Project Report on the Ordered Sets in Data Analysis

Edgar Zakharyan

Moscow, 2020

1 FCA algorithm for test data

I used tic-tac-toe dataset for testing my FCA lazy algorithm. It consists of 10 columns, first 9 of which are used as training data and last one is target ("positive"or "negiatove"). All values in columns are binary. There is a train and test datasets. Here is a head of scaled train dataset with all categorical features:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	0	0	1	0	0	1
1	1	1	1	1	0	0	0	0	1	1
2	1	1	1	1	0	0	0	0	0	1
3	1	1	1	1	0	0	0	0	0	1
4	1	1	1	1	0	0	0	0	0	1

1.1 FCA lazy algorithms:

First algorithm:

The algorithm is based on the sum of the intersection of the features of an unknown type with examples with plus and minus context. When number of intersections with plus context with our example divided to number of plus context is higher, than number of intersections with minus context divided to number of minus context with our example, we classify our example object as positive one. And vice versa.

$$positive = \sum_{i \in G^+} |g^{'} \cap g_i^+|$$

$$negative = \sum_{i \in G^-} |g^{'} \cap g_i^-|$$

where G^+ is positive context, G^i is negative context.

Second algorithm:

The algorithm is based on the **normalized** sum of the intersection of the features of an unknown type with examples with plus and minus context. When number of intersections with plus context with our example divided to number of plus context is higher, than number of intersections with minus context divided to number of minus context with our example, we classify our example object as positive one. And vice versa.

$$positive = \frac{1}{|G^+|} \sum_{i \in G^+} |g^{'} \cap g_i^+|$$

$$negative = \frac{1}{|G^-|} \sum_{i \in G^-} |g^{'} \cap g_i^-|$$

where G^+ is positive context, G^i is negative context.

Third algorithm:

Here we calculate the vote in proportion to the "relative power of intersection"of the intersection which are signs of the unknown example and (+) example with the rest (+) - examples, if this intersection is embedded in them. And vice versa.

$$positive = \frac{1}{|G^+|} \sum_{i \in G^+} \frac{1}{|g_i^+||G^+|} |(g^{'} \cap g_i^+) \cap g_i^+|$$

$$negative = \frac{1}{|G^-|} \sum_{i \in G^-} \frac{1}{|g_i^-||G^-|} |(g^{'} \cap g_i^-) \cap g_i^-|$$

where G^+ is positive context, G^i is negative context.

Performance of each algorithm was measured by those metrics:

- True Positive
- True Negative
- False Positive
- False Negative
- True Positive Rate
- True Negative Rate

- Negative Predictive Value
- False Positive Rate
- False Discovery Rate
- Accuracy
- Precision
- Recall

Here are results of our algorithms for tic-tac-toe dataset:
First algorithm:

- True Positive: 61
- True Negative: 0
- False Positive: 32
- False Negative: 0
- True Positive Rate: 1.0
- True Negative Rate: 0.0
- Negative Predictive Value: nan
- False Positive Rate: 1.0
- False Discovery Rate: 0.34408602150537637
- Accuracy: 0.6559139784946236
- Precision: 0.6559139784946236
- Recall: 1.0

Second algorithm:

- True Positive: 34
- True Negative: 24
- False Positive: 8
- False Negative: 27
- True Positive Rate: 0.5573770491803278
- True Negative Rate: 0.75
- Negative Predictive Value: 0.47058823529411764

- False Positive Rate: 0.25
- False Discovery Rate: 0.19047619047619047
- Accuracy: 0.6236559139784946
- Precision: 0.8095238095238095
- Recall: 0.5573770491803278

Third algorithm:

- True Positive: 55
- True Negative: 32
- False Positive: 0
- False Negative: 6
- True Positive Rate: 0.9016393442622951
- True Negative Rate: 1.0
- Negative Predictive Value: 0.8421052631578947
- False Positive Rate: 0.0
- False Discovery Rate: 0.0
- Accuracy: 0.9354838709677419
- Precision: 1.0
- Recall: 0.9016393442622951

For comparison with state of the art algorithms, I decided to choose KNN and Logistic regression algorithms:

Comparing with KNN algorithm (state of the art algorithm):

- True positive: 60
- True Negative: 26
- False Positive: 6
- False Negative: 1
- True Positive Rate: 0.9836065573770492
- True Negative Rate: 0.8125
- Negative Predictive Value: 0.9629629629629629

- False Positive Rate: 0.1875
- False Discovery Rate: 0.09090909090909091
- Accuracy: 0.9247311827956989
- Precision: 0.9090909090909091
- Recall: 0.9836065573770492

Comparing with Logistic regression algorithm:

- True positive: 58
- True Negative: 28
- False Positive: 5
- False Negative: 2
- True Positive Rate: 0.953603117045
- True Negative Rate: 0.8045
- Negative Predictive Value: 0.925629629
- False Positive Rate: 0.1844
- False Discovery Rate: 0.089140808
- Accuracy: 0.91423523
- Precision: 0.903414241
- Recall: 0.953241550240

2 My own dataframe

For my own research I used the dataset using a mobile app called ASDTests (ASDtests.com) to screen autism in toddlers. `https://www.kaggle.com/fabdelja/autism-screening-for-toddlers`

I used those binary features:

Variable in Dataset	Corresponding Q-chat-10-Toddler Features
A1	Does your child look at you when you call his/her name?
A2	How easy is it for you to get eye contact with your child?
A3	Does your child point to indicate that s/he wants something? (e.g. a toy that is out of reach)
A4	Does your child point to share interest with you? (e.g. poin9ng at an interes9ng sight)
A5	Does your child pretend? (e.g. care for dolls, talk on a toy phone)
A6	Does your child follow where you're looking?
A7	If you or someone else in the family is visibly upset, does your child show signs of wan9ng to comfort them? (e.g. stroking hair, hugging them)
A8	Would you describe your child's first words as:
A9	Does your child use simple gestures? (e.g. wave goodbye)
A10	Does your child stare at nothing with no apparent purpose?

And also those:

- Jaundice. Whether the case was born with jaundice
- Family_mem_with_ASD. Whether any immediate family member has a PDD
- Age_Mons_(11.999, 24.0). Age in months
- Age_Mons_(24.0, 34.0). Age in months
- Age_Mons_(34.0, 36.0). Age in months

As can be seen for age in months I used quantile scaling.
Target value was **Class**, describing is there autism in certain case or not.

Here is a head of dataframe that I used:

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Sex	Jaundice	Family_mem_with_ASD	Age_Mons_(11.999, 24.0]	Age_Mons_(24.0, 34.0]	Age_Mons_(34.0, 36.0]	Class
0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	1	0	negative
1	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	positive
2	1	0	0	0	0	0	1	1	0	1	1	1	0	0	0	1	positive
3	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	positive
4	1	1	0	1	1	1	1	1	1	1	0	0	1	1	0	0	positive
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1049	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	negative
1050	0	0	1	1	1	0	1	0	1	0	1	1	0	1	0	0	positive
1051	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	positive
1052	1	0	0	0	0	0	0	1	0	1	1	0	1	1	0	0	negative
1053	1	1	0	0	1	1	0	1	1	0	1	1	1	1	0	0	positive

Here are results of applying my algorithms to this data:
First algorithm:

- True Positive: 360
- True Negative: 0
- False Positive: 167
- False Negative: 0
- True Positive Rate: 1.0
- True Negative Rate: 0.0
- Negative Predictive Value: nan
- False Positive Rate: 1.0
- False Discovery Rate: 0.31688804554079697
- Accuracy: 0.683111954459203
- Precision: 0.683111954459203
- Recall: 1.0

Second algorithm:

- True Positive: 290
- True Negative: 167
- False Positive: 0
- False Negative: 70
- True Positive Rate: 0.8055555555555556

- True Negative Rate: 1.0
- Negative Predictive Value: 0.7046413502109705
- False Positive Rate: 0.0
- False Discovery Rate: 0.0
- Accuracy: 0.8671726755218216
- Precision: 1.0
- Recall: 0.8055555555555556

Third algorithm:

- True Positive: 286
- True Negative: 167
- False Positive: 0
- False Negative: 74
- True Positive Rate: 0.8188888811889
- True Negative Rate: 1.0
- Negative Predictive Value: 0.7786848072562358
- False Positive Rate: 0.0
- False Discovery Rate: 0.0
- Accuracy: 0.88007590132827327
- Precision: 1.0
- Recall: 0.8988888888888889

State of the art algorithms results:
Comparing with KNN algorithm (state of the art algorithm):

- True positive: 338
- True Negative: 161
- False Positive: 6
- False Negative: 22
- True Positive Rate: 0.9388888888888889
- True Negative Rate: 0.9640718562874252

- Negative Predictive Value: 0.8797814207650273
- False Positive Rate: 0.03592814371257485
- False Discovery Rate: 0.01744186046511628
- Accuracy: 0.9468690702087287
- Precision: 0.9825581395348837
- Recall: 0.9388888888888889

Comparing with Logistic regression algorithm:

- True positive: 334
- True Negative: 165
- False Positive: 5
- False Negative: 23
- True Positive Rate: 0.978882348852
- True Negative Rate: 0.9540712141
- Negative Predictive Value: 0.9411421412412
- False Positive Rate: 0.031424250992222111
- False Discovery Rate: 0.010943222
- Accuracy: 0.98214111491
- Precision: 0.9894124445
- Recall: 0.9690124001201

3 Conclusion

As can be seen, the most efficient FCA algorithm is third one with "relative power of intersection". However, state of the art algorithms K-means and logistic regression showed better results. Nevertheless, all algorithms used in this work, besides of first, which was the simplest FCA algorithm, showed very high results in all metrics for both test dataset tic-tac-toe and my own one with data about toddlers autism.