Supervised and Experiential Learning
Practical work 1: PRISM algorithm implementation

Pol Roca Llaberia

April 6, 2021

Master in Artificial Intelligence

Universitat Politècnica de Catalunya

1 Introduction

In this document we will see a possible implementation of the PRISM modular rule inducer algorithm [1], along with a brief evaluation of its performance by applying it to three different datasets and analyzing the obtained results.

2 Implementation

Algorithm 1 shows the pseudo-code of this implementation of PRISM, which is the core logic of what was then programmed in Python.

Algorithm 1 PRISM

Input: A supervised dataset $\mathcal{D}$ with a set of attributes $\mathcal{A}$
Output: A ruleset RS induced from $\mathcal{D}$

if $\mathcal{A}$ has numeric attributes **then**
    $\mathcal{D} \leftarrow \mathcal{D}$ with discretized numeric attributes
end if
$\mathcal{D} \leftarrow \mathcal{D}$ without duplicates
$RS \leftarrow \varnothing$ ▷ Initialize ruleset
for each class label L_i **do**
    $I \leftarrow \mathcal{D}$
    $C_i \leftarrow$ Number of instances in I having label L_i
    while $C_i > 0$ **do**
        $R \leftarrow$ New empty rule with target L_i
        $R_{Cov} \leftarrow I$ ▷ Instances covered by R
        $U \leftarrow \mathcal{A}$ ▷ Unused attributes
        $P \leftarrow 0$ ▷ Precision score
        while $|U| > 0$ and $P \neq 1$ **do**
            for each attribute A_j in U **do**
                for each possible value V_k of A_j in R_{Cov} **do**
                    $p \leftarrow$ Compute precision of R in R_{Cov} after adding selector A_j, V_k
                    if $p \geq P$ **then** ▷ Count true positives if it is equal
                        Save A_j, V_k as best selector
                        $P \leftarrow$ p
                    end if
                end for
            end for
            Append the best selector to the antecedent of R
            Remove the attribute of the best selector from U
            Update R_{Cov}
        end while
        $RS \leftarrow RS \cup R$
        $I \leftarrow I \setminus R_{Cov}$
        $C_i \leftarrow C_i - |R_{Cov}|$
    end while
end for

The algorithm is very straight forward just as the one proposed in the original paper. For each class label we generate rules that have 100% precision until all instances from that class are classified. Instances covered by a rule are temporarily removed and the rest of them are the target for the next rules until all of them are classified. Rules are generated incrementally by adding at each time the selector that achieves the best precision when it is added, using the number of true positives to break ties, and until all the instances covered by the rule are from the target label. In the actual implementation the recall score is also computed additionally for each rule, which is the second value in the last part when rules are printed out (being precision the first one).

Moreover, this algorithm constrains the dataset not to contain duplicates. In particular, this is done in line 4 where instances having the same attributes (without considering their label) are dropped. This is to prevent the inconsistency of having two identical cases with a different label, something that would prevent the creation of rules with the maximum precision as it should, apart from being illogical. Although this process could be done beforehand during the preprocessing stage, because the discretization step is included in the algorithm, it must be performed after that step since the discretization may produce those duplicates.

Speaking of the discretization, the user may choose by a parameter the strategy to be employed. Besides, in any case the default behavior is to generate bins considering data frequency that contain the same amount of instances, and this is actually how disctretization was done in all of the following experiments.

Lastly, when it comes to using the ruleset to predict a new set of instances, the rules are simply applied in order of addition and the first to activate is the one that determines the label from its consequent. In case no rule is fired, the majority label is predicted.

3 Evaluation

In this section we will analyze the results of applying the algorithm on three datasets from the UCI repository, namely: *hepatitis*. *cmc* and *nursery* [2]. It is worth mentioning that since some of these datasets contain missing values, some preprocessing steps were performed previously to running PRISM in which these values were filled with modes and means respectively to categorical and numerical attributes. By the way, data was randomly split into train and test sets with a proportion of 80/20.

3.1 Small dataset

The first dataset to apply the algorithm is the *hepatitis* dataset. This small dataset has only 155 instances, 6 numeric attributes, 13 categorical attributes and a class attribute with 2 different labels.

Dataset	# rules	# avg of selectors	Compute time	Accuracy
hepatitis	26	2.85	0.15s	0.871

Table 1: Classification results for the *hepatitis* dataset.

Table 1 shows a summary of the classification of the dataset and the performance of the algorithm. Additionally, since very few rules were produced in this case, these can be seen in listing 2 in the appendix section of the document. There, an interpetable list of rules can be observed, with the

attributes in the antecedent parts, the consequent, and the precision and recall scores (in that order) at the end of each rule.

As we can see, this method is able to achieve a relatively good performance on this dataset, since the number of rules and the average number of selectors are fairly low, and an accuracy of 0.871 is quite good. Even more, we must take into account the fact that this dataset contains 6 numeric attributes that had to be discretized. This is a handicap because in the discretization process some data is often lost, since we loose the ability to distinguish instances that initially were only different in the numeric attributes and that after discretizing them, they fall in the same bin so that they become duplicates. This loss is increased as lower is the number of bins, but in this case, the best results were achieved with 3 bins per attribute.

3.2 Medium dataset

The *cmc* dataset is a medium-sized database with 1473 instances. It has 2 numeric attributes, 7 categorical attributes and 3 possible class labels.

Dataset	# rules	# avg of selectors	Compute time	Accuracy
cmc	486	5.67	2.87s	0.5288

Table 2: Classification results for the *cmc* dataset.

A summary of the results of applying PRISM to this dataset can be seen in table 2. In this case, the numeric attributes were discretized into 5 bins. Seemingly, this dataset was harder to classify. The algorithm induced a total of 486 rules, a very high number considering that this amount is one third of the total amount of instances in the dataset. The average number of conditions in the antecedent part of these rules is also high, being 5.35. However, neither of these two factors helped the algorithm to achieve a decent accuracy, as only 0.5288 was scored.

This dataset is probably just difficult to classify with the attributes it provides. Despite, there could be a chance that the numeric attributes were actually very relevant but part of that information was lost during the discretization phase. In fact, if we take a look at the last rules (they are not listed in this document because the list is too large), they have a coverage or recall of 0.0034. This means that these rules probably cover just one single instance each, since that number is similar to $1/((1473 * 0.8)/3) = 0.0025$, the approximate proportion that an instance has within its class (we multiply by 0.8 because it is the size of the training set while 3 is the number of classes).

On the one hand, this dataset could be too hard to generalize or could have too much noise, as PRISM has to create very specific rules to be able to classify all the instances. On the other hand, this could be a result of one of the drawbacks of PRISM, which is the fact that each rule always has to have 100% precision, and this could penalize the generalization capability because rules are more fine grained, less flexible, and sometimes, too specific.

3.3 Large dataset

In the last experiment a larger dataset was employed, i.e. the *nursery* dataset. This database is formed by 12960 instances with 8 categorical attributes and 5 possible class labels, hence, it did not require any discretization.

Dataset	# rules	# avg of selectors	Compute time	Accuracy
nursery	539	5.36	8.18s	0.9792

Table 3: Classification results for the *nursery* dataset.

With an accuracy of 0.9792 (see table 3) we can say this is a proper scenario for this algorithm to shine. Here, no information was lost during the preprocessing of the dataset since all the attributes are categorical. The amount of rules (539) is significantly low compared to the number of instances (12960), thus, there was probably no overfitting. However, the average length of the antecedent is quite high (5.36) considering that 8 is the maximum. This could mean that there is not a dominant subset of attributes more correlated than the others, as most of the time many attributes were needed in order to achieve the 100% precision of the rules.

Other than that, the execution took notably longer compared to the previous experiments, but that is actually a decent time considering that this dataset is almost 10x the size of the *cmc* dataset.

4 Reproducibility

A simple CLI was built so as to be able to reproduce the experiments described in this document. Following are some examples of commands needed to run the algorithm on the different datasets:

Listing 1: Command examples to execute the experiments

```
$ python3 main.py --help
$ python3 main.py --dataset hepatitis
$ python3 main.py -d hepatitis
$ python3 main.py -d cmc
$ python3 main.py -d nursery
$ python3 main.py -d nursery --seed 123
```

References

[1] J. Cendrowska, "Prism: An algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, vol. 27, no. 4, pp. 349–370, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020737387800032

[2] Christopher L. DuBois, "UCI network data repository," http://networkdata.ics.uci.edu, 2008.

A Lists of rules

Listing 2: Rules induced from the small dataset

```
 1: IF (ascites == True) AND (sgot >= 43.0) AND (sgot < 86.0) THEN (class == die) [1.0,
     0.1739]
 2: IF (ascites == True) AND (bilirubin >= 1.4) AND (histology == True) THEN (class ==
    die) [1.0, 0.1739]
 3: IF (varices == True) AND (spiders == False) THEN (class == die) [1.0, 0.0435]
 4: IF (spiders == True) AND (alk_phosphate >= 81.0) AND (alk_phosphate < 103.06) AND (
    age >= 36.0) AND (age < 46.0) THEN (class == die) [1.0, 0.0435]
 5: IF (varices == True) AND (alk_phosphate < 81.0) THEN (class == die) [1.0, 0.0435]
 6: IF (spiders == True) AND (liver_firm == False) AND (alk_phosphate < 81.0) THEN (
    class == die) [1.0, 0.087]
 7: IF (bilirubin >= 1.4) AND (antivirals == True) AND (age >= 36.0) AND (age < 46.0)
    THEN (class == die) [1.0, 0.0435]
 8: IF (bilirubin >= 1.4) AND (spleen_palpable == True) AND (albumin < 3.8) THEN (class
     == die) [1.0, 0.1304]
 9: IF (varices == True) AND (sgot < 43.0) AND (spleen_palpable == False) THEN (class
    == die) [1.0, 0.087]
10: IF (bilirubin >= 1.4) AND (liver_big == False) AND (sgot >= 86.0) THEN (class ==
   die) [1.0, 0.087]
11: IF (anorexia == True) AND (malaise == False) THEN (class == die) [1.0, 0.0435]
12: IF (anorexia == True) AND (age >= 46.0) AND (liver_firm == False) AND (steroid ==
   False) THEN (class == die) [1.0, 0.0435]
13: IF (sex == male) THEN (class == live) [1.0, 0.1224]
14: IF (fatigue == False) AND (varices == False) THEN (class == live) [1.0, 0.3571]
15: IF (albumin >= 4.0) AND (spiders == False) THEN (class == live) [1.0, 0.2143]
16: IF (bilirubin >= 0.9) AND (bilirubin < 1.4) AND (age < 36.0) THEN (class == live)
   [1.0, 0.051]
17: IF (histology == False) AND (alk_phosphate >= 81.0) AND (alk_phosphate < 103.06)
   THEN (class == live) [1.0, 0.0612]
18: IF (liver_big == False) AND (sgot < 43.0) THEN (class == live) [1.0, 0.0408]
19: IF (ascites == False) AND (sgot >= 43.0) AND (sgot < 86.0) AND (protime >= 60.52)
   THEN (class == live) [1.0, 0.0408]
20: IF (bilirubin >= 0.9) AND (bilirubin < 1.4) AND (age >= 46.0) THEN (class == live)
   [1.0, 0.0306]
21: IF (fatigue == False) AND (age >= 36.0) AND (age < 46.0) THEN (class == live) [1.0,
     0.0102]
22: IF (albumin >= 3.8) AND (albumin < 4.0) AND (steroid == True) THEN (class == live)
   [1.0, 0.0102]
23: IF (age < 36.0) AND (malaise == False) THEN (class == live) [1.0, 0.0204]
24: IF (anorexia == True) AND (alk_phosphate < 81.0) THEN (class == live) [1.0, 0.0102]
25: IF (albumin >= 3.8) AND (albumin < 4.0) AND (anorexia == True) AND (malaise == True
   ) THEN (class == live) [1.0, 0.0204]
26: IF (age < 36.0) AND (steroid == True) AND (liver_firm == False) THEN (class == live
   ) [1.0, 0.0102]
```