Algorithme Apriori

algorithme Apriori est donné par R. Agrawal et R. Srikant en 1994 pour trouver des ensembles d'éléments fréquents dans un ensemble de données pour la règle d'association booléenne. Le nom de l'algorithme est Apriori car il utilise une connaissance préalable des propriétés des ensembles d'éléments fréquents. Nous appliquons une approche itérative ou une recherche par niveau où k itemsets fréquents sont utilisés pour trouver k+1 itemsets.

Pour améliorer l'efficacité de la génération par niveau d'ensembles d'éléments fréquents, une propriété importante est utilisée, appelée *propriété Apriori*, qui aide en réduisant l'espace de recherche.

Propriété Apriori –

Tous les sous-ensembles non vides de l'ensemble d'éléments fréquents doivent être fréquents. Le concept clé de l'algorithme d'Apriori est son anti-monotonie de la mesure de soutien. Apriori suppose que

```
Tous les sous-ensembles d'un ensemble d'éléments fréquents doivent être
fréquents (propriété Apriori).
Si un ensemble d'éléments est peu fréquent, tous ses surensembles seront
peu fréquents.
```

Considérez l'ensemble de données suivant et nous trouverons des ensembles d'éléments fréquents et générerons des règles d'association pour eux.

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

le nombre minimum de support est de 2
la confiance minimum est de 60%

Détermination des itemsets fréquents

Étape 1 : K=1

(I) Créez un tableau contenant le nombre de supports de chaque élément présent dans l'ensemble de données - Appelé **C1 (ensemble candidat)**

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

Étape 2 : K=2

- Générez l'ensemble candidat C2 à l'aide de L1 (c'est ce qu'on appelle l'étape de jointure). La condition pour joindre L k-1 et L k-1 est qu'il doit avoir (K-2) éléments en commun.
- Vérifiez que tous les sous-ensembles d'un ensemble d'éléments sont fréquents ou non et s'ils ne sont pas fréquents, supprimez cet ensemble d'éléments.
- Trouvez maintenant le nombre de supports de ces ensembles d'éléments en effectuant une recherche dans l'ensemble de données.

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) comparer le nombre de support du candidat (C2) avec le nombre de support minimum (ici min_support=2 si support_count de l'élément de l'ensemble candidat est inférieur à min_support puis supprimer ces éléments) cela nous donne l'ensemble d'éléments 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

Étape 3:

- Générez l'ensemble candidat C3 à l'aide de L2 (étape de jointure). La condition pour joindre L k-1 et L k-1 est qu'il doit avoir (K-2) éléments en commun. Donc ici, pour L2, le premier élément doit correspondre.
  Ainsi, l'ensemble d'éléments généré en joignant L2 est {I1, I2, I3}{I1, I2, I5}{I1, I3, i5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}

- Vérifiez si tous les sous-ensembles de ces ensembles d'éléments sont fréquents ou non et sinon, supprimez cet ensemble d'éléments. (Ici, les sous-ensembles de {I1, I2, I3} sont {I1, I2},{I2, I3}, {I1, I3} qui sont fréquents. Pour {I2, I3, I4}, le sous-ensemble {I3, I4} n'est pas fréquent, alors supprimez-le. De même, vérifiez pour chaque ensemble d'éléments)
- trouver le nombre de supports de ces ensembles d'éléments restants en effectuant une recherche dans l'ensemble de données.

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

(II) Comparez le nombre de support du candidat (C3) avec le nombre de support minimum (ici min_support=2 si support_count de l'élément de l'ensemble candidat est inférieur à min_support puis supprimez ces éléments) cela nous donne l'ensemble d'éléments L3.

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

Étape 4:

- Générez l'ensemble candidat C4 à l'aide de L3 (étape de jointure). La condition pour joindre L k-1 et L k-1 (K=4) est qu'ils doivent avoir (K-2) éléments en commun. Donc ici, pour L3, les 2 premiers éléments (éléments) doivent correspondre.
- Vérifiez que tous les sous-ensembles de ces ensembles d'éléments sont fréquents ou non (Ici, l'ensemble d'éléments formé en joignant L3 est {I1, I2, I3, I5} donc son sous-ensemble contient {I1, I3, I5}, ce qui n'est pas fréquent). Donc pas d'éléments en C4

- Nous nous arrêtons ici car aucun ensemble d'éléments fréquents n'est trouvé plus loin

Elaboration des règles de confiances

Ainsi, nous avons découvert tous les ensembles d'items fréquents. Maintenant, la génération d'une règle d'association forte entre en scène. Pour cela, nous devons calculer la confiance de chaque règle.

Confiance –

Une confiance de 60 % signifie que 60 % des clients qui ont acheté du lait et du pain ont également acheté du beurre.

```
Confidence(A->B)=Support_count(AUB)/Support_count(A)
```

Donc ici, en prenant un exemple d'ensemble d'éléments fréquents, nous allons montrer la génération de règles.
Itemset {I1, I2, I3} //de L3 les

règles SO peuvent être
[I1^I2]=>[I3] //confidence = sup(I1^I2^I3)/sup(I1^I2) = 2/4* 100=50%
[I1^I3]=>[I2] //confiance = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%*
[I2^I3]=>[I1] //confiance = sup(I1^I2^I3)/sup(I2^I3) = 2/4100=50%
[I1]=>[I2^I3] //confiance = sup(I1^I2^ I3)/sup(I1) = 2/6*100=33%*
[I2]=>[I1^I3] //confiance = sup(I1^I2^I3)/sup(I2) = 2/7100= 28%
[I3]=>[I1^I2] //confiance = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%

Donc, si la confiance minimale est de 50 %, alors les 3 premières règles peuvent être considérées comme des règles d'association fortes, mais comme la confiance min est de 60% alors cette règle forte.

Limites de l'algorithme Apriori

L'algorithme Apriori peut être lent. La principale limitation est le temps nécessaire pour contenir un grand nombre d'ensembles candidats avec des ensembles d'éléments très fréquents, un support minimum faible ou de grands ensembles d'éléments, c'est-à-dire que ce n'est pas une approche efficace pour un grand nombre d'ensembles de données. Par exemple, s'il y a 10^4 à partir d'ensembles de 1 éléments fréquents, il doit générer plus de 10^7 candidats en longueur 2 qui à leur tour seront testés et accumulés. De plus, pour détecter des motifs fréquents de taille 100, c'est-à-dire v1, v2... v100, il doit générer 2^100 ensembles d'éléments candidats qui entraînent une génération coûteuse et fastidieuse de candidats. Ainsi, il vérifiera de nombreux ensembles à partir d'ensembles d'éléments candidats, il analysera également la base de données plusieurs fois à plusieurs reprises pour trouver des ensembles d'éléments candidats. Apriori sera très faible et inefficace lorsque la

capacité mémoire est limitée avec un grand nombre de transactions. [La source :https://arxiv.org/pdf/1403.3948.pdf]

Ressources:

https://www.geeksforgeeks.org/apriori-algorithm/