# Ordered Sets for Data Analysis.
# Homework project in FCA: report

O.R.Don

December 12, 2019

## Lazy FCA classification algorithm

Given dataset divided into positive and negative class objects. Algorithm takes a set of unclassified objects and tries to determine whether new object is in positive or negative class.

Algorithm:

0. Getting intents of all positive and negative objects;
1. For every unclassified object get its intent;
2. Intersect unclassified intent with all positive and negative intents;
3. If unclassified has non empty intersections only with positive intents, then it is marked as positive, if it has non empty intersections only with negative intents, it is marked as negative, else it is marked as contradictory.

During the research, there was written python script for testing lazy FCA classification algorithm. This script was tested on the test data of tic-tac-toe. The algorithm has pour results, since only less than 20% of objects were classified correctly.

## Algorithm modification 1

For enhancing the classification accuracy, some modifications were made. On the $2^{nd}$ step of the algorithm there was added counting ratio of non empty intersections with positive objects to the cardinal of all positive objects:

$$u = \text{unclassified object}$$
$$G_+ = \text{set of positive training objects}$$
$$G_- = \text{set of negative training objects}$$
$$pos = \frac{|\{g | g \in G_+, u \cap g \neq \emptyset\}|}{|G_+|}$$
$$neg = \frac{|\{g | g \in G_-, u \cap g \neq \emptyset\}|}{|G_-|}.$$

After intersections there was added an aggregation function:

$$aggr(pos, neg) = \frac{pos + 1}{neg + 1}.$$

The threshold for the aggregation function was set to 1, which means "if ratio of positive intersections count is greater than or equal to negative intersections count, then the object is marked as positive, else as negative".

This modification was enough to enhance accuracy for test data to 95%.

# Dataset

Dataset chosen for the project was taken from UCI ML Repository (link to the dataset). There are two sets with grades in math and Portuguese. Math dataset was taken for the project. It consists of data about students from two schools, there are categorical and numerical features. The chosen target feature is average score of student. To be more exact, object has positive class if its average score is greater than or equal to some value, otherwise it has negative class.

It was decided to make binarization of data for classification. All categorical features were translated to format $' < categorytitle >:< categoryvalue >'$. All numerical features which are represented as disjoint intervals were treated as categorical features (e.g. traveltime - home to school travel time: 1 - ¡15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - ¿1 hour). Numerical features which are not disjoint are 'Medu', 'Fedu', 'failures', 'absences'. The first three of them were translated by the rule:

$$\forall a((a > b \vee a = \text{'x'}) \rightarrow b = \text{'x'}),$$

where $a$ is binarized numerical feature.

The only left feature 'absences' was translated by the intervals: [0;1), [1;2), [2;3), [3;6) [6;max). This fragmentation was achieved experimentally as the best for classification.

# Algorithm modification 1 experiments

On the chosen dataset the algorithm worked much worse, than for the test data. It turned out, that almost all students has similar attributes, so the aggregation function has been given 1 for almost all unclassified objects, which means that by such aggregation positive and negative objects are indistinguishable.

# Algorithm modification 2

Since chosen aggregation works bad with the chosen dataset, it was decided to add some new metrics. On the $2^{nd}$ step of the algorithm, there was added counting of amount of attributes in the intersection. After that this amount was divided by amount of all attributes and cardinal of positive or negative objects set:

$$attr(o) = \text{amount of attributes in o}$$
$$pos\_attr = \frac{\sum_{g \in G_+} attr(u \cap g)}{|G_+|}$$
$$neg\_attr = \frac{\sum_{g \in G_-} attr(u \cap g)}{|G_-|}.$$

And the aggregation function was transformed to:

$$aggr(pos, neg, pos\_attr, neg\_attr) = \frac{pos + 1}{neg + 1} \cdot \frac{pos\_attr + 1}{neg\_attr + 1}.$$

# Algorithm modification 2 experiments

Experiments were using different amount of features and different values for average score: 0.4, 0.5, 0.53 (average score of all students), 0.8. Worst results

were achieved for score 0.8, best for 0.4, results for 0.5 and 0.53 are almost the same.

For testing algorithm was used k fold cross validation, with k=4.

For each run of the algorithm data was randomly recomposed.

There were made experiments with different threshold values for aggregation function with rule:

$$aggr \geq threshold \Rightarrow \text{ positive class}$$
$$aggr < threshold \Rightarrow \text{ negative class.}$$

The worst results were achieved on dividing students by average score 0.8. Best threshold for aggregation is 1.025. Precision less than 20%, recall around 60%, accuracy around 60% and F1 score less than 20%. This can be explained as there not many students who got average score greater than 0.8 and all students share to much similar attributes.

The best results were achieved on dividing students by average score 0.4. Best threshold for aggregation is 1.005. Precision almost 80%, recall more than 90%, accuracy more than 75%, F1 score around 85%. This can mean that students with worst results have more common attributes with each other.

The base average score for dividing student was 0.53 which is an average score of all students. For this score the best threshold for aggregation is 1.015. Precision is around 60%, recall is greater than precision and mostly stays in interval between 65% to 75%, accuracy around 60% and F1 score is a bit greater than accuracy, mostly on 3-5%. This can mean that students who get score more than average and those who get less, do not differ to much from each other.

# Comparison with classical algorithms

For productivity comparison there were made experiments using:

1. Decision tree
2. K nearest neighbours
3. Naive Bayes
4. Support vector classification (SVC)

There were made the same experiments using k fold cross validation with k=4 and differentiating between binarized, non binarized data and student score threshold 0.4, 0.53, 0.8.

| Results comparison | | | | |
|---|---|---|---|---|
| Algorithm and parameters | Accuracy | Precision | Recall | F1 Score |
| LazyFCA; thresh: 0.4 bin | 0.772 | 0.829 | 0.879 | 0.853 |
| LazyFCA; thresh: 0.53 bin | 0.625 | 0.634 | 0.672 | 0.651 |
| LazyFCA; thresh: 0.8 bin | 0.661 | 0.115 | 0.608 | 0.137 |
| DT; thresh: 0.4 non bin | 0.734 | 0.852 | 0.808 | 0.827 |
| DT; thresh: 0.53 non bin | 0.610 | 0.646 | 0.578 | 0.605 |
| DT; thresh: 0.8 non bin | 0.879 | 0.000 | 0.000 | 0.000 |
| DT; thresh: 0.4 bin | 0.757 | 0.851 | 0.843 | 0.845 |
| DT; thresh: 0.53 bin | 0.593 | 0.622 | 0.588 | 0.599 |
| DT; thresh: 0.8 bin | 0.896 | 0.104 | 0.071 | 0.084 |
| KNN; thresh: 0.4 non bin | 0.823 | 0.844 | 0.955 | 0.895 |
| KNN; thresh: 0.53 non bin | 0.562 | 0.577 | 0.612 | 0.589 |
| KNN; thresh: 0.8 non bin | 0.927 | 0.000 | 0.000 | 0.000 |
| KNN; thresh: 0.4 bin | 0.805 | 0.834 | 0.942 | 0.885 |
| KNN; thresh: 0.53 bin | 0.562 | 0.577 | 0.612 | 0.589 |
| KNN; thresh: 0.8 bin | 0.927 | 0.000 | 0.000 | 0.000 |
| NB; thresh: 0.4 non bin | 0.805 | 0.837 | 0.939 | 0.884 |
| NB; thresh: 0.53 non bin | 0.638 | 0.612 | 0.844 | 0.707 |
| NB; thresh: 0.8 non bin | 0.689 | 0.200 | 0.402 | 0.175 |
| NB; thresh: 0.4 bin | 0.276 | 0.989 | 0.090 | 0.148 |
| NB; thresh: 0.53 bin | 0.638 | 0.612 | 0.844 | 0.707 |
| NB; thresh: 0.8 bin | 0.246 | 0.084 | 0.933 | 0.154 |
| SVC; thresh: 0.4 non bin | 0.792 | 0.803 | 0.978 | 0.882 |
| SVC; thresh: 0.53 non bin | 0.615 | 0.593 | 0.845 | 0.707 |
| SVC; thresh: 0.8 non bin | 0.927 | 0.000 | 0.000 | 0.000 |
| SVC; thresh: 0.4 bin | 0.810 | 0.822 | 0.971 | 0.890 |
| SVC; thresh: 0.53 bin | 0.615 | 0.593 | 0.845 | 0.695 |
| SVC; thresh: 0.8 bin | 0.927 | 0.000 | 0.000 | 0.000 |

On the table we can see that best classification is provided by Naive Bayes (except for binarized case with student score threshold 0.8). In all other cases LazyFCA works slightly better than Decision Tree, K nearest neighbours and SVC

(in case of student score threshold 0.8 all of these three has awful precision and recall scores).