BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

Course – Data Analytics Open Elective

UID	2021300126
Name	Pranay Singhvi
Class and Batch	TE Computer Engineering - Batch A
Date	19/03/2024
Lab #	5
Aim	To perform classification using Apriori Algorithm
Theory	**Apriori Algorithm** **Apriori algorithm** is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets. To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called *Apriori property* which helps by reducing the search space. **Apriori Property –** All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that *All subsets of a frequent itemset must be frequent(Apriori property).* *If an itemset is infrequent, all its supersets will be infrequent.* Before we start understanding the algorithm, go through some definitions which are explained in my previous post. Consider the following dataset and we will find frequent itemsets and generate association rules for them.

TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

minimum support count is 2

minimum confidence is 60%

Step-1: K=1

(I) Create a table containing support count of each item present in dataset –

Called **C1(candidate set)**

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

(II) compare candidate set item's support count with minimum support count(here min_support=2 if support_count of candidate set items is less than min_support then remove those items). This gives us itemset L1.

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Step-2: K=2

- Generate candidate set C2 using L1 (this is called join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common.
- Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.(Example subset of{I1, I2} are {I1}, {I2} they are frequent.Check for each itemset)
- Now find support count of these itemsets by searching in dataset.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

(II) compare candidate (C2) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

Step-3:

- Generate candidate set C3 using L2 (join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common. So here, for L2, first element should match.

So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, i5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}

- Check if all subsets of these itemsets are frequent or not and if not, then remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2},{I2, I3},{I1, I3} which are frequent. For {I2, I3, I4}, subset {I3, I4} is not frequent so remove it. Similarly check for every itemset)
- find support count of these remaining itemset by searching in dataset.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

(II) Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

Step-4:

- Generate candidate set C4 using L3 (join step). Condition of joining L_{k-1} and L_{k-1} (K=4) is that, they should have (K-2) elements in common. So here, for L3, first 2 elements (items) should match.
- Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contains {I1, I3, I5}, which is not frequent). So no itemset in C4
- We stop here because no frequent itemsets are found further

Thus, we have discovered all the frequent item-sets. Now generation of strong association rule comes into picture. For that we need to calculate confidence of each rule.

Confidence –

A confidence of 60% means that 60% of the customers, who purchased milk and bread also bought butter.

Confidence(A->B)=Support_count(A∪B)/Support_count(A)

So here, by taking an example of any frequent itemset, we will show the rule generation.

Itemset {I1, I2, I3} //from L3

SO rules can be

[I1^I2]=>[I3] //confidence = sup(I1^I2^I3)/sup(I1^I2) = 2/4*100=50%

[I1^I3]=>[I2] //confidence = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%

[I2^I3]=>[I1] //confidence = sup(I1^I2^I3)/sup(I2^I3) = 2/4*100=50%

[I1]=>[I2^I3] //confidence = sup(I1^I2^I3)/sup(I1) = 2/6*100=33%

[I2]=>[I1^I3] //confidence = sup(I1^I2^I3)/sup(I2) = 2/7*100=28%

[I3]=>[I1^I2] //confidence = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%

So if minimum confidence is 50%, then first 3 rules can be considered as strong association rules.

Limitations of Apriori Algorithm

Apriori Algorithm can be slow. The main limitation is time required to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets i.e. it is not an efficient approach for large number of datasets. For example, if there are 10^4 from frequent 1- itemsets, it need to generate more than 10^7 candidates into 2-length which in turn they will be tested and accumulate. Furthermore, to detect frequent pattern in size 100 i.e. v1, v2… v100, it have to generate 2^100 candidate itemsets that yield on costly and wasting of time of candidate generation. So, it will check for many sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions.

Code

```
data = [
    ['T1', ['Milk', 'Bread', 'Eggs']],
    ['T2', ['Bread', 'Butter']],
    ['T3', ['Bread', 'Cheese']],
    ['T4', ['Milk', 'Bread', 'Butter']],
    ['T5', ['Milk', 'Cheese']],
    ['T6', ['Bread', 'Cheese']],
    ['T7', ['Milk', 'Cheese']],
    ['T8', ['Milk', 'Bread', 'Cheese', 'Eggs']],
    ['T9', ['Milk', 'Bread', 'Cheese']]
]


init = []
for i in data:
    for q in i[1]:
        if(q not in init):
            init.append(q)
init = sorted(init)
print(init)

sp = 0.8
s = int(sp*len(init))
print("Support:", s)

from collections import Counter

c = Counter()
for i in init:
    for d in data:
        if(i in d[1]):
            c[i]+=1
print("C1:")
for i in c:
    print(str([i])+": "+str(c[i]))
print()
l = Counter()
for i in c:
    if(c[i] >= s):
        l[frozenset([i])]+=c[i]
print("L1:")
for i in l:
    print(str(list(i))+": "+str(l[i]))
print()
pl = l
pos = 1
for count in range (2,1000):
    nc = set()
```

```
    temp = list(l)
    for i in range(0,len(temp)):
        for j in range(i+1,len(temp)):
            t = temp[i].union(temp[j])
            if(len(t) == count):
                nc.add(temp[i].union(temp[j]))
    nc = list(nc)
    c = Counter()
    for i in nc:
        c[i] = 0
        for q in data:
            temp = set(q[1])
            if(i.issubset(temp)):
                c[i]+=1
    print("C"+str(count)+":")
    for i in c:
        print(str(list(i))+": "+str(c[i]))
    print()
    l = Counter()
    for i in c:
        if(c[i] >= s):
            l[i]+=c[i]
    print("L"+str(count)+":")
    for i in l:
        print(str(list(i))+": "+str(l[i]))
    print()
    if(len(l) == 0):
        break
    pl = l
    pos = count
print("Result: ")
print("L"+str(pos)+":")
for i in pl:
    print(str(list(i))+": "+str(pl[i]))
print()

from itertools import combinations

# Prompt the user to input the minimum confidence percentage
min_confidence = float(input("Enter the minimum confidence percentage: "))

for l in pl:
    c = [frozenset(q) for q in combinations(l,len(l)-1)]
    mmax = 0
    for a in c:
        b = l-a
        ab = l
        sab = 0
```

```
        sa = 0
        sb = 0
        for q in data:
            temp = set(q[1])
            if a.issubset(temp):
                sa += 1
            if b.issubset(temp):
                sb += 1
            if ab.issubset(temp):
                sab += 1

        # No need to calculate confidence percentage here

        # Compare confidence with minimum confidence input by the user
        confidence_a = sab / sa * 100
        if confidence_a >= min_confidence:
            print(str(list(a)) + " -> " + str(list(b)) + " = " + str(confidence_a)
+ "%")

        confidence_b = sab / sb * 100
        if confidence_b >= min_confidence:
            print(str(list(b)) + " -> " + str(list(a)) + " = " + str(confidence_b)
+ "%")

    curr = 1
    print("choosing:", end=' ')
    for a in c:
        b = l - a
        ab = l
        sab = 0
        sa = 0
        sb = 0
        for q in data:
            temp = set(q[1])
            if a.issubset(temp):
                sa += 1
            if b.issubset(temp):
                sb += 1
            if ab.issubset(temp):
                sab += 1

        # No need to calculate confidence percentage here

        confidence_a = sab / sa * 100
        if confidence_a >= min_confidence:
            print(curr, end=' ')
        curr += 1
```

```
        confidence_b = sab / sb * 100
        if confidence_b >= min_confidence:
            print(curr, end=' ')
        curr += 1

    print()
    print()
```

Output

```
pranaysinghvi@Pranays-MacBook-Air 8)DA % /usr/local/bin/python3 "/Users/pranaysinghvi/Library/Cl
-Personal/SPIT College/3)Class/Semester 6/8)DA/1)Experiment/5_/Experiment 5 Code.py"
Support: 4
C1:
['Bread']: 7
['Butter']: 2
['Cheese']: 6
['Eggs']: 2
['Milk']: 6

L1:
['Bread']: 7
['Cheese']: 6
['Milk']: 6

C2:
['Bread', 'Cheese']: 4
['Cheese', 'Milk']: 4
['Bread', 'Milk']: 4

L2:
['Bread', 'Cheese']: 4
['Cheese', 'Milk']: 4
['Bread', 'Milk']: 4

C3:
['Bread', 'Milk', 'Cheese']: 2

L3:

Result:
L2:
['Bread', 'Cheese']: 4
['Cheese', 'Milk']: 4
['Bread', 'Milk']: 4

Enter the minimum confidence percentage: 60
['Cheese'] -> ['Bread'] = 66.67%
['Cheese'] -> ['Bread'] = 66.67%


['Cheese'] -> ['Milk'] = 66.67%
['Milk'] -> ['Cheese'] = 66.67%
['Milk'] -> ['Cheese'] = 66.67%
['Cheese'] -> ['Milk'] = 66.67%


['Milk'] -> ['Bread'] = 66.67%
['Milk'] -> ['Bread'] = 66.67%
```

Conclusion