Lazy classification algorithm

Introduction

In this work, 4 algorithms for lazy classification of objects represented by binary features. For non-binary data, scaling is implemented. The algorithms were tested on Tic-Tac-Toe data. The best algorithm result shows:
{'accuracy': 0.9692307692307692,
'precision': 0.9682539682539683,
'recall': 1.0,
'f1': 0.9838709677419354,
'TN_Pred_Rate': 1.0,
'FP_Rate': 0.9375,
'FN_Rate': 0.0,
'FDISC_Rate': 0.32967032967032966}

Algorithm 1

This algorithm computes the intersection of the description of the classified object and objects from plus and minus contexts. Then the falsifiability of the hypothesis is checked:
- each of the objects from the plus-context counts as a positive result if its intersection with the classified object is not included in the description from the minus-context.
- each of the objects from the minus-context counts as a negative result if its intersection with the classified object is not included in the description from the plus-context
Falsified hypothesis don't count.
As a result, the decision is made by the simple majority method.

Algorithm 2

In this algorithm, unlike the previous one, for each example its support in plus and minus contexts is considered.
As a result, a class is selected that matches the context with more support.

Algorithm 3

In this algorithm, unlike the first one, falsified hypotheses can participate in counting, but only if their support is greater than closure in the opposite context. It also introduces a restriction on the cardinality of the

intersection - the intersection of the classified object and the plus or minus context must include no less than C* 100% features.

Algorithm 4
This algorithm is a compilation of the second and third algorithms; in addition to checking that support is greater than closure in the opposite context, the difference between them is also taken into account when voting. It is also checked that the possibility of intersection must be greater than the constant C.

Finding the optimal value of the parameter
For the 3rd algorithm:

C	Precision	Recall	F1	Accuracy
0.5	0.9870	0.9952	0.9907	0.9272
0.55	0.9875	1.0	0.9934	0.9288
0.6	0.9668	0.9644	0.9624	0.8336
0.65	0.9087	0.9477	0.9239	0.8110
0.7	0.8852	0.9469	0.9118	0.8115

For the 4th algorithm, the results are almost identical. As a result, relying on such metrics as F1 and Accuracy, it was decided to take the constant C = 0.55.

Results
As a result, for the data train1 and test1 from Tic-Tac-Toe (the final testing was carried out only on them due to the computational complexity of the algorithms), the following results were obtained:

Algorithm	Precision	Recall	F1	Accuracy

1	0.7922	1.0	0.8841	0.8280
2	0.9686	1.0	0.9839	0.9692
3	0.8906	0.9661	0.9268	0.8205
4	0.6761	0.8571	0.7559	0.6396

As you can see, you can , tha the 2nd algorithm showed the best result.
Result as below:
{'accuracy': 0.9692307692307692,
'precision': 0.9682539682539683,
'recall': 1.0,
'f1': 0.9838709677419354,
'TN_Pred_Rate': 1.0,
'FP_Rate': 0.9375,
'FN_Rate': 0.0,
'FDISC_Rate': 0.32967032967032966}