

Marketing Analytics
Professor Swagato Chatterjee
Vinod Gupta School of Management
Indian Institute of Technology, Kharagpur
Lecture 43
RFM and Market Basket Analysis (Contd.)

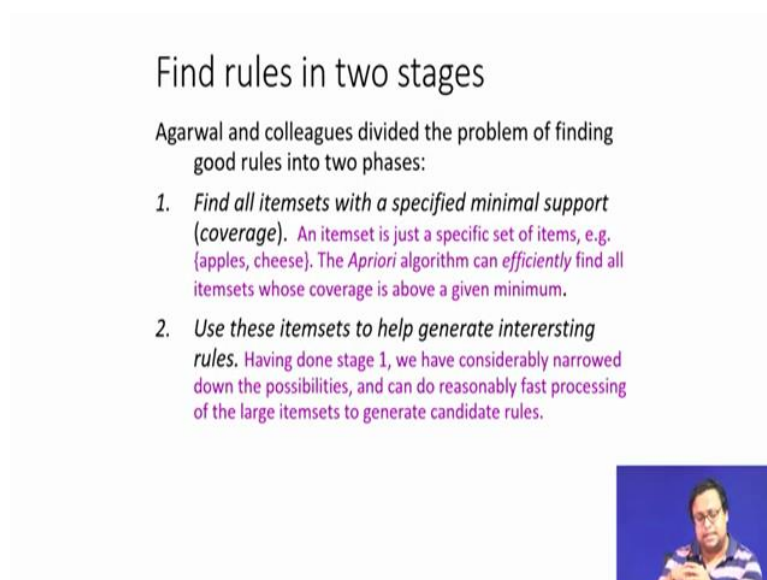
(Refer Slide Time: 00:23)



The slide features a blue header with two logos: the Indian Institute of Technology Kharagpur logo on the left and the NPTEL logo on the right. Below the header, the text reads: "NPTEL ONLINE CERTIFICATION COURSES", "MARKETING ANALYTICS", "DR SWAGATO CHATTERJEE", "VINOD GUPTA SCHOOL OF MANAGEMENT, IIT KHARAGPUR", "Module 08:", and "RFM and MARKET BASKET ANALYSIS".

Hello, everybody, welcome to Marketing Analytics course. We are in week 8, session 4 actually and we are discussing Market Basket analysis. Till now, we have discussed about the what is the requirement of Market Basket Analysis and the basic easier version of the algorithm. Now, let us talk about (())(00:31) algorithm which is the advanced version which can be used for larger data sets.

(Refer Slide Time: 0:51)



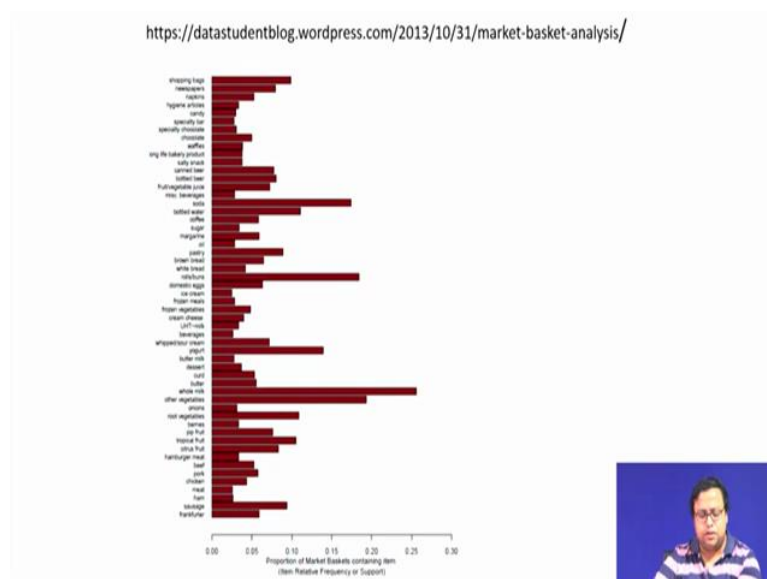
The slide is titled "Find rules in two stages". The text below the title reads: "Agarwal and colleagues divided the problem of finding good rules into two phases:". This is followed by a numbered list of two points. The first point is: "1. Find all itemsets with a specified minimal support (coverage). An itemset is just a specific set of items, e.g. (apples, cheese). The Apriori algorithm can efficiently find all itemsets whose coverage is above a given minimum." The second point is: "2. Use these itemsets to help generate interesting rules. Having done stage 1, we have considerably narrowed down the possibilities, and can do reasonably fast processing of the large itemsets to generate candidate rules." In the bottom right corner, there is a small video inset showing a man with glasses and a striped shirt looking at a device.

So, I was here in this presentation, I will just scroll it up. And I was basically just one minute. I was here in the presentation, yes. So, I was talking about find the rule in 2 stages for this particular thing. So, find all items with a specified minimal support. So, first of all you will not do it for everybody, I will only find out those guys which have minimal support.

So, item set is just as a specific set of items, like I told when apples and cheese occurred, what was the rule if you remember? That apples and cheese occurs, honey also occurs, so apple and cheese is the item set. So, the first I will choose such kind of combinations, which has a minimum cut off, minimum support.

So, what is that? Let say I will say 10, 10 out or at least 1 percent, 1 percent of the whole data set they should occur, less than 1 percent will not even consider those combination that is number one. Now, these use these items is to help generating the rules, so you do not create a rule, you just break based on certain conditions. So, having done stage one, we have considerably narrowed down the possibilities and to can do reasonably fast processing on the large itemsets to general candidate rules.

(Refer Slide Time: 02:00)



So, this is what we do. So, we find out all the products and market of proportion of market baskets containing these items and then we do something.


(Refer Slide Time: 02:10)

Terminology

k-itemset : a set of k items. E.g.
 {beer, cheese, eggs} is a 3-itemset
 {cheese} is a 1-itemset
 {honey, ice-cream} is a 2-itemset

support: an itemset has support $s\%$ if $s\%$ of the records in the DB contain that itemset.

minimum support: the Apriori algorithm starts with the specification of a minimum level of support, and will focus on itemsets with this level or above.



So, for an example, terminology says that there are k itemset let us say, a set of k items. So, it if it is a three item set that means a set of three items, if it is a one item set than it is a set of one item, two item set means it is a set of two items. But its support an items set has s percentage support that means in the data set, there are s percentage transactions, which has these two guys together. And what is the minimum support? The Apriori algorithm starts with the specification of a minimum support.

So, we do not use all itemsets, we only use those itemsets which has a minimum number of occurrences. Let say if my database is 20 million as I told 20 million and I am saying 1 percent should occur, that means what? That means basically 200,000 times that itemsets should occur, 1 percent is a very big number actually, we do not even say 1 percent, we say 0.01, 0.01 percent means?

1 by 10001 10 to the power 4. So, 20 million comes to be, 20 million point 01 percent is basically how much? 2000, so 2000 times one items set should occur, then only we will go ahead and create the rule otherwise, I will not create the rule, so, that is the first job.

(Refer Slide Time: 03:30)

Terminology

large itemset: doesn't mean an itemset with many items. It means one whose support is at least minimum support.

L_k : the set of all large k -itemsets in the DB.

C_k : a set of *candidate* large k -itemsets. In the algorithm we will look at, it generates this set, which contains all the k -itemsets that might be large, and then eventually generates the set above.

What is a large item set? Does not mean an itemset, which with many items. It means one whose support is at least had the minimum support that is how we use these large itemset definition. So, L_k is the set of all large k item sets in the database. And C_k is a set of candidate, so C_k is a set of candidate large k item sets.

In the algorithm we look at it generates this set which contents all k itemsets which might be large and then eventually. So, C_k is basically one particular itemset in L_k , L_k is the superset of all large k itemsets in DB and what is large? Large is basically the num item sets which has support more than the cut off. So, that is something that we will use this terminologies will use to define the algorithm.

(Refer Slide Time: 04:27)

ID	a	b	c	d	e	f	g	h	i
1	1	1		1				1	1
2			1	1	1				
3		1	1			1			
4		1				1			1
5					1		1		
6						1			1
7	1			1				1	
8						1			1
9			1		1				
10		1					1		
11					1		1		
12	1								
13			1			1			
14			1			1			
15								1	1
16				1					
17	1					1			
18	1	1	1	1				1	
19	1	1		1			1	1	
20					1				

E.g.
3-itemset {a,b,h}
has support 15%

2-itemset {a, i}
has support 0%

4-itemset {b, c, d, h}
has support 5%

If minimum support is 10%, then {b} is a large itemset, but {b, c, d, h} is a *small* itemset!



Now, here you just see that, which are the three item sets? So, if this is the product, one of the three itemset is a, b, h and a, b, h is occurring here, a, b and h is occurring here and then a, b and h is occurring here, a, b and h is occurring here. So, ID number 1, 18 and 19, in these three cases a, b and h is happening.

So, a, b and h is a three itemset, three does itemset, because three items are there. Its support is 3 by 20, which is 15 percent, here this which is 15 percent. Similarly, a comma i is a two item set whose support is 0 percent nowhere is occurring and etc. So, now if I take the cut off as 10 percent then this guy will be a part of my large, this guy will be a large item set.

So, if the minimum support is 10 percent then b is a large itemset, you see but b, c, d, h is a small itemset, because this is occurring only 5 percent but b is occurring 1, 2, 3, 4, 5, 6 is a one item itemset which is occurring 6 time out of 20 that means 30 percent, so which is a large item set, so minimum support is 10 percent.

(Refer Slide Time: 05:48)

The Apriori algorithm for finding large itemsets efficiently in big DBs

```

1: Find all large 1-itemsets
2: For (k = 2 ; while Lk-1 is non-empty; k++)
3   {Ck = apriori-gen(Lk-1)
4   For each c in Ck, initialise c.count to zero
5   For all records r in the DB
6     {Cr = subset(Ck, r); For each c in Cr, c.count++ }
7   Set Lk := all c in Ck whose count >= minsup
8 } /* end -- return all of the Lk sets.
```



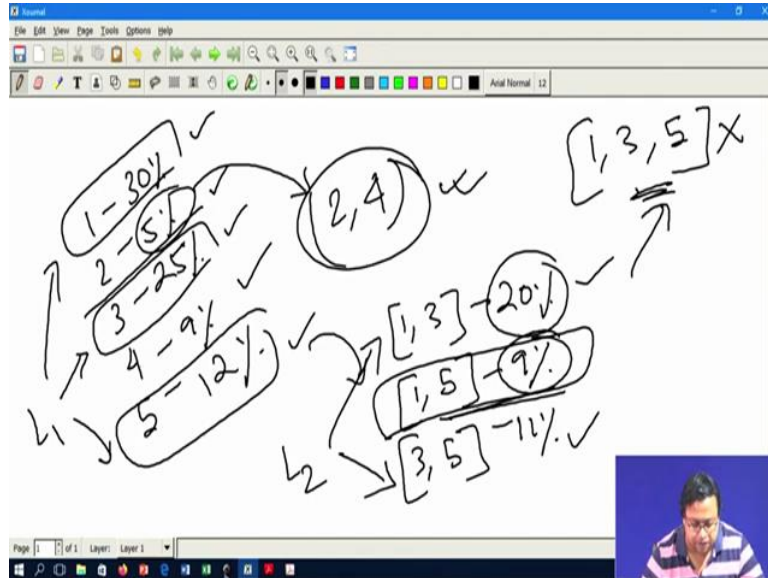
So, the Apriori algorithm for finding large items item set efficiently in big databases. So, this is the algorithm that is saying, so it find all large one itemsets that is the first job that means, find out which items are occurring at least 10 percent of times if that is your cost of. Then for k is equal to 2, while L k minus 1 is non-empty, k plus plus. So, when L k minus 1 is non-empty, it is not blank then increase the k, otherwise keep the k like that.

So, if two items are not occurring 10 percent times then three items will not occur 3 10 percent times that is it is a, a and b together are not occurring 10 percent times, then can a, b, h occur 10 percent times? No. So, if a and b this is not occurring 10 percent times, that means

a and b is not a part of the large data set, large itemset then a, b, h cannot be a part of the large itemset.

So, you will find further find out item sets for only those guys whose individual items were there in the earlier version. That is let us say, when you I am trying to analyse two itemset I will only use those products for which one item set was a part of the large category.

(Refer Slide Time: 07:31)



So, to give an example, let us give an example what I am trying to say here. I am trying to say, I will just change the page size and etc. Let us talk about it, so I am going to say that I have item 1, item 2, item 3 item 4 and item 5 and these items. This has 30 percent, this is 5 percent, this is 25 percent, this is 9 percent and this is 12 percent fair enough.

Now, I am saying that I will find out two combinations. Now, if 2 has occurred only 5 percent times, then 2 comma 4, can it occur more than 5 percent times? No, because 2 has occurred. So, out of what 2 has occurred, one subset is where 2 and 4 will occur. So, this will be always lower than this, the percentage of this will always be lower than this, so, I will rather will not see any combination which has this.

So, I will probably also not see any combination which has this, I will see only the combinations of these guys. So, that is something which is important to understand that. So, this guy, this person, this person and this person is something that I will consider, because 1 and 3 happening together will be always be lower than the individual probably so 1 happening together, so this is something

So, from here, what are the two item things I will say? Then I will say that 1, 3 I will check 1, 5 I will check and 3, 5 I will check, fair enough. Now, let us say 1, 3 is around 20 percent and 1, 5 is around let us say 9 percent and 3, 5 is around let us say 11 percent, fair enough. Now, what is the probability that 1 comma 3, 5 comma happened?


See one subset on this is 1, 3 another subset of this is 1, 5. 1, 3 occurs 20 percent 1, 5 occurs 9 percent, so this guy has to be lower than 9 percent, has to be because out of those cases, where 1, 5 occurs, there are some cases where 3 also occurs, so this guy has to be lower than 9 percent.

So, I will practically not considered this combination, because this is the combination which is not a large one 10 percent was my cut off. So, then I will, so, if I do not consider these basically I am not consider other two combinations will also give me this. So, not consider any three itemset. So, the only 5 only item sets which are there in my these things these are my L 1s and these are my L 2, L 3 is 0, L 3 is blank, non-empty, empty.

(Refer Slide Time: 10:31)

The Apriori algorithm for finding large itemsets efficiently in big DBs

```
1: Find all large 1-itemsets
2: For (k = 2 ; while Lk-1 is non-empty; k++)
3   {Ck = apriori-gen(Lk-1)
4   For each c in Ck, initialise c.count to zero
5   For all records r in the DB
6   {Cr = subset(Ck, r); For each c in Cr, c.count++ }
7   Set Lk := all c in Ck whose count >= minsup
8 } /* end -- return all of the Lk sets.
```



So, only for non-empty data sets it is saying, whenever L k minus 1 is non-empty then only you go ahead, otherwise you do not go ahead, otherwise you do not increase k. And then you do the same thing, you find out what are the how many sets are there? How many counts are there? The one that I just showed in the in the in the basic picture, they find out the count, so they deduce the number of choice sets. So, you can run this particular algorithm in the database and you will find out that these are the three items set which has 10 percent support.

(Refer Slide Time: 10:55)

Generating candidate itemsets ...


Suppose these are the only 3-itemsets all have >10% support:

- {a, b, c}
- {a, e, g}
- {e, f, h}
- {e, f, k}
- {p, q, r}

One possibility:

1. note all the items involved:
{a, b, c, e, f, g, h, k, p, q, r}
2. generate all subsets of 4 of these:
{a,b,c,e}, {a,b,c,f}, {a,b,c,g}, {a,b,c,h},
{a,b,c,k}, {a,b,c,p},... etc ... there are
330 possible subsets in this case !

But, hold on: we can easily see that {a,b,c,e} couldn't have 10% support – because {a,b,e} is *not* one of our 3-itemsets



How do we generate the 4 item sets? While it have 10 percent support, so anybody which non-combination of this you will find out and if they have 10 percent support, then you consider them as 4 item sets.

So, one possibility is that, note all of these items are involved a, b, c, e, f, g, h to r. Generate all possible 4 combinations and then you find out who of them is, but that will again take lots of times. But hold on, we can easily see that a, b, c, e could not have 10 percent support, because a, b, e is not one of our three item sets. See a, b, e was not at all in our three item set, so a, b, e cannot have 10 percent support, so then a, b, c, e also cannot have 10 percent support, if a, b, e cannot have 10 percent support.

So, I have to create combinations in such a way such that these guys, so the all the subset of the original these combinations are there here, all the possible subsets. So, a, b, c has to be there b, c has to be there a, c all possible subset of the four item combination should be here, then only we can go ahead and find out that particular guys support, otherwise we will not find out.

(Refer Slide Time: 12:23)


A neat Apriori trick

- i. enforce that subsets are always arranged 'lexicographically' (or similar), as they are already on the left
- ii. **Only** generate $k+1$ -itemset candidates from k -itemsets that differ in the last item.

So, in this case, the only candidate 4-itemset would be:

{e, f, h, k}

{a, b, c}
{a, e, g}
{e, f, h}
{e, f, k}
{p, q, r}



So, the same goes for several other of these subjects. So, enforce that subsets are always arranged lexicographically and they are already on the left. Only generate k plus one items from k items which differ in the last item, so that is what it is the algorithm that they are giving.

(Refer Slide Time: 12:39)


A neat Apriori trick

- i. enforce that subsets are always arranged 'lexicographically' (or similar), as they are already on the left
- ii. **Only** generate $k+1$ -itemset candidates from k -itemsets that differ in the last item.

And in this case, the only candidate 5-itemsets would be:

{a, e, g, r, w}, {n, q, r, t, v}

{a, b, c, e}
{a, e, g, r}
{a, e, g, w}
{e, f, k, p}
{n, q, r, t}
{n, q, r, v}
{n, q, s, v}



And this case, the only five items will be a, e, g or w and n, q, r, t, f, you can check that. So, only the last item there, change in the fast three items say I putting it lexicographically and changing the last item.

(Refer Slide Time: 12:54)

A neat Apriori trick

This trick

- **guarantees** to capture the itemsets that have enough support,
- will still generate **some** candidates that don't have enough support, so we still have to check them in the 'pruning' step,
- is particularly convenient for implementation in a standard relational style transaction database; it is a certain type of 'self-join' operation.



So, this trick guarantees to capture the items that have enough support. Will still generate some candidates that do not have enough support, so will still have to check them in the pruning step. And it is particularly convenient for implementation in a standard and relational style transactional data. So, that is what happens in the background of our a Apriori algorithm.


(Refer Slide Time: 13:20)

From itemsets to rules

The Apriori algorithm finds interesting (i.e. frequent) itemsets.


E.g. it may find that {apples, bananas, milk} has coverage 30%
-- so 30% of transactions contain each of these three things.

What can you say about the coverage of {apples, milk}?



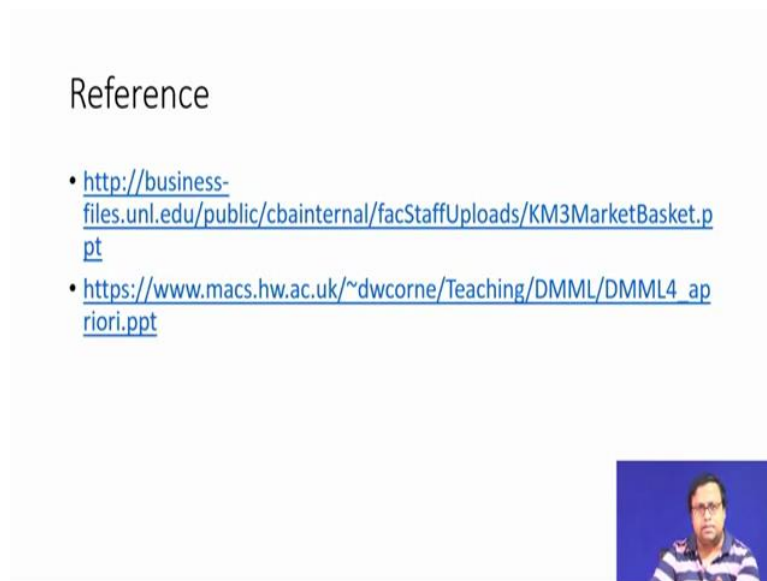
We can invent several potential rules, e.g.:
IF basket contains apples and bananas, it also contains MILK.

Suppose support of {a, b} is 40%; what is the confidence of th



So, there is some example given, I will not spend my time on this example you can check out and then we try out and find out the rules that what kind of rules. And what can you say about the coverage of apples and milk? We can invest several potential rules if basket conscious, apples and bananas, it also contains milk, so this is something. So, support of a, b is 40 percent which where is the confidence of this rule, you have to find out that.

(Refer Slide Time: 13:41)



Reference

- <http://business-files.unl.edu/public/cbainternal/facStaffUploads/KM3MarketBasket.ppt>
- https://www.macs.hw.ac.uk/~dwcorne/Teaching/DMML/DMML4_apriori.ppt

So, the items has been taken from these two links, you can also go and read about read more details from this links. And in the next video we will actually do in a hands on way how to deal with this thing. So, some appendix and this thing is also there is the presentation, you can look up and try out on your own.

So, thank you for being in this particular video, we have discussed the algorithm and we have discussed the usage of marketing Market Basket Analysis in a quite a bit. In the next video we will actually discuss how to do it in our in a hands on way. Thank you very much.