ISSN:0975-9646

FDM (Fast Distributed Mining) over normal mining algorithm based on A-priori property and its application in market basket analysis

Sateesh Reddy, Ravi Konaraddi, Sivagama Sundari G

CSE Department, MVJCE
channasandra, Bangalore, India

***Abstract*— in this paper we are going to discuss about the difference between FDM and normal data mining algorithm in finding frequent item-sets globally as well as locally. How this difference can be used as an application in market basket analysis with respect to supplier of the item. We also use the concept of hierarchical data mining for applying different support at different levels.**

Keywords—Distributed Mining, Frequent Item-sets, Association Rules, hierarchical mining.

1 Introduction

We study the difference between FDM and normal A-priori algorithm for mining of frequent item-sets. In A-priori algorithm mining will be performed on all the transactions, which is obtained by merging transactions from different sources. This results in finding globally frequent item-sets. In case of FDM, mining will be performed separately with respect to transactions that corresponds to separate sources, these results in locally frequent item-sets. Then, these locally frequent items from separate sources will be merged (union) to obtain candidate set which will be broadcasted to all the sources to find the local support count of all the items in them, on which the union will be performed at each source i.e..to sum up the local support count of each item in the candidate set from neighboring sources. Then consider only those items whose global support count (sum of local support count of each item in candidate set) satisfies minimum support count, this results in globally frequent item-set. So, in FDM we will be finding both locally frequent item-set as well as globally frequent.

In this paper we discuss the idea based on one scenario. Consider in a region there are 3 stores s1, s2 & s3. Here the supplier belongs to a lays company wants to perform analysis to find which flavors of lays are purchased more frequently with respect to each store. So, only those flavors which are sold frequently will be supplied next. Here in this scenario we cannot use normal A-priori mining algorithm, because it gives frequent item-set globally for that region. Since we need to find frequently sold lays flavors with respect to each store we need to find locally frequent lays flavors first and then globally frequent lays flavors by performing union on them. So, we will use FDM in this paper. Here, first let us try to understand normal mining algorithm and FDM with their difference below.

2 Normal Mining Algorithm using A-priori property

This is the most influential algorithm used in finding the frequent item-sets which employs iteration approach call level-wise search. Were, frequent-k item-sets are used to derive frequent-(k+1) item-sets. Like, if L_1 is frequent-1 item-set using which we derive L_2 i.e frequent-2 item-set and then L_3 so on.. Until there are no frequent item-sets. To find each L_k entire database has to be scanned. So, to improve the efficiency of level-wise generation of frequent item-set we use another property called A-priori property.

A-priori property defines 2 important conditions:

i) If an item-set is said to be frequent then all of its non-empty subset must also be frequent.

Ex: If item-set I = {milk, bread} is frequent-2 then, milk & bread must be frequent-1.

An item-set I is said to be frequent if it satisfies minimum support threshold. This defines minimum no. of transactions in the database that must contain the items in the item-set I. If I doesn't satisfy minimum support then it is not frequent.

ii) A-priori property is based on another most important property called anti-monotone property. Which states that if a set fails a test, then all of its superset will also fail the test.

Ex: If an item-set I = {milk, bread} fails to satisfy. minimum support to be frequent-2, then all of its superset like {milk, bread, egg} or {milk, bread,

biscuit} also fails to support minimum support threshold to be frequent-3 because {milk, bread} are not frequent-2 so it cannot be frequent-3.

A-priori algorithm involves 2 steps:

i) Join step:
To find L_k, first we need to join L_{k-1} to itself (L_{k-1} natural join L_{k-1}). The resultant is represented as C_k i.e candidate K item-Set.

ii) Prune step:
C_k is the superset of L_k. Its members may or may not be Frequent but all the frequent-K item-sets must be in C_k. After finding C_k by joining L_{k-1} to itself we scan the database to find the count of the members in C_k. Then, those item-sets in C_k which satisfies minimum support count are considered as members of L_k.

Example: Let D be the database of 18 transactions having items {1, 2, 3, 4, 5} and support S = 1/3.
D = {12, 12345, 124, 1245, 14, 145, 235, 24, 24, 1234, 134, 23, 234, 2345, 1234, 124, 134, 23}
Min_sup_count = 1/3 X 18 = 6

Iteration 1:
First we need to find C_1 (candidate-1 item-set)

C_1 = {1(11), 2(14), 3(10), 4(14), 5(5)}

Now, consider only those items from C_1 which satisfies min_sup_count to obtain L_1 (frequent-1 item-set)

L_1 = {1, 2, 3, 4}

Iteration 2:
To obtain C_2 we need to join L_1 to itself.

C_2 = {12(7), 13(5), 14(10), 23(8), 24(10), 34(7)}

L_2 = {12, 14, 23, 24, 34}

Iteration 3:
To obtain C_3 we need to join L_2 to itself.

C_3 = {123(3), 124(6), 134(5), 234(5)}

L_3 = {124}

Hence, frequent item-sets generated are {1, 2, 3, 4, 12, 14, 23, 24, 34, 124}.

3 Fast Distributed Mining (FDM)

The FDM algorithm proceeds as follows:

1) **Initialization**: It is assumed that the players have already jointly calculated F_s^{k-1}. The goal is to proceed and calculate F_s^k.

2) **Candidate Sets Generation**: Each player P_m computes the set of all (k − 1)-item-sets that are locally frequent in his site and also globally frequent; namely, P_m computes the set $F_s^{(k-1,\ m)} \cap F_s^{(k-1)}$. He then applies on that set the A-priori algorithm in order to generate the set $B_s^{(k,\ m)}$ of Candidate k-item-sets.

3) **Local Pruning**: For each $X \in B_s^{(k,\ m)}$, P_m computes Supp (X_m). He then retains only those item-sets that are locally s-frequent. We denote this collection of item-sets by $C_s^{(k,\ m)}$.

4) **Unifying the candidate item-sets**: Each player broadcasts his $C_s^{(k,\ m)}$ and then all players compute $C_s := U\ C_s^{(k,\ m)}$ for m = 1 to M.

5) **Computing local support**: All players compute the local supports of all item-sets in C_s^k.

(6) **Broadcast Mining Results**: Each player broadcasts the local supports that he computed. From that, everyone can compute the global support of every item-set in C_s^k. Finally, F_s^k is the subset of C_s^k that consists of all globally s-frequent item-sets.

Example:

Let D be a database of N = 18 item-sets over a set of L = 5 items, A = {1, 2, 3, 4, 5}. It is partitioned between M = 3 players, and the corresponding partial databases are:

D_1 = {12, 12345, 124, 1245, 14, 145, 235, 24, 24}
D_2 = {1234, 134, 23, 234, 2345}
D_3 = {1234, 124, 134, 23}

For example, D_1 includes N_1 = 9 transactions, the third of which (in lexicographic order) consists of 3 items — 1, 2 and 4.
Setting s = 1/3, an item-set is s-frequent in D if it is supported by at least 6 = sN of its transactions. In this case,

F_s^1 = {1, 2, 3, 4}
F_s^2 = {12, 14, 23, 24, 34}
F_s^3 = {124}
$F_s^4 = Fs^5 = \emptyset$

F_s represent all the frequent item-sets generated

$F_s = F_s^1\ U\ F_s^2\ U\ F_s^3$

Iteration 1: All the three players compute $C_s^{(1, m)}$ of all 1-item-sets that are locally frequent at their partial databases

$C_s^{(1, 1)} = \{1, 2, 4, 5\}$, $C_s^{(1, 2)} = \{1, 2, 3, 4\}$, $C_s^{(1, 3)} = \{1, 2, 3, 4\}$

Now, C_s^1 that is candidate - 1 item-sets is obtained by performing union,

$C_s^1 = C_s^{(1, 1)} \cup C_s^{(1, 2)} \cup C_s^{(1, 3)}$
$C_s^1 = \{1, 2, 3, 4, 5\}$

From C_s^1 consider only that item which satisfies minimum support 6 to get F_s^1 that are frequent – 1 item-sets.

$F_s^1 = \{1, 2, 3, 4\}$

Iteration 2:

$C_s^{(2, 1)} = \{12, 14, 24\}$
$C_s^{(2, 2)} = \{13, 14, 23, 24, 34\}$
$C_s^{(2, 3)} = \{12, 13, 14, 23, 24, 34\}$

$C_s^2 = C_s^{(2, 1)} \cup C_s^{(2, 2)} \cup C_s^{(2, 3)}$
$C_s^2 = \{12, 13, 14, 23, 24, 34\}$

$F_s^2 = \{12, 14, 23, 24, 34\}$

Iteration 3:

$C_s^{(3, 1)} = \{124\}$
$C_s^{(3, 2)} = \{234\}$
$C_s^{(3, 3)} = \{124\}$

$C_s^3 = C_s^{(3, 1)} \cup C_s^{(3, 2)} \cup C_s^{(3, 3)}$
$C_s^3 = \{124, 234\}$

$F_s^3 = \{124\}$

4 Difference between FDM and Normal A-priori mining Algorithm

In the previous section we saw the examples of both A-priori and FDM. Were, A-priori is applied on all the transactions at once to find the item-sets which satisfies global support. Were as in case of FDM which is distributed mining, transactions will be divided into partitions with respect to players or stores. Then, we find the item-sets which are locally frequent with respect to each store and then perform the union of these locally frequent item-sets to obtain candidate-K item-set (C_k), which will be broadcasted to all the other stores to calculate the local support count of all the item-sets in C_k. Now find the sum of the local support counts of each item in C_k from all the store and consider only those item-sets from C_k which satisfies global support.

So, in FDM we find item-sets which satisfies locally and item-sets which satisfy globally. Were, in A-priori we find item-sets which are globally frequent.

5 Advantages of FDM over A-priori (normal mining Algorithm) and its application with respect to supplier in market basket analysis

To explain how FDM will be helpful with respect to the supplier for supplying the items to the stores based on the previous history of items sold in each store. Let us take an example, consider a region (area) in which there are 3 stores S_1, S_2 & S_3. The supplier belongs to lay's company and he wants to know what flavors of lays are sold frequently so that he can supply only those flavors which are frequently sold. For illustration let us consider 3 flavors "Lays salted" as L_s, "Lays australian" as L_a and "Lays indian" as L_i.

Consider in 3 stores S_1, S_2 & S_3 100 transactions has been carried out in a week and consider each store is supplied with 60 packs of lays, 3 flavors of 20 packs each. Here we will be using the concept of hierarchical data mining so that different support can be applied at different level. The steps are as followed:

1) First we need to find out of 100 transactions weather lays is frequent or not for the given minimum support, through which we can find how many packs of lays are sold in that particular store in a week. Assuming that each transaction can have only one pack of lays instantly.

Ex: min_sup = 40% then, min_sup_count = 0.4 x100 = 40. If out of 100 transactions, in S_1 if 45 transactions contains lays that means 45 packs of lays has been sold, in S_2 if 51 packs are sold and in S_3 if 40 packs are sold. Then lays is frequently sold in all the 3 stores.

2) After finding no. of packs sold by each store, now we need to find out of those no. of packs sold how many packs belongs to each of the 3 flavors. So that we can find what flavors are frequent for the given minimum support.

Ex: In S_1 out of 45, let L_s=20, L_a =20, L_i=5.
In S_2 out of 51, let L_s=15, L_a=16, L_i=20.
In S_3 out of 40, let L_s=5, L_a=20, L_i=15.

Since 20 packs of each flavor is sold to each store, let min_sup = 60%

So, min_sup_count = 0.6 X 20 = 12. That means for a particular flavor to be frequent minimum of 12 packs of that flavor has to be sold. So, in S_1 frequent flavors are L_s & L_a, in S_2 frequent flavors are L_s, L_a & L_i, in S_3 frequent flavors are L_a & L_i.

Note - In step 1 we used min_sup as 40% to find weather lays is frequently sold or not, in step 2 we used min_sup as 60% to find which flavors of lays are frequent in store. Since we used two different min_sup at two different steps is known as hierarchical data mining.

The above details of no. of packs that are sold with respect to each store and with respect to each flavor in each store is

shown in below table. In the table for example 20(15) means out of 20 packs of lays which is been supplied 15 are sold.

Lays Flavor's/ Stores	S_1	S_2	S_3
L_s	20(20)	20(15)	20(5)
L_a	20(20)	20(16)	20(20)
L_i	20(5)	20(20)	20(15)

Supplier knows that item lays is frequent in all the stores S_1, S_2 & S_3. Now, supplier needs to know which are the frequent flavors that are sold in these stores were the lays is frequent. Assume min_sup = 60% and item-set I = $\{L_s, L_a, L_i\}$. Since min_sup = 60%, min_sup_count = 0.6 X 20 = 12. So, if a flavor is said to be frequent in a store then minimum of 12 packs of that flavor has to be sold by that store.

We apply FDM, first to know which flavors of lays are locally frequent with respect to each store and then find the flavors which are globally frequent by performing union. The procedure is as follows:

$C_s^{(1, S1)} = \{L_s, L_a\}$

$C_s^{(1, S1)}$, means frequent-1 lays flavors in store S_1 which are locally s-frequent that is L_s and L_a because in S_1 "Lays salted" of 20 packs and "Lays australian" of 20 packs are sold which satisfies min_sup_count = 12 packs.

$C_s^{(1, S2)} = \{L_s, L_a, L_i\}$

$C_s^{(1, S3)} = \{L_a, L_i\}$

So, these are the lays flavors which are frequently sold locally in S_1, S_2 & S_3. Now supplier can supply only those flavors.

To know the flavors which are globally frequent perform the union of $C_s^{(1, S1)}$, $C_s^{(1, S2)}$ & $C_s^{(1, S3)}$ to get C_s^1 which is candidate-1 item-set.

$C_s^1 = C_s^{(1, S1)} \cup C_s^{(1, S2)} \cup C_s^{(1, S3)}$
$C_s^1 = \{L_s, L_a, L_i\}$

This C_s^1 will be broadcasted to all the stores S_1, S_2 & S_3 to find the local support count of all the items in each store. Now, each store will broadcast the local support count it computed for each item in C_s^1 to all the neighboring stores in the region. All the stores compute the sum of local support count with respect to each item in C_s^1. Then consider only those items from C_s^1 which satisfies global support count which gives F_s^1.

$C_s^{(1, count(S1))} = \{L_s(20), L_a(20), L_i(5)\}$

Here, $C_s^{(1, count(S1))}$ means the local support count of all the items in $C_s(1)$ with respect to S_1.

$C_s^{(1, count(S2))} = \{L_s(15), L_a(16), L_i(20)\}$

$C_s^{(1, count(S3))} = \{L_s(5), L_a(20), L_i(15)\}$

$C_s^{(1, count)} = C_s^{(1, count(S1))} \cup C_s^{(1, count(S2))} \cup C_s^{(1, count(S3))}$

$C_s^{(1, count)} = \{L_s(40), L_a(56), L_i(40)\}$

Global min_sup = 40%, so global min_sup_count = 0.4 X 100 = 40. So, consider only those flavors from $C_s^{(1, count)}$ which satisfies global min_sup_count 40 to get globally frequent lays flavors F_s^1.
In $C_s^{(1, count)}$ all the lays flavors satisfy global min_sup_count 40,

$F_s^1 = \{L_s, L_a, L_i\}$.

In FDM most important part is finding the item-sets which are locally frequent which helps the supplier to supply only those items which are locally frequent in those stores. Globally frequent item-sets are found just to keep the record for each region, which are the frequently sold items in that region. But for supplying items it is better to consider locally frequent item-set and not globally.

What if instead of using FDM if we have applied or used normal A-priori mining algorithm?

As we know that normal A-priori mining is applied on all the transactions from different stores at once. If normal A-priori algorithm was applied on all the transactions from stores S_1, S_2 & S_3 then we would not be able to find item-sets which are locally frequent like $C_s^{(1, S1)}$, $C_s^{(1, S2)}$ & $C_s^{(1, S3)}$ like in FDM. Instead we would have ended up directly with item-sets which are globally frequent like F_s^1.

Let us apply normal A-priori algorithm for the above transactions from S_1, S_2 & S_3. Given that each store all supplied with 60 packs of lays with 3 flavors of 20 packs each. So that in that region 180 packs are supplied, 60 packs of each flavor.

Here, transactions from all the stores S_1, S_2 & S_3 are merged to-gather so we will have 300 transactions (100 transactions from each store). Out of these 300 transactions 136 transactions contains lays (45 from S_1, 51 from S_2 & 40 from S_3) that means totally 136 packs of lays are sold. Out of these 136 packs we need to find how many packs with respect to each flavor are sold. That will be $L_s = 40$, $L_a = 56$, $L_i = 40$.

Normal A-priori algorithm is applied as follows assuming global min_sup = 60%. Globally 60 packs of each flavor is supplied. So, min_sup_count = 0.6 X 60 = 36 packs. A particular lays flavor is said to be globally frequent if a minimum of 36 packs are sold globally.

Item-set I = $\{L_s, L_a, L_i\}$

First we need to find candidate-1 item-set C_s^1 which contains all the items in item-set I and their global count as follows

C_s^1	Item	count
	L_s	40
	L_a	56
	L_i	40

From the above table that is candidate-1 item-set C_s^1 consider only those items which satisfy min_sup_count = 36 to obtain globally frequent-1 item-set F_s^1,

$F_s^1 = \{L_s, L_a, L_i\}$

Now, in this case supplier only know these are the flavors of lays which are frequently sold in that region but, he doesn't know what flavors of lays are sold frequently in each store. So, supplier will again supply 60 packs to each store with 3 flavors of 20 packs each. That means supplier will again supply "lays Indian" L_i flavor of 20 packs to store S_1 even though L_i flavor is not frequently sold in S_1 because of which the item reach its expiry date since it will be unsold and it is a loss to the company if it is the case with respect to thousands or many stores.
So, we can conclude that FDM has an upper hand than Normal A-priori mining algorithm.

6 Identifying all (s; c)-association rules

Once the set F of all s-frequent item-sets is found, we may Proceed to look for all (s, c) – association rules (rules with support at least sN and confidence at least c) For X, Y ∈ F_s, where X intersection Y = Ø, the corresponding association rule X ⇒ Y has confidence at least c if and only if,

$supp(X \cup Y) / supp(X) = c$

or

$$C_{X,Y} := \sum_{m=1}^{M} (supp_m(X \cup Y) - c \cdot supp_m(X)) >= 0$$

7 Conclusion

Here in the example which we took to explain the application with respect to the supplier using FDM, we made an assumption that each transaction contains only one pack of lays. Like in case of S_1 were 45 transactions out of 100 contained lays, means 45 packs of lays are sold by S_1. If a transaction can have more than one lays pack sold, in case were a person purchase more than one lays pack. Then the no. of lays pack sold by that store can be greater than the no. of transactions that contain lays item. In this case how to find the no. of lays packs that are sold by the store?

While finding globally frequent item-set, each store broadcast the local support count of items in candidate-1 item-set $C_s^1 = \{L_s, L_a, L_i\}$ to all other store's to find the global support count by performing union on them. Here, the local support count of items in C_s^1 are private with respect to each store so when they are broadcasted to other stores it must be sent in a secured manner, so that the local support counts of items in C_s^1 with respect to a particular store will not be known to other stores. Union must be performed securely.

In store S_1 consider 30 packs of lays are sold which is less than min_sup_count = 40. So, this item is not frequent and supplier will not supply lays to that store. But, out of these 30 packs which are sold, if 20 are L_s, 5 are L_a & 5 are L_i. This means "Lays salted" L_s flavor which are supplied to store S_1 with 20 packs has sold all the 20 packs so L_s flavor is frequent in store S_1. Even though L_s is frequent supplier will fail to supply this flavor to that store S_1.

Acknowledgment

My sincere thanks to sivagama sundari G for her guidance in publishing the paper.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In VLDB, pages 487–499, 1994.

[2] D.W.L. Cheung, J. Han, V.T.Y. Ng, A.W.C. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In PDIS, pages 31–42, 1996.

[3] D.W.L Cheung, V.T.Y. Ng, A.W.C. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. IEEE Trans. Knowl. Data Eng., 8(6):911–922, 1996.

[4] R. Srikant and R. Agrawal. Mining generalized association rules. In VLDB, pages 407–419, 1995.

[5] T. Tassa and D. Cohen. Anonymization of centralized and distributed social networks by sequential clustering. IEEE Transactions on Knowledge and Data Engineering, 2012.

[6] T. Tassa. Secure Mining of Association Rules in Horizontally Distributed Databases. IEEE Transactions on Knowledge and Data Engineering, 2014.