Intelligence Artificielle

Fouille de données : Règles d'association

Maria Malek

Département Informatiques

Les règles d’association - Terminologie

- Domaine décrit par une liste d’atomes appelée items.

Les règles d’association - Terminologie

- Domaine décrit par une liste d’atomes appelée items.
- Application : panier de ménagère dans un supermarché : [vin, fromage, chocolat]

Les règles d’association - Terminologie

- Domaine décrit par une liste d’atomes appelée items.
- Application : panier de ménagère dans un supermarché : [vin, fromage, chocolat]
- Un ensemble d’items est une suite d’items exprimée dans un ordre donné

Les règles d’association - Terminologie

- Domaine décrit par une liste d’atomes appelée items.
- Application : panier de ménagère dans un supermarché : [vin, fromage, chocolat]
- Un ensemble d’items est une suite d’items exprimée dans un ordre donné
- Une transaction est un ensemble d’items, exemples

Les règles d’association - Terminologie

- Domaine décrit par une liste d’atomes appelée items.
- Application : panier de ménagère dans un supermarché : [vin, fromage, chocolat]
- Un ensemble d’items est une suite d’items exprimée dans un ordre donné
- Une transaction est un ensemble d’items, exemples

  T1 [vin, fromage, viande]

Les règles d'association - Terminologie

- Domaine décrit par une liste d'atomes appelée items.
- Application : panier de ménagère dans un supermarché : [vin, fromage, chocolat]
- Un ensemble d'items est une suite d'items exprimée dans un ordre donné
- Une transaction est un ensemble d'items, exemples

  T1 [vin, fromage, viande]

  T2 [vin, fromage, chocolat]

Les règles d'association - Terminologie

- Domaine décrit par une liste d'atomes appelée items.
- Application : panier de ménagère dans un supermarché : [vin, fromage, chocolat]
- Un ensemble d'items est une suite d'items exprimée dans un ordre donné
- Une transaction est un ensemble d'items, exemples

  T1 [vin, fromage, viande]

  T2 [vin, fromage, chocolat]
- Un ensemble D de transactions correspond à un ensemble d'apprentissage

Les règles d’association - Objectif

- Objectif : chercher les associations à partir de D

Les règles d'association - Objectif

- Objectif : chercher les associations à partir de D

  T1 $vin \rightarrow fromage$

Les règles d'association - Objectif

- Objectif : chercher les associations à partir de D

T1 $vin \rightarrow fromage$

T2 $vinfromage \rightarrow jambon$

Les règles d’association - Définition

- Ensemble d’items fréquents : motif fréquent dans la base de transactions

Les règles d’association - Définition

- Ensemble d’items fréquents : motif fréquent dans la base de transactions

  minSupp un paramètre

Les règles d’association - Définition

- Ensemble d’items fréquents : motif fréquent dans la base de transactions

  minSupp un paramètre

- Trouver tous les ensembles d’items fréquents de longueurs différentes, exemple

Les règles d’association - Définition

- Ensemble d’items fréquents : motif fréquent dans la base de transactions

  minSupp un paramètre

- Trouver tous les ensembles d’items fréquents de longueurs différentes, exemple

  1. Si ABCD est un ensemble d’items fréquent

Les règles d'association - Définition

- Ensemble d'items fréquents : motif fréquent dans la base de transactions

  minSupp un paramètre

- Trouver tous les ensembles d'items fréquents de longueurs différentes, exemple
  1. Si ABCD est un ensemble d'items fréquent
  2. Construire la règle $AB \Rightarrow CD$ ssi $support(ABCD)/support(AB) \geq minConf$

Les règles d'association - Définition

- Ensemble d'items fréquents : motif fréquent dans la base de transactions

  minSupp un paramètre

- Trouver tous les ensembles d'items fréquents de longueurs différentes, exemple
  1. Si ABCD est un ensemble d'items fréquent
  2. Construire la règle $AB \Rightarrow CD$ ssi $support(ABCD)/support(AB) \geq minConf$
  3. minConf est un paramètre

Les règles d'association - Algorithmes

- Terminologie

Les règles d'association - Algorithmes

- Terminologie

  L_k est l'ensemble constitué des sous-ensembles d'items fréquents de longueur k.

Les règles d'association - Algorithmes

- Terminologie

  L_k est l'ensemble constitué des sous-ensembles d'items fréquents de longueur k.

  C_k est un ensemble constitué des sous-ensembles d'items candidats de longueur k, notons bien que $L_k \subset C_k$

Les règles d'association - Algorithmes

- Terminologie

  L_k est l'ensemble constitué des sous-ensembles d'items fréquents de longueur k.

  C_k est un ensemble constitué des sous-ensembles d'items candidats de longueur k, notons bien que $L_k \subset C_k$

- **Propriété** Soit X_k un sous-ensemble d'items fréquent, tous les sous-ensembles d'items contenus dans X_k et qui soient de longueurs inférieurs à k sont fréquents.

Les règles d'association - Algorithmes

- Terminologie

  L_k est l'ensemble constitué des sous-ensembles d'items fréquents de longueur k.

  C_k est un ensemble constitué des sous-ensembles d'items candidats de longueur k, notons bien que $L_k \subset C_k$

- **Propriété** Soit X_k un sous-ensemble d'items fréquent, tous les sous-ensembles d'items contenus dans X_k et qui soient de longueurs inférieurs à k sont fréquents.

  1. Si ABCD est un ensemble d'items fréquent

Les règles d'association - Algorithmes

- Terminologie

  L_k est l'ensemble constitué des sous-ensembles d'items fréquents de longueur k.

  C_k est un ensemble constitué des sous-ensembles d'items candidats de longueur k, notons bien que $L_k \subset C_k$

- **Propriété** Soit X_k un sous-ensemble d'items fréquent, tous les sous-ensembles d'items contenus dans X_k et qui soient de longueurs inférieurs à k sont fréquents.

  1. Si ABCD est un ensemble d'items fréquent
  2. ABC,ABD, BCD, AB,AC,BC,BD,CD,A,B,C,D les sont aussi.

Les règles d'association - Apriori

- Calculer L_1

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$
  2. TantQue $t \in D$

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$
  2. TantQue $t \in D$
     (a) $C_t \leftarrow sousEns(C_k, t)$

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$
  2. TantQue $t \in D$
     (a) $C_t \leftarrow sousEns(C_k, t)$
     (b) TantQue $c \in C_t$

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$
  2. TantQue $t \in D$
     (a) $C_t \leftarrow sousEns(C_k, t)$
     (b) TantQue $c \in C_t$
        i. c.count++

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$
  2. TantQue $t \in D$
     (a) $C_t \leftarrow sousEns(C_k, t)$
     (b) TantQue $c \in C_t$
        i. c.count++
  3. $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$
  2. TantQue $t \in D$
     (a) $C_t \leftarrow sousEns(C_k, t)$
     (b) TantQue $c \in C_t$
        i. c.count++
  3. $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$
  4. $k \leftarrow k + 1$

Les règles d'association - Apriori

- Calculer L_1
- $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$
  2. TantQue $t \in D$
     (a) $C_t \leftarrow sousEns(C_k, t)$
     (b) TantQue $c \in C_t$
         i. c.count++
  3. $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$
  4. $k \leftarrow k + 1$
- RETOURNER $\bigcup_k L_k$

Les règles d’association - Apriori - Suite

- L’algorithme apriori-gen, **la phase joindre** :

Les règles d'association - Apriori - Suite

- L'algorithme apriori-gen, **la phase joindre** :
  1. insert into C_k

Les règles d'association - Apriori - Suite

- L'algorithme apriori-gen, **la phase joindre** :
  1. insert into C_k
  2. select $p[1], p[2], ..p[k-1], q[k-1]$

Les règles d'association - Apriori - Suite

- L'algorithme apriori-gen, **la phase joindre** :
  1. insert into C_k
  2. select $p[1], p[2], ..p[k-1], q[k-1]$
  3. from p,q

Les règles d'association - Apriori - Suite

- L'algorithme apriori-gen, **la phase joindre** :
  1. insert into C_k
  2. select $p[1], p[2], ..p[k-1], q[k-1]$
  3. from p,q
  4. Where $p[1] = q[1]..p[k-2] = q[k-2]$,$p[k-1] < q[k-1]$

Les règles d'association - Apriori - Suite

- L'algorithme apriori-gen, **la phase joindre** :
  1. insert into C_k
  2. select $p[1], p[2], ..p[k-1], q[k-1]$
  3. from p,q
  4. Where $p[1] = q[1]..p[k-2] = q[k-2]$,$p[k-1] < q[k-1]$
- L'algorithme sousEns calcule le sous ensemble $C_t \subseteq C_k$, **la phase effacer**

Les règles d'association - Apriori - Suite

- L'algorithme apriori-gen, **la phase joindre** :
  1. insert into C_k
  2. select $p[1], p[2], ..p[k-1], q[k-1]$
  3. from p,q
  4. Where $p[1] = q[1]..p[k-2] = q[k-2]$,$p[k-1] < q[k-1]$
- L'algorithme sousEns calcule le sous ensemble $C_t \subseteq C_k$, **la phase effacer**
  1. Si $L_3 = \{\{123\}, \{124\}, \{134\}, \{135\}, \{234\}\}$,

Les règles d'association - Apriori - Suite

- L'algorithme apriori-gen, **la phase joindre** :
  1. insert into C_k
  2. select $p[1], p[2], ..p[k-1], q[k-1]$
  3. from p,q
  4. Where $p[1] = q[1]..p[k-2] = q[k-2]$,$p[k-1] < q[k-1]$
- L'algorithme sousEns calcule le sous ensemble $C_t \subseteq C_k$, **la phase effacer**
  1. Si $L_3 = \{\{123\}, \{124\}, \{134\}, \{135\}, \{234\}\}$,
  2. la phase $joindre$ donne comme résultat $C_4 = \{\{1234\}, \{1345\}\}$

Les règles d'association - Apriori - Suite

- L'algorithme apriori-gen, **la phase joindre** :
  1. insert into C_k
  2. select $p[1], p[2], ..p[k-1], q[k-1]$
  3. from p,q
  4. Where $p[1] = q[1]..p[k-2] = q[k-2]$,$p[k-1] < q[k-1]$
- L'algorithme sousEns calcule le sous ensemble $C_t \subseteq C_k$, **la phase effacer**
  1. Si $L_3 = \{\{123\}, \{124\}, \{134\}, \{135\}, \{234\}\}$,
  2. la phase $joindre$ donne comme résultat $C_4 = \{\{1234\}, \{1345\}\}$
  3. la phase $effacer$ donne le résultat: $C_4 = \{\{1234\}$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$
     (a) $C_t \leftarrow \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$
     (a) $C_t \leftarrow \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$
     (b) TantQue $c \in C_t$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$
     (a) $C_t \leftarrow \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$
     (b) TantQue $c \in C_t$
        i. c.count++

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$
     (a) $C_t \leftarrow \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$
     (b) TantQue $c \in C_t$
        i. c.count++
     (c) Si $C_t <> \phi$ Alors

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$
     (a) $C_t \leftarrow \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$
     (b) TantQue $c \in C_t$
        i. c.count++
     (c) Si $C_t <> \phi$ Alors
        i. $\hat{C}_k \leftarrow + < t.TID, C_t >$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$
     (a) $C_t \leftarrow \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$
     (b) TantQue $c \in C_t$
        i. c.count++
     (c) Si $C_t <> \phi$ Alors
        i. $\hat{C}_k \leftarrow + < t.TID, C_t >$
  3. $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$, $k \leftarrow k + 1$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$
     (a) $C_t \leftarrow \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$
     (b) TantQue $c \in C_t$
        i. c.count++
     (c) Si $C_t <> \phi$ Alors
        i. $\hat{C}_k \leftarrow + < t.TID, C_t >$
  3. $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$, $k \leftarrow k + 1$
- RETOURNER $\bigcup_k L_k$

Les règles d'association - AprioriTid

- Calculer L_1, $\hat{C}_1 \leftarrow D$, $k \leftarrow 2$
- TantQue $L_{k-1} <> \phi$
  1. $C_k \leftarrow apriori - gen(L_{k-1})$, $\hat{C}_k \leftarrow \phi$
  2. TantQue $t \in \hat{C}_{k-1}$
     (a) $C_t \leftarrow \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$
     (b) TantQue $c \in C_t$
        i. c.count++
     (c) Si $C_t <> \phi$ Alors
        i. $\hat{C}_k \leftarrow + < t.TID, C_t >$
  3. $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$, $k \leftarrow k + 1$
- RETOURNER $\bigcup_k L_k$

Génération de règles - 1

- Approche descendante de génération fondée sur deux propriétés :

Génération de règles - 1

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance simple** : Nous testons les règles ayant le nombre de conditions minimal pour un sous-ensemble fréquent, exemple :

Génération de règles - 1

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance simple** : Nous testons les règles ayant le nombre de conditions minimal pour un sous-ensemble fréquent, exemple :
     R1 $A \Rightarrow B, C$

Génération de règles - 1

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance simple** : Nous testons les règles ayant le nombre de conditions minimal pour un sous-ensemble fréquent, exemple :

     R1 $A \Rightarrow B, C$

     R2 $A, B \Rightarrow C$

Génération de règles - 1

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance simple** : Nous testons les règles ayant le nombre de conditions minimal pour un sous-ensemble fréquent, exemple :
     R1 $A \Rightarrow B, C$
     R2 $A, B \Rightarrow C$
- confiance(R1)=support(ABC)/support(A),

Génération de règles - 1

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance simple** : Nous testons les règles ayant le nombre de conditions minimal pour un sous-ensemble fréquent, exemple :
     R1 $A \Rightarrow B, C$
     R2 $A, B \Rightarrow C$
- confiance(R1)=support(ABC)/support(A),
- confiance(R2)=support(ABC)/support(AB),

Génération de règles - 1

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance simple** : Nous testons les règles ayant le nombre de conditions minimal pour un sous-ensemble fréquent, exemple :
     R1 $A \Rightarrow B, C$
     R2 $A, B \Rightarrow C$
- confiance(R1)=support(ABC)/support(A),
- confiance(R2)=support(ABC)/support(AB),
- confiance(R2)>confinace(R1).

Génération de règles - 2

- Approche descendante de génération fondée sur deux propriétés :

Génération de règles - 2

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance stricte** : Nous commençons par la recherche par les ensembles fréquents les plus grands, exemple :

Génération de règles - 2

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance stricte** : Nous commençons par la recherche par les ensembles fréquents les plus grands, exemple :
     R1 $A \Rightarrow B, C, D$

Génération de règles - 2

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance stricte** : Nous commençons par la recherche par les ensembles fréquents les plus grands, exemple :

     R1 $A \Rightarrow B, C, D$

     R2 $A \Rightarrow B, C$

Génération de règles - 2

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance stricte** : Nous commençons par la recherche par les ensembles fréquents les plus grands, exemple :
     R1 $A \Rightarrow B, C, D$
     R2 $A \Rightarrow B, C$
- confiance(R1)=support(ABCD)/support(A),

Génération de règles - 2

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance stricte** : Nous commençons par la recherche par les ensembles fréquents les plus grands, exemple :
     R1 $A \Rightarrow B, C, D$
     R2 $A \Rightarrow B, C$
- confiance(R1)=support(ABCD)/support(A),
- confiance(R2)=support(ABC)/support(A),

Génération de règles - 2

- Approche descendante de génération fondée sur deux propriétés :
  1. **Redondance stricte** : Nous commençons par la recherche par les ensembles fréquents les plus grands, exemple :
     R1 $A \Rightarrow B, C, D$
     R2 $A \Rightarrow B, C$
- confiance(R1)=support(ABCD)/support(A),
- confiance(R2)=support(ABC)/support(A),
- confiance(R2)>confinace(R1)