Chapitre 5 : Règles d'association

Introduction aux Règles d'Association

Les règles d'association sont un élément fondamental de l'analyse de données, particulièrement utiles pour découvrir des relations intéressantes et non triviales dans de grands ensembles de données. Elles permettent d'identifier des corrélations entre différents éléments dans une base de données, ce qui peut révéler des informations précieuses et éventuellement exploitables. Une règle d'association est une implication du type "si X, alors Y", où X et Y sont des ensembles d'éléments.

Importance des Règles d'Association :

Les règles d'association sont utilisées pour explorer les modèles de comportement des clients, pour recommander des produits pertinents, pour optimiser les processus de fabrication, et dans bien d'autres domaines. Elles peuvent révéler des tendances cachées et aider à prendre des décisions éclairées.

Utilisations Pratiques :

Les règles d'association sont appliquées dans divers domaines. Dans le domaine du marketing, elles fournissent à comprendre les habitudes d'achat des clients et à cibler les offres spécifiques. Dans la recommandation de produits, elles sont pour suggérer des articles complémentaires. Elles sont également précieuses dans l'optimisation des chaînes d'approvisionnement, la bioinformatique, etc.

Définition des Termes Clés :

Itemset (Ensemble d'Articles) : Un groupe d'articles qui peuvent apparaître ensemble, par exemple, {pain, lait, œufs}.

Règle d'Association : Une implication du type "Si A, alors B", où A et B sont des ensembles d'articles.

Support : La fréquence d'apparition d'un ensemble d'articles dans l'ensemble de données.

Confiance : La probabilité conditionnelle que la règle soit vraie, donnée la présence d'un ensemble d'articles.

Lift : Mesure l'importance d'une règle d'association par rapport à la fréquence d'apparition des articles individuels.

Algorithme a priori

```
Algorithme Apriori
Entrées : base de données de transactions D, seuil de support minimum σ
Sorties : ensemble d'ensembles fréquents FreqSets
1. C1 ← Ensemble des motifs de taille 1
2. FreqSets ← Ensemble vide
3. Pour chaque motif c dans C1 :
   Calculer le support de c dans la base D
   Si support(c) ≥ σ, alors
      Ajouter c à FreqSets
4. i ← 2
5. Tant que Ci-1 n'est pas vide faire :
   Ci ← Générer des combinaisons de motifs de taille i à partir de Ci-1
   Pour chaque motif c dans Ci :
      Calculer le support de c dans la base D
      Si support(c) ≥ σ, alors
         Ajouter c à FreqSets
   i ← i + 1
6. Retourner FreqSets
```

L'algorithme Apriori est l'un des algorithmes les plus utilisés pour extraire des règles d'association à partir de grands ensembles de données. Il fonctionne en trois étapes principales :

- Génération des Ensembles Fréquents : Dans cette étape, les ensembles d'éléments (item sets) ayant un support supérieur ou égal à un seuil prédéfini sont identifiés comme des ensembles fréquents.
- Génération des Règles Potentielles : En utilisant les ensembles fréquents, des règles d'association potentielles sont générées en utilisant différentes combinaisons.
- Sélection des Règles Finales : Les règles potentielles sont filtrées en fonction du seuil de confiance afin de ne conserver que celles considérées comme significatives.

Avantages et inconvénients de l'algorithme Apriori

Avantages

- Simplicité : Facile à comprendre et à implémenter, adapté aux non-experts en apprentissage automatique.
- Économie de temps : Élimine rapidement les ensembles peu fréquents, économisant du temps de calcul.
- Interprétabilité : Génère des règles d'association simples et compréhensibles pour la prise de décision et la recommandation.

Inconvénients

- Explosion combinatoire : Souffre de l'explosion combinatoire dans les bases de données volumineuses.
- Seuil de support nécessaire : Exige une sélection minutieuse du seuil de support, impactant la qualité des résultats.
- Difficulté avec les transactions longues : Moins efficace avec des transactions contenant de nombreux articles, augmentant le temps de calcul.

Exercice sur l'algorithme Apriori :

Considérons un ensemble de transactions d'achat dans un supermarché. Chaque transaction est représentée sous forme d'ensembles d'articles achetés. Vous bénéficiez des transactions suivantes :

Transaction 1: {pain, lait, œufs}

Transaction 2: {pain, couches, jus, œufs}

Transaction 3: {lait, couches, jus, soda}

Transaction 4: {pain, lait, couches, jus}

Transaction 5: {pain, lait, couches, soda}

Vous souhaitez utiliser l'algorithme A priori pour trouver les ensembles d'articles fréquents avec un seuil de support minimum de 3 transactions. Suivez les étapes de l'algorithme Apriori pour répondre aux questions suivantes :

- Recherche des K-items fréquents :

Trouvez les ensembles d'articles fréquents de taille 1 (c'est-à-dire les articles individuels) et calculez leur support.

Utilisez les ensembles d'articles fréquents de taille 1 pour générer les ensembles d'articles fréquents de taille 2 et calculez leur support.

Utilisez les ensembles d'articles fréquents de taille 2 pour générer les ensembles d'articles fréquents de taille 3 et calculez leur support.

- Génération des règles d'association :

À partir des ensembles d'articles fréquents, générerez toutes les règles d'association possibles avec un seuil de confiance de 60 %.

Solution

La base de données formelle (O, P, R) est la suivante :

R	pain	lait	œufs	couches	jus	soda
X1	1	1	1	0	0	0
X2	1	0	1	1	1	0
X3	0	1	0	1	1	1
X4	1	1	0	1	1	0
X5	1	1	0	1	0	1

Avec :

O : X1, X2, X3, X4, X5

P : pain, lait, œufs, couches, jus, soda

Relation R = X1Rpain = 1

Etape 1 : i=1

C1 = {pain, lait, œufs, couches, jus, soda}

Support (Pain) =4

Support (Lait) =4

Support (Œufs) =2 < 3

Support (Couches) =4

Support (Jus) =3

Support (soda) =2 < 3

F1 = {pain, lait, couches, jus}

Etape 2 : i=2

C2 = {pain lait, pain couche, pain jus, lait couche, lait jus, couches jus}

Support (pain lait) =3

Support (pain couche) =3

Support (pain jus) =2 < 3

Support (lait couche) =3

Support (lait jus) =2 < 3

Support (couches jus) =3

F2 = {pain lait, pain couches, lait couches, couches jus}

Etape 3 : i=3

C3 = {pain lait couche, pain lait jus, pain couches jus}

Support (pain lait couche) = 2 < 3

Support (pain lait jus) = 1 < 3

Support (pain couches jus) = 2 < 3

F3 = { Ø }

F1∪F2∪F3 = {pain, lait, couches, jus, pain lait, pain couches, lait couches, couches jus}

Règles d'association avec un seuil de confiance de 60 % :

Degré d'un motif >=2

Pour générer des règles d'association, nous utilisons les ensembles d'articles fréquents :

pain→lait : Confiance = support (pain lait)/ support (pain) = ¾ = 75% …

Motif fréquent	Règle	Confiance	Qualité
pain lait	pain→lait	3/4 = 75%	Bonne qualité
	lait→pain	3/4 = 75%	Bonne qualité
pain couches	pain→couches	3/4 = 75%	Bonne qualité
	couches→ pain	3/4 = 75%	Bonne qualité
lait couches	lait→couches	3/4 = 75%	Bonne qualité
	couches→ lait	3/4 = 75%	Bonne qualité
couches jus	couches→jus	3/4 = 75%	Bonne qualité
	jus→ couches	3/3 = 100%	Meilleure qualité

Toutes les règles générées sont acceptées

Algorithme FP-Tree

L'algorithme FP-growth (Frequent Pattern Growth) est une alternative à priori pour extraire des règles d'association. Il est plus efficace pour traiter de grandes bases de données. Voici les étapes de l'algorithme :

- Construction de l'Arbre FP-tree : Dans cette étape, un arbre de fréquence (FP-tree) est construit en utilisant les transactions. Cet arbre représente les ensembles fréquents et simplifie le processus d'extraction.
- Extraction des Ensembles Fréquents : Les ensembles fréquents sont extraits de l'arbre FP-tree en utilisant un processus de projection.
- Génération des Règles d'Association : Les règles d'association sont générées en utilisant les ensembles fréquents extraits.

Avantages et inconvénients :

Avantages de l'arbre FP (FP-Tree) :

- Efficacité de stockage : Réduit la mémoire nécessaire pour stocker de grandes bases de données transactionnelles.
- Efficacité de construction : Plus rapide que les méthodes comme Apriori pour la création de l'arbre initial.
- Recherche rapide : La recherche d'item sets fréquents et de règles d'association est généralement plus rapide une fois l'arbre FP construit.

Inconvénients de l'arbre FP :

- Complexité de construction : La construction initiale peut être coûteuse en temps et en mémoire, surtout pour de grandes bases de données.
- Mise à jour difficile : Les modifications dans la base de données peuvent nécessiter une reconstruction complète de l'arbre FP.
- Taille de l'arbre : Dans certaines situations, l'arbre FP peut devenir très grand, ce qui peut consommer beaucoup de mémoire.

Exercice sur l'algorithme FP-growth :

L'algorithme FP-Growth (Frequent Pattern Growth) est utilisé pour extraire des motifs fréquents à partir de données transactionnelles en utilisant une structure de données appelée FP-Tree (Frequent Pattern Tree).

Supposons que vous ayez le jeu de données de transactions suivant avec min support = 2 :

T1: {A, B, D}

T2: {B, C, D, E}

T3: {A, B, D, E}

T4: {A, C}

T5: {B, D}

Construisez l'arbre FP-Tree à partir de ces transactions.

Générez les motifs fréquents à partir de l'arbre FP-Tree.

Solution

Matrice de données formelle :

	A	B	C	D	E
T1	1	1	0	1	0
T2	0	1	1	1	1
T3	1	1	0	1	1
T4	1	0	1	0	0
T5	0	1	0	1	0

Maintenant nous devons compter la fréquence de chaque élément unique dans les transactions.

A : 3

B : 4

C : 2

D : 4

E : 2

Trions les éléments en fonction de leur fréquence décroissant : B, D, A, C, E.

Trions aussi les transactions :

T1: {B, D, A}

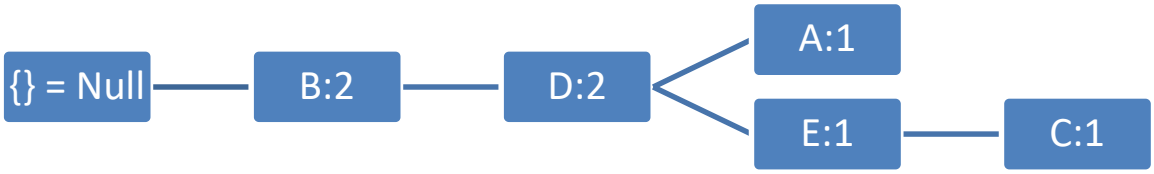
T2: {B, D, E, C}

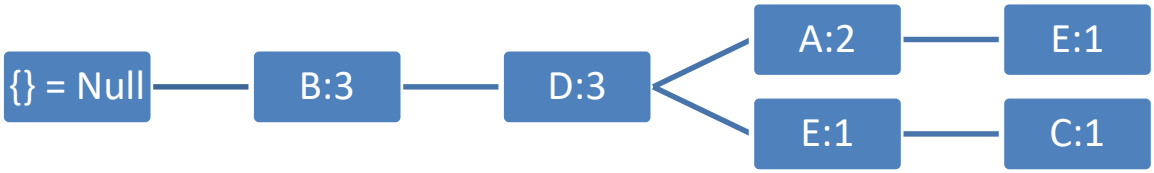
T3: {B, D, A, E}

T4: {A, C}

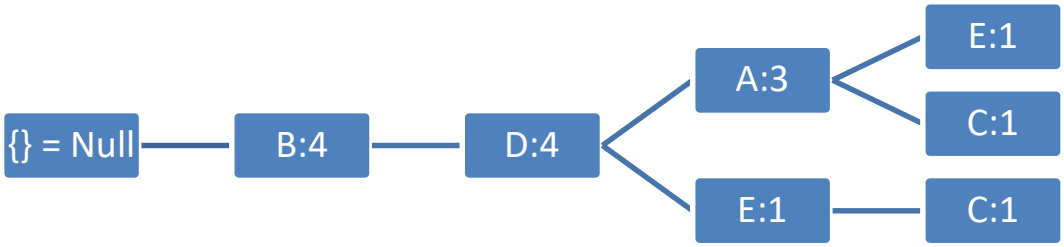
T5: {B, D}

Ensuite, construisons l'arbre FP-Tree en parcourant chaque transaction et en insérant les éléments triés dans l'arbre. Nous créons un nœud racine (FP-Tree) sans article associé.





Arbre finale:



Exploration de FP-Tree :

	base de modèle conditionnel	Arbre-FP conditionnel	modèle fréquent
E	{B, D, A: 1} {B, D: 1}	{B:2, D:2}	{B, E:2} {D, E:2} {B, D, E:2}
D	{B: 4}	{B:4}	{B, D:4}
C	{B, D, A: 1} {B, D, E: 1}	{B:2, D:2}	{B, C:2} {D, C:2} {B, D, C:2}

Donc l’ensemble des items fréquents :

A, B, C, D, E, BC, BD, BE, CD, DE, BDC, BDE.

En utilisant la même manière utilisée dans l’algorithme apriori, nous faisons extraire les règles d’association.

Comparaison et Discussion

La comparaison entre Apriori et FP-growth est basée sur les performances et la complexité. FP-growth est souvent plus rapide en raison de la structure de l'arbre FP-tree, mais il peut consommer plus de mémoire pour la construction de cet arbre. A priori, bien que plus simple, peut être moins efficace pour des bases de données volumineuses.