Data Mining Assignment V

Fang Zhou

1. Introduction

In data mining area, except the algorithms we have learnt, there is still one representative set classification algorithms, named Classification By Association (CBA) classifier. In this assignment, I will test the efficacy and efficiency of CBA classifier, especially for the apriori algorithm. I will also compare the performance of apriori with the classification algorithms we used in the assignment III, which includes K-nearest-neighbors algorithm, decision tree algorithm, and naive Bayesian algorithm.

2. Methods

2.1 Article Feature

In this assignment, I classify documents based on the word frequency. Word frequency is collect the number of one word appears in the article. It is a very important feature for an article. I use the most frequent N words in the whole corpus to create the feature vectors for the articles. If the article has one of the most frequent N words, then I will add this word into this article's feature vectors. So the value domain in the feature vector is string, not an integer or a float like previous assignment.

2.2 Data Pre-processing

However, the vectors collected in the last assignment don't fit well for this assignment. For example, some articles may miss the topic part so that it cannot be classified into any categories. I pre-process the data following the rules shown below:

- If an article does not have the body part or its body part is empty, then skip this article.
- If an article does not have the topic part or its topic part is empty, then skip this article.
- If an article has more than one topic, then random pick up one topic for this article.

After this processing, the number of articles decreases to 8654.

Each article is bonded with only one topic. This processing makes the future processing easy and correct. Figure I shows the distribution of topics after preprocessing. It should be noted that I remove the topics with the number less than 30 and sum them up into one categories, others, in Figure I, because of the limited space.

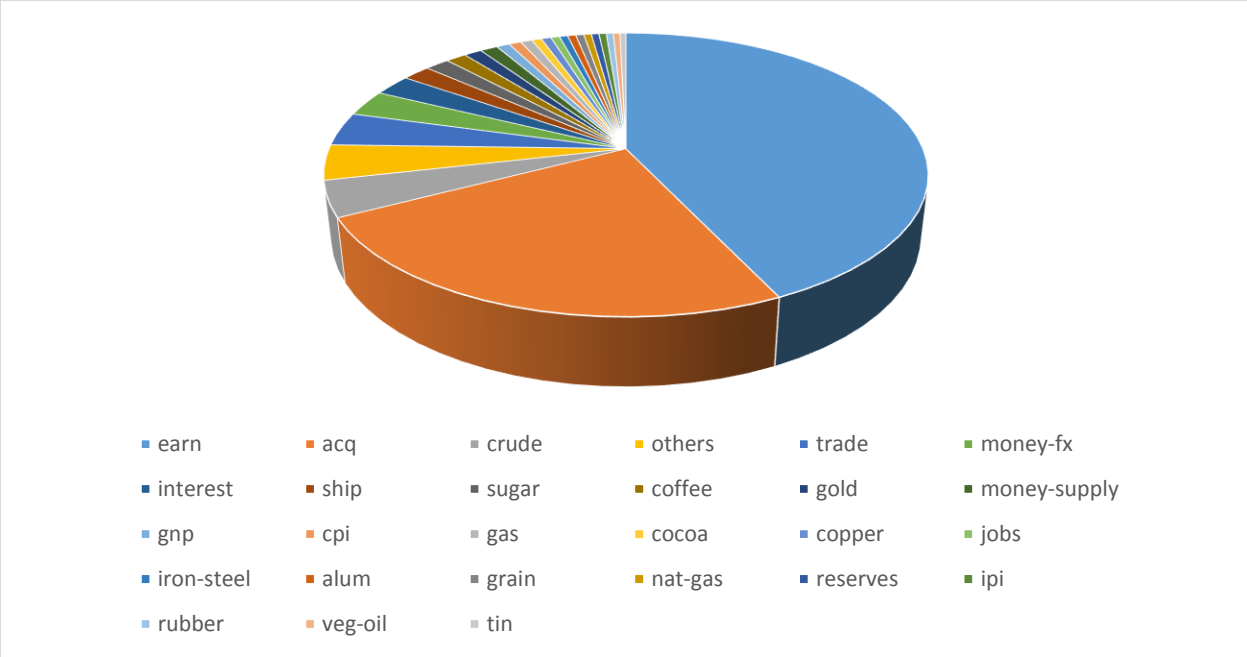

Figure I: The distribution of topics

As we can see, there are two main topics in the corpus after processing. They are “earn” and “acq”, which occupies 43.16% and 24.56%. The input is not very uniform so that it may lead the prediction obviates to the two most topics.

2.3 CBA Classification Algorithm

There are several famous CBA classification algorithms. In this assignment, I choose to use the most famous algorithm, apriori. The main idea of apriori is to use the frequent itemsets to generate association rules. Then use these rules to classify the feature vectors. To generate rules, we should provide the support value and confidence value to the classifier in the course of training. In general, these two values have direct influence on the quality of rules for the classifier, except the feature.

3. Experiments

3.1 Testbed

I test the program on a single machine. The machine has a 2.4 GHz 6-core Intel Xeon CPU E5-2630, 24 GB memory and one 500 GB Western Digital SATA hard drives.

3.2 Experiments Details

I first extract the feature vector for each document with n (n = 1000, 2000, 3000, 4000, and 5000) dimensions. Then I randomly shuffle the feature vectors and split them with 8:2 decoupling. In class, the instructor mentioned that people usually use single digit value for support and confidence in practical. Based on this experience, I prepare 5 values (2, 3, 5, 7, and 10) for support value and confidence value. So there are 25 probabilities for support value and confidence value, such as s=2 and c=10. It should be

noted that the values I show here are in a percent scope. For example, s=2 means support value is equal to 2%.

I run apriori approach with different support and confidence values working on different training datasets generated above. After generating rules, I rank these rules and classify the testing datasets with these rules.

I run each test for 10 times and calculate the average value of them. All the results are the average value without special notifications.

3.3 Implementation

In this assignment, I use Borgelt's apriori executable file, which we discussed and demonstrated in the class, to mine association rules (http://www.borgelt.net/apriori.html). I also submit this program to the server. It should be noted that if there is no rule working for one article, this article will be classified to the most frequency class, earn.

However, the rules cannot be directly used to classify the document. So I implement several programs to rank the rules, classify the documents, and collect the statistical data, including record the online time, offline time, and accuracy.

4. Results

4.1 Efficiency

First, I will show the offline cost of the apriori algorithm. The amount of data in the training datasets with different dimension is the same, so I use the total running time instead of the Table I shows the running time of apriori algorithm. The unit is second.

Table I: Offline cost (second) of the apriori algorithm

Offline	1000	2000	3000	4000	5000
s=2,c=2	22.136	28.382	38.098	36.339	42.57
s=2,c=3	16.15	19.666	17.363	22.823	19.432
s=2,c=5	16.172	16.378	24.817	16.418	19.842
s=2,c=7	11.417	11.219	15.306	14.783	14.017
s=2,c=10	8.399	9.12	11.791	10.032	10.309
s=3,c=2	6.659	6.697	7.913	8.102	7.867
s=3,c=3	5.235	6.624	7.202	6.947	8.25
s=3,c=5	5.23	5.897	5.708	5.451	5.003
s=3,c=7	4.207	4.437	5.173	4.833	3.986
s=3,c=10	3.077	2.807	4.516	3.406	3.262
s=5,c=2	1.575	2.033	2.312	2.004	2.233
s=5,c=3	1.323	1.843	1.735	1.772	1.89
s=5,c=5	0.99	1.285	1.425	1.208	1.315
s=5,c=7	0.894	0.933	0.831	1.118	1.06

s=5,c=10	0.572	0.607	0.631	0.813	0.848
s=7,c=2	0.594	0.713	0.711	0.952	0.831
s=7,c=3	0.603	0.66	0.632	0.659	0.698
s=7,c=5	0.4	0.477	0.425	0.467	0.387
s=7,c=7	0.383	0.342	0.387	0.385	0.434
s=7,c=10	0.24	0.253	0.255	0.264	0.288
s=10,c=2	0.256	0.271	0.316	0.322	0.323
s=10,c=3	0.178	0.241	0.222	0.219	0.235
s=10,c=5	0.158	0.19	0.202	0.205	0.211
s=10,c=7	0.145	0.144	0.147	0.182	0.164
s=10,c=10	0.114	0.13	0.138	0.151	0.159

At first sight, it seems that there are no rules for the offline cost. However, when I check the number of rules generated by each test, I find that the offline cost is proportional to the number of rules in most cases. The numbers of generating rules are shown in Table II. For examples, in the third row of Table II, the number of rules for the vector with 3000 features is larger than the one of the vector with 5000 features. Then look back to Table I and we can find the offline cost of the vector with 1000 features is larger than the one with 5000 features.

Table II: Number of rules generated by the apriori algorithm

Rules	1000	2000	3000	4000	5000
s=2,c=2	299386	311115	380759	379775	343717
s=2,c=3	210450	210404	182377	231743	204349
s=2,c=5	171180	133837	227538	146115	173882
s=2,c=7	131968	119401	147331	146658	140780
s=2,c=10	103542	100971	134060	105417	120102
s=3,c=2	81654	71307	75283	83115	73648
s=3,c=3	56705	71898	55418	62905	65775
s=3,c=5	48922	49148	49285	49240	44592
s=3,c=7	43380	40788	46679	42989	37159
s=3,c=10	31702	32101	43520	36870	35722
s=5,c=2	16727	17043	18725	16285	17002
s=5,c=3	11661	14476	14197	13109	13852
s=5,c=5	9459	10211	12085	9875	10715
s=5,c=7	8454	8793	8430	10303	8952
s=5,c=10	6674	6460	6717	7834	7951
s=7,c=2	6503	6028	5990	6828	6323
s=7,c=3	5222	4989	5318	4822	5165
s=7,c=5	4013	4377	3634	3913	3302
s=7,c=7	3717	3243	3488	3285	3538
s=7,c=10	2924	2619	2772	2656	2831
s=10,c=2	2572	2318	2429	2510	2416

s=10,c=3	1802	2031	1681	1667	1631
s=10,c=5	1318	1402	1521	1488	1323
s=10,c=7	1253	1130	1171	1210	1010
s=10,c=10	1029	1056	1030	1094	975

Another interesting observation is the larger s and larger c will lead to less number of rules. This is very easy to understand because the test with larger s and c values can remove more features in each loop than the one with smaller s and c.

Table III shows the online cost of apriori algorithm. Because the numbers of articles in test sets are the same for vectors with different dimensions, I use the whole running time to show the online cost, like Table I. The unit in Table III is second.

Table III: Online cost (second) of the apriori algorithm

Online	1000	2000	3000	4000	5000
s=2,c=2	2.118s	3.319s	6.109s	7.933s	13.055s
s=2,c=3	1.889s	3.585s	5.287s	8.537s	11.563s
s=2,c=5	1.804s	3.117s	5.052s	6.634s	11.903s
s=2,c=7	1.907s	3.010s	6.057s	7.681s	9.594s
s=2,c=10	1.957s	3.232s	5.253s	8.545s	10.440s
s=3,c=2	1.229s	1.647s	2.838s	4.107s	6.855s
s=3,c=3	1.011s	1.684s	3.278s	4.086s	5.518s
s=3,c=5	1.271s	2.205s	2.964s	5.327s	6.097s
s=3,c=7	0.931s	1.501s	3.305s	4.703s	6.324s
s=3,c=10	1.108s	1.568s	3.385s	3.991s	5.404s
s=5,c=2	0.463s	0.576s	1.155s	1.426s	2.300s
s=5,c=3	0.387s	0.736s	1.023s	1.845s	2.196s
s=5,c=5	0.389s	0.729s	1.073s	1.533s	2.324s
s=5,c=7	0.342s	0.590s	0.941s	1.580s	2.319s
s=5,c=10	0.385s	0.660s	0.832s	1.289s	2.282s
s=7,c=2	0.261s	0.452s	0.862s	1.249s	1.836s
s=7,c=3	0.262s	0.495s	0.787s	1.179s	1.522s
s=7,c=5	0.262s	0.374s	0.675s	1.302s	1.216s
s=7,c=7	0.247s	0.425s	0.721s	1.097s	1.424s
s=7,c=10	0.219s	0.477s	0.586s	0.849s	1.634s
s=10,c=2	0.273s	0.372s	0.800s	1.110s	1.866s
s=10,c=3	0.168s	0.466s	0.676s	1.059s	1.808s
s=10,c=5	0.185s	0.289s	0.601s	0.741s	1.444s
s=10,c=7	0.139s	0.318s	0.454s	0.788s	1.299s
s=10,c=10	0.126s	0.224s	0.441s	0.764s	1.021s

As Table III shown, for the vectors with the same dimensions, the test with larger s and larger c need

lower online cost, in most cases. This is very easy to understand. From Table II, we have known larger s and larger c generates less rules. When we classify one article, we actually need to traverse the rule set until we find the appropriate rule. So the online cost is affected by the number of rules.

4.2 Quality

I collect the accuracy for each test and show the results in Table IV. It's very hard to get some conclusions from table IV. But it seems the test with larger support value performs worse than the test with less support value. And in most cases, the test with larger confidence value performs better than the test with less confidence value. When the test with too large support and confidence values, the performance is not good. The reason is obvious. Too large support and confidence values bring too few rules. The rules in this case cannot work well for most articles so that many articles have to be classified into the default classification, earn. This harms the performance.

Table IV: Online cost (second) of the apriori algorithm

Accuracy	1000	2000	3000	4000	5000
s=2,c=2	53.99%	56.18%	56.01%	56.13%	52.66%
s=2,c=3	55.61%	54.62%	57.11%	56.42%	60.17%
s=2,c=5	54.28%	57.05%	55.32%	58.79%	56.76%
s=2,c=7	58.21%	56.42%	55.95%	58.50%	58.50%
s=2,c=10	57.05%	58.44%	58.84%	57.63%	59.48%
s=3,c=2	51.21%	49.88%	50.40%	53.70%	54.39%
s=3,c=3	51.45%	53.64%	48.21%	55.26%	50.52%
s=3,c=5	53.76%	51.45%	56.53%	57.46%	53.58%
s=3,c=7	54.80%	53.64%	56.53%	54.86%	53.58%
s=3,c=10	54.97%	53.01%	52.95%	54.39%	54.45%
s=5,c=2	34.74%	35.84%	35.66%	36.42%	33.12%
s=5,c=3	34.74%	33.35%	38.73%	41.91%	37.11%
s=5,c=5	35.43%	36.24%	37.75%	43.01%	42.49%
s=5,c=7	42.20%	42.31%	45.66%	43.70%	40.81%
s=5,c=10	45.78%	42.89%	46.59%	44.16%	46.76%
s=7,c=2	19.71%	24.68%	32.72%	26.82%	20.12%
s=7,c=3	28.03%	33.64%	27.80%	25.03%	33.76%
s=7,c=5	29.08%	33.70%	29.77%	39.31%	32.77%
s=7,c=7	38.50%	33.99%	40.58%	33.87%	33.29%
s=7,c=10	44.57%	44.16%	39.02%	43.82%	45.49%
s=10,c=2	17.98%	21.56%	19.08%	15.66%	19.77%
s=10,c=3	17.86%	20.87%	22.08%	22.43%	24.34%
s=10,c=5	32.43%	27.46%	33.64%	27.05%	35.49%
s=10,c=7	43.35%	21.62%	35.32%	27.46%	29.83%
s=10,c=10	31.27%	34.22%	32.89%	33.06%	40.23%

5. Comparison

In this section, I will compare the performance of apriori with the algorithms we used in assignment III, including K-nearest-neighbors algorithm, decision tree algorithm, and naive Bayesian algorithm. In assignment III, I use feature vectors with 1,000 dimensions. Because of the limited space, I will simply show the online cost, offline cost, and accuracy of the best cases for these algorithms. The results can be seen in Table V. The units of offline cost and online cost are second.

Table V: Comparison among different algorithms

	Offline Cost	Online Cost	Accuracy
apriori (s=2,c=7)	11.417	1.907	58.21%
KNN (K=133)	2.20359182	70.123656	43.88%
Tree	5.22655082	0.05124187	25.01%
Bayes	0.25466256	0.56750298	8.84%

From Table V, we can see that apriori has the highest accuracy. However, it needs the largest offline cost. KNN also provides acceptable accuracy. The problem of KNN is that online cost is too high. Decision tree algorithm's accuracy is not good as KNN and apriori. But the online cost is very low. For Bayes algorithm, the offline cost and online cost are very low. But it has the lowest accuracy. I will continue conclude the observations in the last section.

6. Conclusion

It is very complicated and difficult to explain the phenomenon we have tested and observed. However, we can conclude some results we have observed:

- For apriori algorithm,
    - Typically, larger support value and confidence value bring less number of rules.
    - Typically, larger support value and confidence value bring lower offline cost.
    - Typically, larger support value and confidence value bring lower online cost.
    - In most cases, larger confidence value leads to higher accuracy.
    - In most cases, larger support value leads to lower accuracy.
- When we want to build a classifier, we should choose the appropriate one based on our demands.
    - If we want high accuracy, apriori and KNN are good choices. Furthermore, if we want to have good estimating response time, we'd better use apriori. On the other hand, if we want to get the classifier as soon as possible, we'd better use KNN.
    - If we want to make a balance between efficacy and efficiency, the decision tree algorithm is very good. It has very low offline cost, low online cost, and middle accuracy, compared to other algorithms.
    - If we want the shortest time for training and estimating, we can choose Bayes algorithm. However, it should be noted that the accuracy of Bayes is very low.