

Ordered sets for Data Analysis

Aleksey Ryabykin

12/12/22



Datasets

I have chosen three datasets to check the base algorithm and my proposed one performances. All of them belong to the UCI Machine Learning Repository:

- Car dataset:

Instances: 1728 Features: 7

- Mushrooms dataset:

Instances: 8124 Features: 23

- Congressional voting records:

Instances: 435 Features: 17



Car Dataset



Mushrooms dataset



Congress Dataset

Initial algorithm

Assume that we want to make a prediction for description $x \subseteq M$ given the set of training examples $X_{\text{train}} \subseteq 2^M$ and the labels $y_x \in \{\text{False}, \text{True}\}$, corresponding to each $x \in X_{\text{train}}$.

First, we split all examples X_{train} to positive X_{pos} and negative X_{neg} examples:

$$X_{\text{pos}} = \{x \in X_{\text{train}} \mid y_x \text{ is True}\}, \quad X_{\text{neg}} = X \setminus X_{\text{pos}}.$$

To classify the description x we follow the procedure:

1. Count the number of counterexamples for positive examples:

For each positive example $x_{\text{pos}} \in X_{\text{pos}}$ we compute the intersection $x \cap x_{\text{pos}}$. Then, we count the counterexamples for this intersection, that is the number of negative examples $x_{\text{neg}} \in X_{\text{neg}}$ containing intersection $x \cap x_{\text{pos}}$.

2. Dually, count the number of counterexamples for negative examples.

Finally, we compare the average number of counterexamples for positive and negative examples. We classify as being positive if the number of counterexamples for positive examples is smaller the one for negative examples.

Proposed improvements to the initial algorithm

The core idea is the same as for the initial algorithm. But the chosen path to calculate the intersections is different with the previous one. The procedure can be expressed in terms of matrix multiplication of bitmasks that had been obtained after binarization.

Also, there is some bad influence of proposed in the initial algorithm normalization of counts of counterexamples.

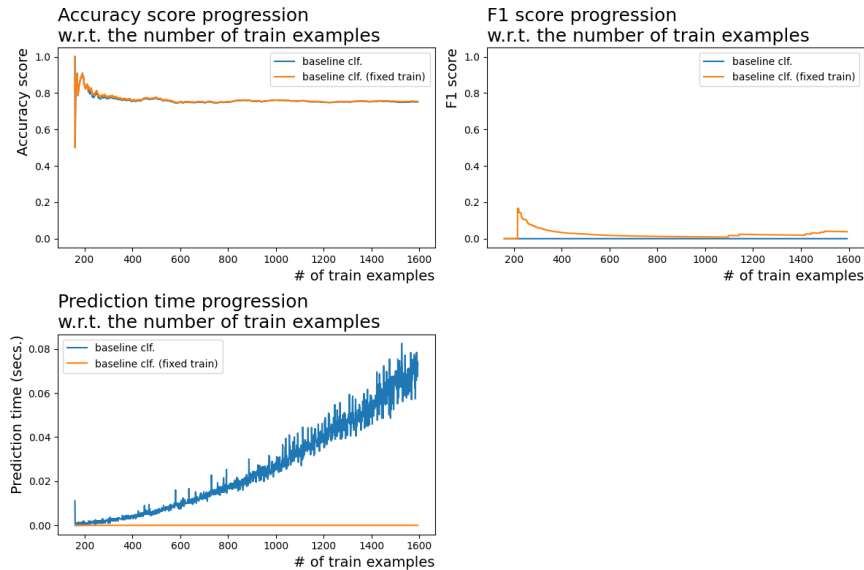
27 lines - 27 Removals

```
1 X_pos = [x_train for x_train, y in zip(X_train, Y_train)
2         if y]
3 X_neg = [x_train for x_train, y in zip(X_train, Y_train)
4         if not y]
5 n_counters_pos = 0 # number of counter examples for
6 positive intersections
7 for x_pos in X_pos:
8     intersection_pos = x & x_pos
9     if len(intersection_pos) < min_cardinality: # t
10        he intersection is too small
11        continue
12
13    for x_neg in X_neg: # count all negative exampl
14        es that contain intersection_pos
15        if (intersection_pos & x_neg) == intersectio
16        n_pos:
17            n_counters_pos += 1
18
19    n_counters_neg = 0 # number of counter examples for
20    negative intersections
21    for x_neg in X_neg:
22        intersection_neg = x & x_neg
23        if len(intersection_neg) < min_cardinality:
24            continue
25
26    for x_pos in X_pos: # count all positive exampl
27        es that contain intersection_neg
28        if (intersection_neg & x_pos) == intersectio
29        n_neg:
30            n_counters_neg += 1
31
32    perc_counters_pos = n_counters_pos / len(X_pos)
33    perc_counters_neg = n_counters_neg / len(X_neg)
34
35    prediction = perc_counters_pos < perc_counters_neg
36    return prediction
```

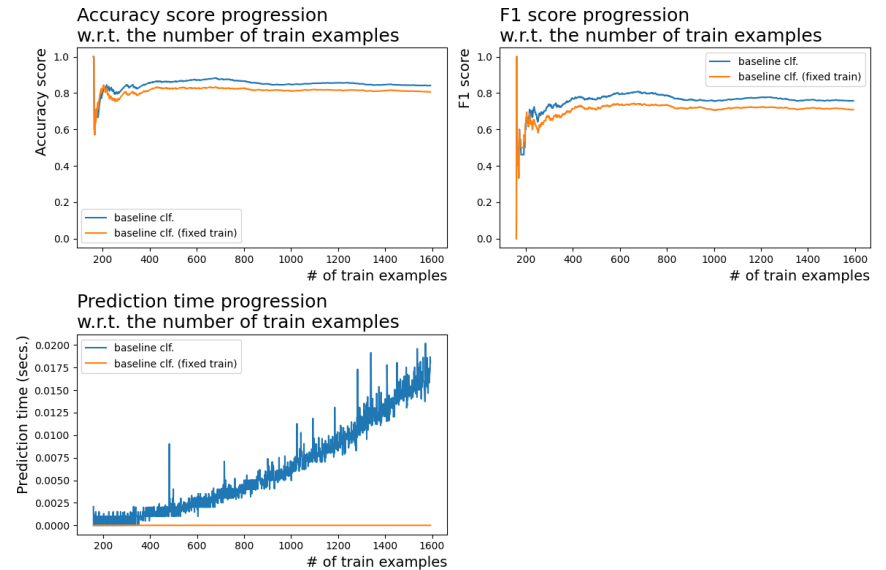
13 lines + 14 Additions

```
1 X_pos = np.array(X_train[np.array(Y_train, dtype=bool)],
2                 dtype=int)
3 X_neg = np.array(X_train[~np.array(Y_train, dtype=boo
4                 l)], dtype=int)
5
6 idxs = X_pos @ x >= min_cardinality
7 n_counters_pos = ((X_neg @ (X_pos & x)[idxs, :].T) ==
8                 (X_pos @ x)[idxs]).sum()
9
10 idxs = X_neg @ x >= min_cardinality
11 n_counters_neg = ((X_pos @ (X_neg & x)[idxs, :].T) ==
12                 (X_neg @ x)[idxs]).sum()
13
14 perc_counters_pos = n_counters_pos / (n_counters_neg
15 + n_counters_pos)
16 perc_counters_neg = n_counters_neg / (n_counters_pos
17 + n_counters_neg)
18
19 prediction = perc_counters_pos < perc_counters_neg
20 return prediction
```

Comparison



Initial algorithm



Improved algorithm

Results (Accuracy scores)

Data	Catboost CLF	DecisionTree CLF	RandomForest CLF	XBGRF CLF	LGBM CLF	Initial Algorithm	Improved Algorithm
Car	0.958	0.816	0.743	0.796	0.813	0.78	0.78
Mush rooms	1	0.926	0.917	0.932	0.927	0.93	0.93
Congress	0.952	0.966	0.961	0.971	0.957	0.7	0.92

Results (F1 scores)

Data	Catboost CLF	DecisionTree CLF	RandomForest CLF	XBGRF CLF	LGBM CLF	Initial Algorithm	Improved Algorithm
Car	0.921	0.838	0.829	0.844	0.865	0.18	0.78
Mush rooms	1	0.937	0.930	0.944	0.941	0.9	0.91
Congress	0.949	0.965	0.961	0.970	0.957	0.66	0.9

Results (Elapsed time)

Data	Catboost CLF	DecisionTree CLF	RandomForest CLF	XBGRF CLF	LGBM CLF	Initial Algorithm	Improved Algorithm
Car	16s	90ms	480ms	220ms	75ms	839ms	560ms
Mush rooms	17s	440ms	850ms	670ms	156ms	3min 13s	43.5s
Congress	1.1s	70ms	360ms	90ms	65ms	303ms	58ms

Results

Code reduction

$\approx 50\%$

Chosen datasets

3

Time improvement

$\approx 6.5x$

Average quality improvement

$\approx 23\%$

Number of comparing models

5

Pattern structures

0

Inferences



Little data

0110
1001
1010

Binary



Interpretable