and sepal_width='(1.998, 2.8]' → Iris-virginica Prec: 1.0 Coverage: 0.1
11. petal_width='(1.7, 2.5]' and sepal_length='(5.5, 6.7]' and sepal_width='(2.8, 3.6]' and petal_length='(2.967, 4.933]' → Iris-virginica Prec: 0.66 Coverage: 0.013
12. petal_length='(4.933, 6.9]' and sepal_length='(6.7, 7.9]' → Iris-virginica Prec: 1.0 Coverage: 0.11
13. petal_length='(4.933, 6.9]' and sepal_width='(1.998, 2.8]' and sepal_length='(5.5, 6.7]' and petal_width='(0.9, 1.7]' → Iris-virginica Prec: 0.75 Coverage: 0.02
14. sepal_width='(1.998, 2.8]' and petal_length='(2.967, 4.933]' and sepal_length='(4.296, 5.5]' and petal_width='(0.9, 1.7]' → Iris-virginica Prec: 0.090 Coverage: 0.0066

It is noticeable how the first rule coveres the 33% of the dataset with only one attribute. This also highlights the need of deciding carefully how to discretize the data.

5.3 Breast cancer

We can notice in table 1 how in this case PRISM starts to give poorer results. The mean error rate has risen up to 0.38%, and the variability is high depending on the fold, with a standard deviation of 0.10. We can see how the algorithm has not been to assign any label to the 9% of the instances on average, meaning that the algorithm does not seem to generalize enough to classify new data. We can see how the number of rules has increased greatly with respect to the previous examples, as well as the execution time.

In the following list, due to the total length of the rules list, there only are shown the first three and the last three rules according to precision and coverage ordering:

1. tumor_size='10-14' and node_caps='no' $\rightarrow$ no-recurrence-events Prec: 1.0 Coverage: 0.0944
2. deg_malig='1' and menopause='ge40' and age='50-59' $\rightarrow$ no-recurrence-events Prec: 1.0 Coverage: 0.052
3. deg_malig='1' and menopause='ge40' and irradiat='no' and breast='left' $\rightarrow$ no-recurrence-events Prec: 1.0 Coverage: 0.052

170. tumor_size='25-29' and menopause='premeno' and age='50-59' and inv_nodes='0-2' and breast='right' and node_caps='no' and deg_malig='1' and breast_quad='left_up' and irradiat='no' $\rightarrow$ recurrence-events Prec: 0.50 Coverage: 0.0034
171. breast='left' and menopause='premeno' and breast_quad='left_low' and tumor_size='20-24' and age='40-49' and deg_malig='2' and inv_nodes='0-2' and node_caps='no' and irradiat='no' $\rightarrow$ recurrence-events Prec: 0.50 Coverage: 0.0034
172. deg_malig='2' and inv_nodes='0-2' and irradiat='no' and breast='right' and menopause='premeno' and node_caps='no' and age='40-49' and tumor_size='25-29' and breast_quad='left_low' $\rightarrow$ no-recurrence-events Prec: 0.33 Coverage: 0.0034

We can notice again how the first rule covers a great amount of the instances, the 9.5%. It is also remarkable that the second and third rules share the first two selectors. The last rules do not have a precision of 1, probably because the involved instances have different labels. It is also interesting to see how specific those rules are, as they cover only less than the 1% of the data and make use of all the attributes.

5.4 Horse Colic

In table 1 we can perceive how the error and non classified rates have decreased, anyway they are still very high and result in very poor performances. The number of rules has increased, but this was expected as there are 26 more instances than in the previous case and remarkably more attributes, however the times has increased more than two times.

The list below shows the first the last three rules according to precision and coverage ordering:

1. mucous_membrane='normal_pink' and surgical_lesion='no' and surgery='no' → lived Prec: 1.0 Coverage: 0.13

2. mucous_membrane='normal_pink' and surgical_lesion='no' and total_protein='(3.214, 31.867]' → lived Prec: 1.0 Coverage: 0.10

3. mucous_membrane='normal_pink' and temp_of_extremities='normal' and respiratory_rate='(7.912, 37.333]' → lived Prec: 1.0 Coverage: 0.10

176. abdominal_distention='slight' and nasogastric_reflux_ph='(5.333, 7.5]' and mucous_membrane='dark_cyanotic' → lived Prec: 1.0 Coverage: 0.0033

177. pulse='(81.333, 132.667]' and cp_data='yes' and abdomo_protein='(3.433, 6.767]' and nasogastric_reflux='none' and respiratory_rate='(7.912, 37.333]' and nasogastric_reflux_ph='(5.333, 7.5]' and total_protein='(3.214, 31.867]' and lesion_1='(-41.11, 13703.333]' and lesion_2='0' and lesion_3='0' and age='adult' and hospital_number='(513688.847, 2114193.667]' and surgical_lesion='yes' and peristalsis='hypomotile' and packed_cell_volume='(40.333, 57.667]' and abdomo_appearance='serosanguious' and temp_of_extremities='cool' and peripheral_pulse='reduced' and surgery='no' and rectal_temp='(39.0, 40.8]' and mucous_membrane='normal_pink' and capillary_refill_time='more_3_sec' and pain='depressed' and abdominal_distention='moderate' and nasogastric_tube='slight' and rectal_exam_feces='normal' and abdomen='distend_large' → died Prec: 0.5 Coverage: 0.0033

178. rectal_temp='(39.0, 40.8]' and packed_cell_volume='(40.333, 57.667]' and pulse='(81.333, 132.667]' and hospital_number='(513688.847, 2114193.667]' and lesion_1='(-41.11, 13703.333]' and lesion_2='0' and lesion_3='0' and age='adult' and nasogastric_tube='slight' and capillary_refill_time='more_3_sec' and abdominal_distention='moderate' and nasogastric_reflux='none' and surgery='no' and respiratory_rate='(7.912, 37.333]' and temp_of_extremities='cool' and peripheral_pulse='reduced' and mucous_membrane='normal_pink' and pain='depressed' and peristalsis='hypomotile' and nasogastric_reflux_ph='(5.333, 7.5]' and rectal_exam_feces='normal' and abdomen='distend_large' and total_protein='(3.214, 31.867]' and abdomo_appearance='serosanguious' and abdomo_protein='(3.433, 6.767]' and surgical_lesion='yes' and cp_data='yes' → euthanized Prec: 0.5 Coverage: 0.0033

We can see how m̈ucous_membraneẅith value pink is the first selector of the first three rules PRISM provides, meaning that it is very informative according to the class l̈ived:. It may be noticed how as it was impossible to achieve accuracy one in the last rules, there have been formed rules with up to 27 attributes, being too specific.

6 Conclussions

In this work we have studied the fundamentals of PRISM, a rule-based classifier system. We have also evaluated it, getting some not very successful results in terms of error and unclassified instances rate in the tested datasets.

We have seen that PRISM tends to create in some cases very specific rules in complex datasets, which could lead to overfit. We have also observed that the number of rules increases importantly according the number of instances and attributes of the datasets.

It is also important to note that the complexity order is high and that the execution time increases greatly with the number of instances and features.

References

[1] Jadzia Cendrowska. Prism: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349 – 370, 1987.

[2] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.