AprioryAlgorithm

2016059198 진승현

1. 프로그램 개요

Transaction list Text File과 Minimum Support를 입력데이터로 받아 Apriori Algorithm을 통해 Association Rule을 Text File로 출력하는 프로그램이다.

2. Apriory Algorithm

Transaction에서 Association Rule을 구하기 위해 Frequent Itemset을 구할 때 Candidates를 최소한으로 줄이기 위해 도입된 방법이다. 기본 원리는

1. 임의의 itemset A, B가 있을 때 A가 B의 부분집합이라면 B의 support는 A의 support보다 작다.

2. 1에 의해 임의의 itemset이 Frequent itemset이 아니라면(Minimum support보다 적다면) 해당 itemset의 superset들은 모두 Frequent itemset이 아니다.

그러므로 해당 itemset의 모든 subsets은 Candidate로 고려하지 않아도 된다. (Pruning)

3. 코드 설명(함수 기준)

1) Support & Confidence Finding

```
#support = 전체 transactions에서 itemset이 포함된 transaction의 비율
def find_support(itemset):
    sup_count = 0
    for transaction in trans:
        if set(transaction).issuperset(set(itemset)): #transation이 itemset을 포함할 경우
            sup_count += 1
    return sup_count/total_trans_num # support = sup_count / 전체 거래 수


#confidence = X->Y의 Association Rule에서 X를 가진 transaction이 Y도 가질 조건적확률
def find_confidence(rule):
    return find_support(rule[0]|rule[1])/find_support(rule[0]) #계산은 Sup(X,Y)/Sup(X)
```

2) Apriori

```
#실제 Apriori 구현
def Apriori(candidates, pruned_sets):

    #next sets initialize
    next_candidates = []
    next_pruned_sets = []

    for i in range(len(candidates)):
        candidate = set(candidates[i])          #support를 계산할 candidate

        for j in range(i+1, len(candidates)):

            original_candidate = candidate.copy()         #계산 후 복구할 original candidate
            candidate = candidate | set(candidates[j])  #합집합 set

            #1. 이미 검사된 candidate이라면 패스
            if candidate in next_candidates or candidate in next_pruned_sets:
                continue

            #2. candidate가 pruned_sets 중 하나의 superset이라면 패스
            pruned_check = False #pruned 여부
            for pruned_set in pruned_sets:
                if candidate.issuperset(set(pruned_set)):
                    pruned_check = True
                    break
            if pruned_check:
                continue

            #3.
            if find_support(candidate) >= min_sup: #Minimum Support를 만족하면 next_candidates로 copy
                next_candidates.append(candidate.copy())
            else:                                  #만족하지 못하면 next_pruned_sets로 copy
                next_pruned_sets.append(candidate.copy())

            candidate = original_candidate #원래값으로

    if(next_candidates == []): #모든 candidate가 pruned되어 next_candidate가 생성되지 않을 때
        return candidates
    else:                     #new_candidates가 있다면 recursive하게 candidates를 추가함
        return Apriori(next_candidates, next_pruned_sets) + candidates
```

3) get Association Rules

```
#itemsets에서 association rules set을 가져오는 함수
def get_rule_sets(itemsets):
    from itertools import chain, combinations
    s = list(itemsets)

    #get Powerset of itemsets
    powerset = list(chain.from_iterable(combinations(s, r) for r in range(1, len(s))))

    #대응하는 subsets끼리 묶어서 return
    return [(set(powerset[i]), set(powerset[len(powerset) - 1 - i])) for i in range(len(powerset))]
```

4. 실행 예제

1)Argument Format

Python AprioryAlgorithm.py [min_support] [input file src] [output file src]

```
2021_ite4005_2016059198₩AprioriAlgorithm>python AprioriAlgorithm.py 5 input.txt output_5.txt

2021_ite4005_2016059198₩AprioriAlgorithm>
```

2) input & output file

input.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
7 14
9
18 2 4 5 1
1 11 15 2 7 16
4 13
2 1 16
15 7 6 11 18 9
12 19 14
11 2 13 4
11 13
7 4 2 17 19 3
8 16 1
18 16 15 10 2 8
6 0 4 5
9
10
1 13 8 9 3 16
4
16 0 6 11 8
```

output_5.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

{1}	{8, 16, 3}	9.40	31.54
{3}	{8, 1, 16}	9.40	31.33
{8}	{16, 1, 3}	9.40	20.80
{16}	{8, 1, 3}	9.40	22.17
{1, 3}	{8, 16}	9.40	87.04
{8, 1}	{16, 3}	9.40	61.04
{16, 1}	{8, 3}	9.40	58.02
{8, 3}	{16, 1}	9.40	36.43
{16, 3}	{8, 1}	9.40	37.30
{8, 16}	{1, 3}	9.40	31.13
{8, 1, 3}	{16}	9.40	97.92
{16, 1, 3}	{8}	9.40	97.92
{8, 1, 16}	{3}	9.40	81.03
{8, 16, 3}	{1}	9.40	39.17
{2}	{8, 16, 3}	5.80	21.97
{3}	{8, 16, 2}	5.80	19.33
{8}	{16, 2, 3}	5.80	12.83
{16}	{8, 2, 3}	5.80	13.68
{2, 3}	{8, 16}	5.80	80.56

5. Specification of Testing

OS : Windows 10 Home (x64)

Language version : Python 3.7.0