Djamila MOKEDDEM

Fouille de Données et Médias Sociaux

Cours et Exercices

Selon le programme de la matière «Fouille de Données et Médias Sociaux» destinée au Master 2 Informatique, Option Systèmes d'Informations et Données

Université Des Sciences et Technologies d'Oran Mohammed Boudiaf, Faculté des Mathématiques et Informatique, Département d'Informatique, 2019-2020.

Préface

Ce polycopié concerne le cours intitulé «Fouille de Données et Médias Sociaux» destiné aux étudiants de Master 2 Informatique, option « Systèmes d'Informations et Données ». Il vise essentiellement à définir le domaine de la fouille de données et à présenter les méthodes les plus utilisées. Il abordera également l'application et l'adaptation de ces approches aux données issues des médias sociaux.

Le document est organisé en trois chapitres, suivi chacun d'une suite de questions et exercices sur le contenu du chapitre. La partie pratique peut être réalisée par le langage Python, R ou tout autre langage adapté à la fouille de données.

Le chapitre 1 est une introduction générale sur la fouille de données. Le domaine du textmining est aussi abordé comme cas particulier. Le chapitre 2 présente les méthodes les plus utilisées dans les différentes tâches du datamining : règles d'association, classification et segmentation.

Le chapitre 3 aborde les aspects principaux pouvant aider à analyser ces données. On présente essentiellement certaines propriétés statistiques ainsi que les mesures permettant de comprendre la structure du réseau. L'assimilation de certaines parties du chapitre nécessite une connaissance préalable des notions de base en théorie des graphes. Plusieurs techniques du datamining peuvent être utilisées et adaptées à la fouille des médias sociaux comme les règles d'associations, les arbres de décision et la segmentation (clustering). On présente aussi dans ce chapitre trois tâches importantes dans le domaine du «social media mining» : la détection des communautés, la classification collective, et les systèmes de recommandation.

Chapitre 1

Introduction au Datamining

1 Introduction

Le datamining ou « fouille de données » regroupe un ensemble de techniques permettant l'extraction, à partir d'un important volume de données brutes, des connaissances originales auparavant inconnues.

Ce chapitre est un aperçu sur la fouille de données où on prend comme exemple le cas de données textuelles ou « textmining ». On présente aussi des exemples d'application de certaines techniques du datamining sur les données issues des médias sociaux.

2 Le processus de découverte de connaissances

L'extraction de connaissances à partir des bases de données ou KDD (*Knowledge Discovery in Databases*) fait référence à un processus multidisciplinaire qui englobe, entre autres, les bases de données, l'apprentissage automatique, la reconnaissance des formes, les statistiques, et la visualisation des données. L'étape de fouille de données ou datamining intervient comme une étape principale dans ce processus (figure 1.1).

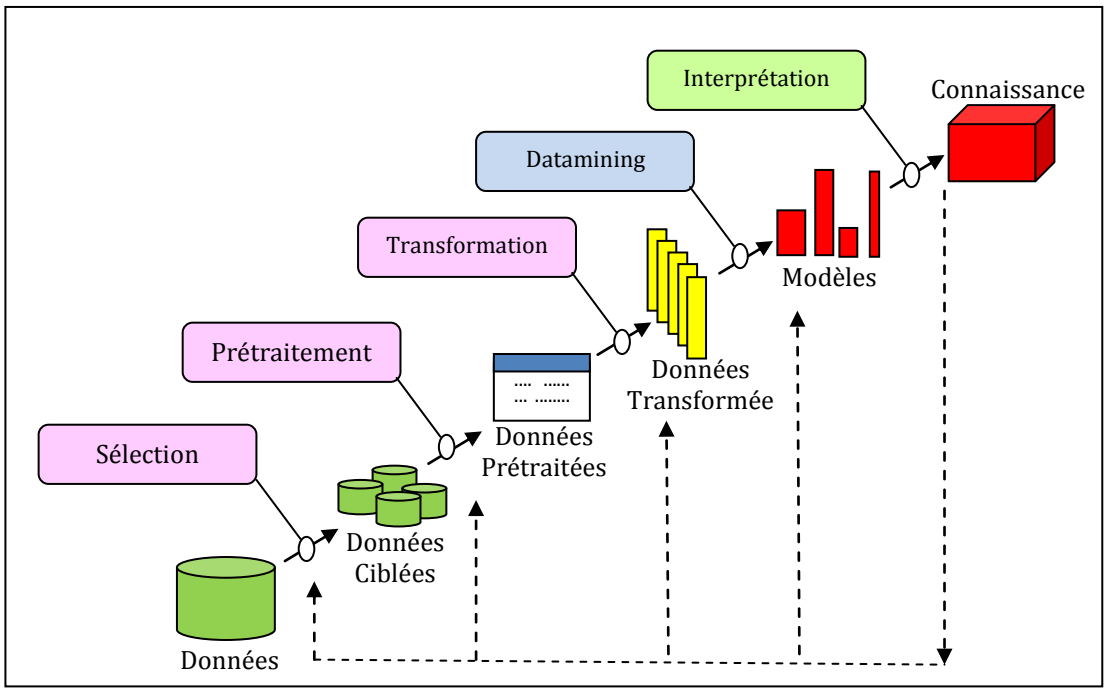


Figure 1.1: Le processus d'extraction de connaissances

2.1 Définition du datamining

Il s'agit d'une "exploration ou fouille" visant à découvrir "de l'information cachée" que les données renferment et que l'on découvre à la recherche d'associations, de règles de classification et d'autres types de connaissances qui serviront à l'aide à la décision. Le datamining peut accomplir différentes tâches avec différents outils. C'est en résumé un processus itératif et interactif qui permet de découvrir de nouvelles connaissances, valides, utiles et compréhensibles, à partir de grandes masses de données.

Les données sont d'abord préparées (discrétisation, traitement des valeurs manquantes, ...etc.). Le datamining intervient donc au niveau des données transformées et permet de produire des modèles (arbres de décision, clusters, réseau de neurone, .. etc.). Le schéma classique de ce processus est illustré dans la figure (Figure 1.1).

2.2 Prétraitement et transformation des données

Les données dans leur état brut peuvent contenir plusieurs anomalies ou avoir un format ou volume qui ne sont pas adéquats à l'objectif visé. Une phase de prétraitement et transformation est généralement nécessaire afin de préparer les données à la tâche du datamining. On cite comme exemples :

- **Structurer les données**

Le format usuel de données traitées par le datamining est celui «Attributs-valeurs», on parle alors de « données structurées ». Beaucoup de données nécessitent une phase de «structuration» afin de les préparer aux algorithmes du datamining, on cite comme exemple les données textuelles (Comment transformer la collection de textes en tableau de valeurs, sans perte d'informations ?!).

- **Traitement des valeurs manquantes**

Les décisions dans de tels cas peut être par exemple : Supprimer les instances concernées ou bien remplacer la valeur manquante par celle la plus commune, ou bien tout simplement les ignorer.

- **Discrétisation**

Dans certains cas, il n'est pas nécessaire de considérer les valeurs réelles exactes, comme par exemple « les mesures de températures » mais seulement des intervalles (faible, normal, élevé).

- **Sélection d'attributs pertinents**

Les attributs ne sont pas nécessairement tous pertinents dans la tâche du datamining (prédire si un client achètera un produit donné n'est pas lié à l'attribut nom) .

- **Extraction d'attributs nouveaux**

De nouveaux attributs sont extraits comme par exemple « Superficie » à partir de « longueur » et « largeur ».

- **Echantillonnage**

Beaucoup d'algorithmes en datamining utilisent seulement un échantillon de données ciblées. Le choix de cet échantillon est fondé sur des méthodes statistiques. Il peut être par exemple : Aléatoire, avec ou sans remplacement (avec remplacement: une instance peut être sélectionnée plusieurs fois, sans remplacement: l'instance est supprimée du groupe une fois sélectionnée).

2.3 Tâches et outils du datamining

Le datamining peut accomplir différentes tâches avec différentes techniques. On cite dans ce qui suit les tâches les plus utilisées.

- **La classification**

Elle permet de prédire si une instance de donnée est membre d'un groupe ou d'une classe prédéfinie selon la valeur de sa caractéristique. Elle peut se faire par plusieurs techniques comme : les arbres de décision, les réseaux de neurones, etc.. La régression permet la prédiction de valeurs de type réelle.

- **La Segmentation (clustering)**

Elle permet le partitionnement d'un ensemble d'entités en un ensemble de sous-groupe ou « clusters » plus homogènes. Parmi les outils de segmentation, on cite l'algorithme K-means.

- **Extraction des règles d'association**

Le but du datamining peut être la description de ce qui se passe au sein d'une base de données complexe de manière à augmenter la compréhension de celle-ci. Les règles d'association sont une technique descriptive : Elles permettent de détecter l'ensemble d'attributs qui ont une occurrence fréquente ensemble, et les règles qui les rassemblent (exp. 90% des clients qui achètent le livre X, achètent aussi le livre Y). Comme exemples d'algorithmes on cite Apriori et Fp-Growth.

3 Le datamining pour les médias sociaux (Media Social Mining)

Les techniques du datamining peuvent être appliquées sur les médias sociaux afin d'extraire une connaissance utile. Les approches les plus répandues seront décrites dans le chapitre 2.

3.1 Collecte des données à partir des médias sociaux

Un ensemble de données sont offerts par certains organismes afin de tester et valider les méthodes proposées dans le media social mining. On cite comme exemples :

- http://socialcomputing.asu.edu
- http://snap.Stanford.edu
- https://github.com/caesar0301/awesome-public-datasets
- https://aminer.org/data

Certains médias sociaux offrent aussi la possibilité de collecter un sous ensemble restreint de leurs données via les APIs. On cite comme exemple :

- https://dev.twitter.com/docs/api
- https://www.flickr.com/services/api/
- https://developers.facebook.com/

3.2 Principales applications du datamining dans les médias sociaux

Les tâches du datamining peuvent être adaptées à la nature des données issues des médias sociaux, on cite comme exemple :

- **La Classification**
  - Prédire si une personne est influente ou non dans twitter
  - Classification des tweets en plusieurs classes prédéfinies: évènement, Opinion, message privé, ….
  - Classification des « Trends Topics TT » en plusieurs catégories (art, politique, sport,….) (TT: Les dix sujets tendances du moment sur Twitter (affichés à gauche))
  - Analyse des sentiments: Déterminer si un texte a été écrit pour exprimer une opinion positive/ négative/ neutre

- **Classification collective**

  Les méthodes de classification traditionnelle ignorent les corrélations entre les instances (attributs locaux) La classe (label) d'un nœud n'est généralement pas indépendante des caractéristiques de ses voisins. La classification collective prend en considération ces corrélations.

- **Détection de communautés**

  Une communauté est ensemble de nœuds plus fortement liés entre eux qu'avec les autres. Connaissant les communautés d'utilisateurs, il est par exemple possible de leur recommander des éléments d'information susceptibles de les intéresser (Marketing).

- **Systèmes de Recommandation**

  L'idée est de présenter à un client une liste personnalisée de suggestions, selon son profil, ses préoccupations et ses attentes

4 Datamining sur les données textuelles «Text Mining»

Le texte est un type de données non structurées très présent dans la majorité des applications. Certaines études estiment que seulement 20% du contenu du web est structuré. Le textmining consiste à extraire une connaissance à partir des données textuelles. Les textes présents dans les médias sociaux peuvent cacher une connaissance qui pourrait être utile dans la prise de décision, comme par exemple l'analyse des sentiments et des opinions.

4.1 Exemple de tâches du textmining

La majorité des outils du datamining peuvent être appliqués sur le texte afin d'effectuer différentes tâches, on cite comme exemple

- Classification de texte : prédiction automatique du sujet d'un texte (exp. détection de spams, ..)
- Partitionnement de documents en groupes homogènes
- Résumé automatique
- Identification des tendances : Analyser l'évolution de la popularité des thèmes dans les réseaux sociaux, identifier les sujets émergents (topic detection).
- Analyse des relations entre les termes (co-ocurrence)
- Nuage des mots (Word clouds)
- Analyse des sentiments

4.2 Prétraitement du texte

Une phase de traitement préalable est nécessaire afin de transformer le corpus textuel à «fouiller» en une forme plus structurée. Le texte est traité dans la plupart des méthodes du textmining en tant que «sac de mots» représenté généralement par un vecteur, où chaque élément désigne le mot et son poids correspondant. Ce dernier est fondé essentiellement sur le nombre d'occurrence du mot dans le texte. La phase de prétraitement nécessite une suite d'opérations standards, dont les détails dépendent de la langue traitée.

4.2.1 Extraction des termes en Sac de mots (Uni-grammes)

On extrait les termes appartenant au texte. Cette approche est la plus simple, mais l'ordre des mots dans le texte n'est pas pris en considération.

4.2.2 Extraction des termes en Sac de N-grammes

Un *N*-gramme est une séquence de *N* mots consécutifs. Contrairement au modèle de sac de mots, les N-grammes permettent de capturer l'ordre des mots dans le texte.

Exemple : *« Fouille de Données et Média sociaux »*

Uni-grammes	*Fouille*	*de*	*Données*	*et*	*Média*	*Sociaux*

Bi-grammes	*Fouille de*	*de Données*	*Données et*	*et Média*	*Média Sociaux*

Une autre approche consiste à extraire les séquences des N-caractères consécutifs (Sac de caractères N-grammes).

Exemple :

« USTO-MB »

tri-grammes caractères	UST	*STO*	*TO-*	*O-M*	*-MB*

La technique des N-grammes présente plusieurs avantages :

- Elle est tolérante aux fautes d'orthographes, par exemple les mots « Données » et «Donées» partagent la majorité des N-grammes caractères.
- Elle opère indépendamment des langues.
- De nombreuses études ont montré que le traitement linguistique (cf. 3.2.3) n'améliore pas la performance dans les systèmes n-gramme

4.2.3 Traitement linguistique

Ce prétraitement dépend de la nature du langage traité. On cite comme exemple:

Filtrage : On supprime les mots qui n'apportent pas beaucoup d'informations, comme les articles, les conjonctions, les prépositions.

Stemming : On supprime le suffixe d'un mot jusqu' à obtenir une racine. Cette racine est appelée stem. Par exemple les mots continu, continua, continuait, continuant sont tous ramenés au stem continu.

Afin d'améliorer l'information disponible sur les mots, une analyse du langage naturelle peut être appliquée dans certaines applications de fouille de texte, comme par exemple :

- Spécifier la nature linguistique de chaque mot (verbe, nom, adjectif,...)
- Enlever l'ambigüité des mots en stockant les synonymes du terme.

4.3 Représentation des textes

Le modèle vectoriel consiste à associer à chaque terme du texte une valeur fondée sur le nombre d'occurrence. Le corpus (collection de textes) est généralement représenté sous forme de matrice $W(n, m)$, avec n : le nombre de termes, m: le nombre d'entités textuelles (documents, tweets, messages,...). $W[ti, dj]$ désigne le poids du terme ti dans l'entité textuelle dj. Plusieurs codage sont utilisés pour mesurer le poids, on présente les plus répandus : TF et IDF

4.3.1 Codage par occurrence des mots (TF : Term Frequency)

Le poids est mesuré par le nombre de fréquence du terme dans le document :

$$TF(t_i, d_j) = \frac{Nombre\ d'occurence\ du\ terme\ t_i\ dans\ le\ document d_j}{Nombre\ total\ de\ termes\ dans\ d_j}$$

4.3.2 Codage TFIDF (Term Frequency Inverse Document Frequency)

TFIDF évalue l'importance d'un mot pour un document relativement à une collection de documents (corpus). Pour un document d_j, un terme qui est présent dans presque tous les autres documents du corpus n'a pas un pouvoir discriminant pour qu'il soit considéré comme «pertinent» pour d_j. Si c'est le cas, son poids doit être faible.

La mesure TFIDF tente d'atténuer l'effet des termes qui surviennent trop souvent dans la collection. Pour cela, on réduit le poids des mots qui sont très fréquents dans la collection grâce à la valeur IDF qui se calcule comme suit :

$$IDF(t_i) = \log\left(\frac{Nombre\ total\ de\ documents}{Nombre\ de\ documents\ qui\ contiennent\ le\ terme\ t_i}\right)$$

La valeur IDF d'un terme rare est élevée, tandis que l'IDF d'un mot plus fréquent dans le corpus est faible. Plus le terme est rare, plus la valeur IDF est élevé. La fonction logarithme est utilisée afin d'amortir la valeur IDF associée à un terme. Le poids $TFIDF$ est le produit de TF et IDF:

$$TFIDF(t_i, d_j) = TF(t_i, d_j) \times IDF(t_i)$$

Exercices

1) Expliquer les différences principales entre le datamining et les requêtes classiques appliquées sur les bases de données relationnelles (SQL par exemple).
2) Proposer et décrire brièvement une solution issue du «datamining» aux situations suivantes :
   a) Les responsables d'un supermarché veulent présenter certains produits Xi $(i = 1\,..\,n)$ en « promotion » afin de développer leurs ventes. Ils veulent aussi éviter de faire la promotion sur un groupe de produits qui sont souvent achetés ensemble.
   b) L'objectif est de minimiser le nombre de clients qui changent d'opérateurs dans une entreprise de téléphonie. On dispose d'un historique sur un ensemble de clients (Frais mensuels, Nombre d'appels échoués, nombre de plaintes, ...)
   c) On voudrait personnaliser la stratégie de marketing en fonction de certaines caractéristiques de la clientèle. Au lieu d'envoyer les mêmes messages à tous les clients, on voudrait cibler chaque type de clientèle par une stratégie différente parmi cinq (5) étudiées.
   d) On dispose d'une base d'images. L'objectif est de trouver des images semblables à une image requête.
3) On voudrait «fouiller» des données issues du média social Facebook. L'objectif est de découvrir la classe de la personne «infulente ou non». On

suppose disposer de plusieurs attributs de chaque acteur dans le réseau (nombre d'amis, nombre de messages postés, nombre de «jaime», ...)

a) Proposer une représentation de ces données.
b) Quelle est la tâche du datamining adéquate (parmi celles étudiées) pour extraire ce type de connaissance ?
c) Comment peut-on avoir des données pour la phase de validation ?
d) Comment peut-on valider les résultats trouvés ?

4) Soit un corpus constitué de deux documents (voir tableau)

a) Donner la matrice des mots avec les poids TF
b) Donner la matrice des mots avec les poids TFIDF
c) Interpréter les résultats TFIDF pour les termes «il» et «dent»

	Terme	Fréquence
Document 1	*il*	1
	a	1
	le	2
	don	1
Document 2	*il*	1
	a	1
	la	2
	dent	3

5) Soient les corpus C1 et C2, on voudrait appliquer la tâche de clustering sur chacun des deux corpus.

a) Montrer sur l'exemple C1 l'intérêt d'utiliser le « sac des n-grammes » par rapport au « sac de mots ».

C1

- *Les Bases de Données*
- *Les Données de Base*

C2

- *Informatisation de la mairie*
- *Informatiser la mairie*
- *Introduire l'informatique à la mairie*

b) Montrer sur l'exemple C2 l'intérêt d'utiliser les « n-grammes caractère » par rapport au n-grammes mots ». Quelle sont les avantages et inconvénients des n-grammes caractères?

Comparer entre les poids des mots qui sont courants dans les documents comme « le, la, un, pour, ... » dans le cas d'utilisation des mesures TF ou TFIDF.

Chapitre 2

Les Méthodes du Datamining

1 Introduction

Le datamining ou «fouille de données» couvre l'ensemble des outils et méthodes qui permettent d'extraire des informations «significatives» depuis de grandes quantités de données. C'est un domaine interdisciplinaire qui utilise des techniques d'apprentissage automatique, de la reconnaissance des formes, des statistiques, des bases de données et de la visualisation. Ce chapitre présente les méthodes les plus utilisées dans les différentes tâches du datamining : règles d'association, classification et segmentation.

2 Extraction des itemsets fréquents

La technique d'extraction des itemsets fréquents ou règles d'association est un des outils les plus répandus en datamining. Elle permet la découverte de règles intelligibles et exploitables dans un ensemble de données volumineux, règles exprimant des corrélations (associations) entre items ou attributs dans une base de donnée transactionnelle ou relationnelle. C'est un sujet de recherche attractif et très actif vu son large champ d'application à divers domaines tels que : le marketing, aide au diagnostic médical, télécommunication, analyse de données spatiales, .. etc. Dans la suite, nous présentons deux algorithmes de référence dans ce domaine: Apriori et FP-Growth.

2.1 Terminologie et Notations

De nombreuses notations et définitions sont nécessaires pour décrire l'approche d'extraction des motifs fréquents :

- **Base de données transactionnelle** : un ensemble $D = \{t1, t2, \ldots \ldots tn\}$ tel que $ti \quad i = 1 \ldots n$ est une transaction.
- **Item (motif) :** Correspond à une valeur d'un attribut dans la base D, Soit $I = \{i1, i2, \ldots . im\}$ un ensemble fini d'items.
- **Itemset :** un itemset X est un ensemble d'items tel que $X \subset I$.
- **K-itemset :** est un itemset de taille k ou itemset avec k items.
- **Support d'un itemset :** Le support d'un itemset X noté $Supp(X)$ est la proportion de transactions de D contenant X : $Supp(X) = Freq(X) / \left| D \right|$ où $Freq(X)$ est le nombre des transactions dans D qui contiennent l'itemset X.

- **Itemset fréquent :** On appelle X un itemset fréquent si $Supp(X)$ est supérieur ou égale à un seuil minimal $Minsupp$.

- **Base conditionnelle :** Une base conditionnelle d'un itemset X est l'ensemble des transactions de la base de données contenant X. De manière formelle, une transaction d'identificateur TID, $t = (TID, Y)$ appartient à la base conditionnelle de X si et seulement si $X \subseteq Y$.

- **Règle d'association :** Implication de la forme $A \rightarrow B, où\ A, B \subset I, A \cap B = \emptyset$ Avec A et B deux itemsets. A est dit prémisse (ou cause) et B est dit conclusion (ou conséquence).

- **Support d'une règle :** Le support d'une règle $R: A \rightarrow B$ noté $Supp(R)$ est la proportion de transaction de D contenant $A \cup B$, il est calculé par $Supp(R) = Freq(A \cup B) / |D|$ où $Freq(A \cup B)$ est le nombre des transactions dans D qui contiennent $A \cup B$.

- **Confiance d'une règle :** La confiance d'une règle $R: A \rightarrow B$ est la proportion de transactions de D contenant l'ensemble A qui contiennent aussi B : $conf(A \rightarrow B) = Freq(A \cup B)/Freq(A)$ où $Freq(A \cup B)$ est le nombre de transactions dans D contenant $A \cup B$. On voit immédiatement que la confiance se définit aussi par un rapport de support : $conf(A \rightarrow B) = sup(A \cup B)/sup(A)$.

- **Règle d'association fréquente :** On appelle R une règle d'association fréquente si $conf(R)$ est supérieur ou égale à un seuil minimal $MinConf$.

Exemple

Soit une base de données transactionnelle constituée de quatre transactions avec leurs identificateurs TID:

TID	Transaction
2000	A, B, C
1000	A, C
4000	A, D
5000	B, E, F

- « B, E, F » est une transaction d'identificateur TID = 5000
- « A» « B» « C» « D» « E» « F» sont des items
- « A, C» est un 2-itemset
- Support (« A, C ») = 2
- Support de la règle d'association (A→C) = 2/4 = 50%
- Confiance de la règle d'association (A→C) = 2/3 = 66,6%

2.2 Représentation compacte des itemsets fréquents

Puisque le nombre d'itemsets fréquent peut être très grand, il est important dans certains cas d'extraire un sous ensemble représentatif (figure 2.1):

- **Itemset fréquent maximal**

Un itemset fréquent est dit maximal si aucun de ses sur-ensemble immédiat n'est fréquent. Etant donné un itemset fréquent X, si en ajoutant n'importe quel autre item Y à X, l'itemset XY devient non fréquent, alors X est dit maximal). Formellement :

X maximale $\Leftrightarrow \exists$ Y: X $\subset$ Y et supp(Y) $\geq$ minSupp

Les itemsets fréquents maximaux réduisent la taille des résultats mais ne fournissent pas le support des sous-ensembles, support qui est nécessaire pour la génération des règles associatives. C'est le cas des algorithmes Maxminer et MAFIA.

- **Itemset fermé fréquent**

Un itemset fréquent est dit fermé si aucun de ses sur-ensemble immédiats n'a un support identique. Formellement:

X fermé $\Leftrightarrow \exists$ Y: X $\subset$ Y et supp(Y) = supp()X

Les itemsets fermés permettent de réduire le nombre des itemsets générés avec la possibilité de reconstruire l'ensemble de tous les itemsets fréquents ainsi que leurs fréquences. On cite comme exemple l'algorithme FP-Close.

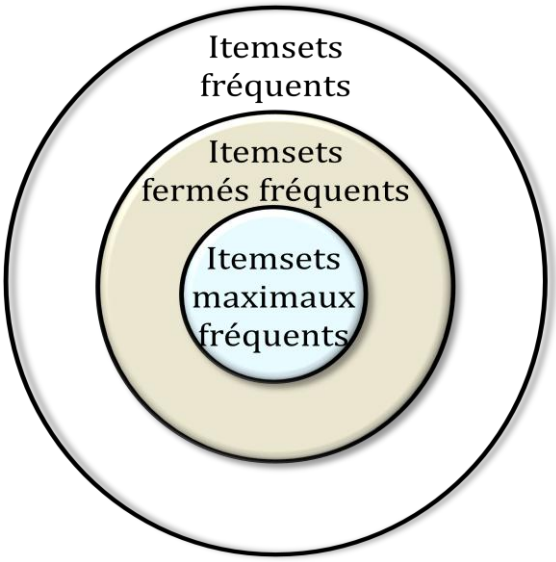


Figure 2.1 Représentation compacte des motifs frequents

2.3 Algorithme Apriori

Agrawal et Sharfer ont proposé le premier algorithme d'extraction des règles d'association dans les bases de données transactionnelles: l'algorithme Apriori. Cet algorithme fonctionne en deux phases :

1) Détermination des itemsets fréquents ($support >= minSupp$)
2) Dérivation des règles d'association valides ($confiance >= minConf$)

2.3.1 Déterminer les itemsets fréquents

La première phase repose sur une approche dite «Générer et Tester», c.-à-d. on génère les itemsets candidats puis on teste s'ils sont fréquents.

Algorithme Apriori
Entrée : D une base de données transactionnelle ou relationnelle, minSup
Sortie: Les itemsets fréquents

Générer les 1-itemsets fréquents ;
Répéter
Générer les (k+1)-itemsets candidats à partir des (k)-itemsets fréquents ;
Elaguer les candidats contenant des sous ensembles non fréquents de taille k ;
Calculer le support de chaque candidat en lisant la base ;
Eliminer les candidats non fréquents ;
jusqu'à n'avoir aucun nouveau itemset fréquent

Algorithme 2.1: Algorithme de principe Apriori

Apriori souffre d'une complexité exponentielle due aux parcours répétés de la base de données, lors de la première phase. Elle est de l'ordre de $O(n.m.2^m)$ avec $m = |I|, n = |T|$. A titre d'exemple, Pour découvrir un motif fréquent de taille 100, on génère presque 2^{100} candidats. En plus, des scans répétitifs de la base sont nécessaires pour compter les supports et décider si le motif candidat est fréquent ou non.

Afin de réduire le nombre de candidats, Apriori utilise une technique d'élagage basée sur la propriété de non-monotonie du support. Cette propriété s'exprime comme suit :

- Tous les sous ensembles d'un itemset fréquent sont fréquents
- Tous les sur-ensembles d'un itemset non fréquent sont non fréquents

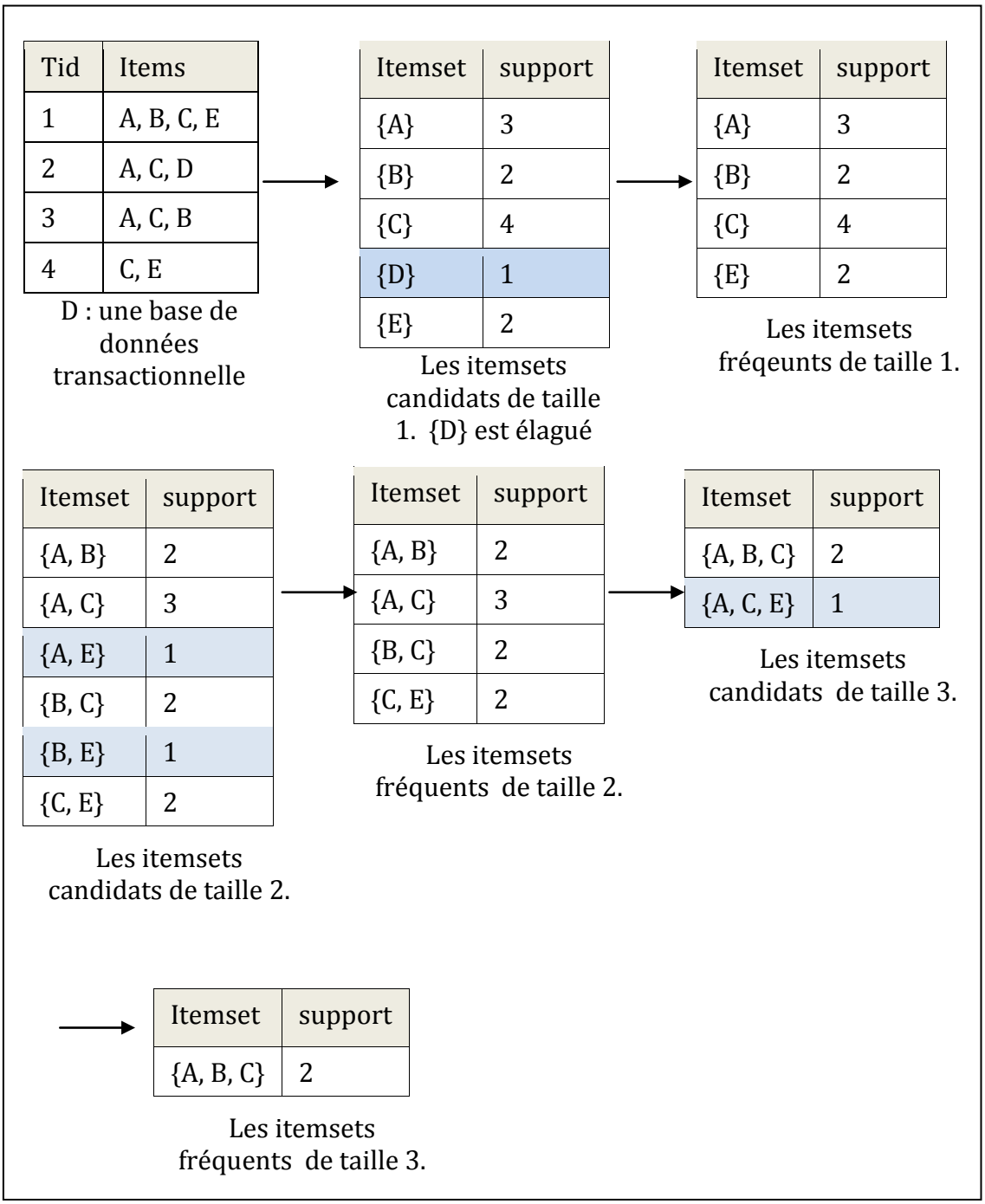


Figure 2.1 Exemple d'application de l'algorithme Apriori (minsup = 0,5)

2.3.2 Extraction des règles d'association

Les règles d'association sont générées à partir des itemsets fréquents selon le principe illustré dans l'algorithme 2.2. Un exemple de dérivation de règles d'association est présenté dans la figure (Figure 2.2).

Algorithme Extraxtion des règles d'association
Entrée : Les itemset fréquents, minSup, minConf
Sortie: Les règles d'association

Pour Chaque itemset fréquent *f*,
    générer tous les sous ensembles non vides de *f*
Fin
Pour Chaque sous ensemble non vide *s* de *f,*
    Si (support(f) / support(s) >= minConf)
    Alors générer la règle ***«s → (f-s)»***

Algorithme 2.2: Extraction des règles d'association

1) **Générer les sous ensembles non vides**
- {A}, {B}, {C}, {A, B}, {A, C}, {B, C}

2) **Générer les règles**
- A → B, C
- B → A, C
- C → A, B
- A, B→ C
- A, C→ B
- B, C→ A

Figure 2.2: Exemple de dérivation des règles d'association à partir de l' itemset fréquent A, B, C

2.4 Algorithme FP-Growth

Afin d'éviter les parcours répétés de la base de données, Han et al ont proposé une méthode différente des approches par niveaux permettant d'extraire des itemsets fréquent sans génération de candidats. Cette méthode s'appelle Fp-Growth (Frequent Pattern Growth).

2.4.1 Construction de la structure FP-tree

Fp-Growth consiste d'abord à compresser la base de données en une structure compacte appelée FP-tree (frequent pattern tree), puis à diviser la base de données ainsi compressée en sous projection de la base de données appelées bases conditionnelles. Chacune de ces projections est associée à un item fréquent. L'extraction des itemsets fréquents se fera sur chacune des projections séparément. Ce processus est réalisé donc en deux étapes :

1) Représenter la base de données dans une structure compacte «FP-tree» après deux scans seulement.
2) Extraire les itemsets fréquents en accédant à l'arbre FP-tree.

FP-tree est une structure arborescente constituée d'un arbre et d'un index ou table de liens (Figure 2. 3):

- **Arbre** : mise à part la racine null, chaque nœud de l'arbre contient trois informations : l'item que représente ce nœud «*itemName*» , sa fréquence «*itemCount*», ainsi que le nœud suivant dans l'arbre.

- **Index** : contient la liste des items fréquents. A chaque item est associé un pointeur indiquant le premier nœud de l'arbre contenant cet item.

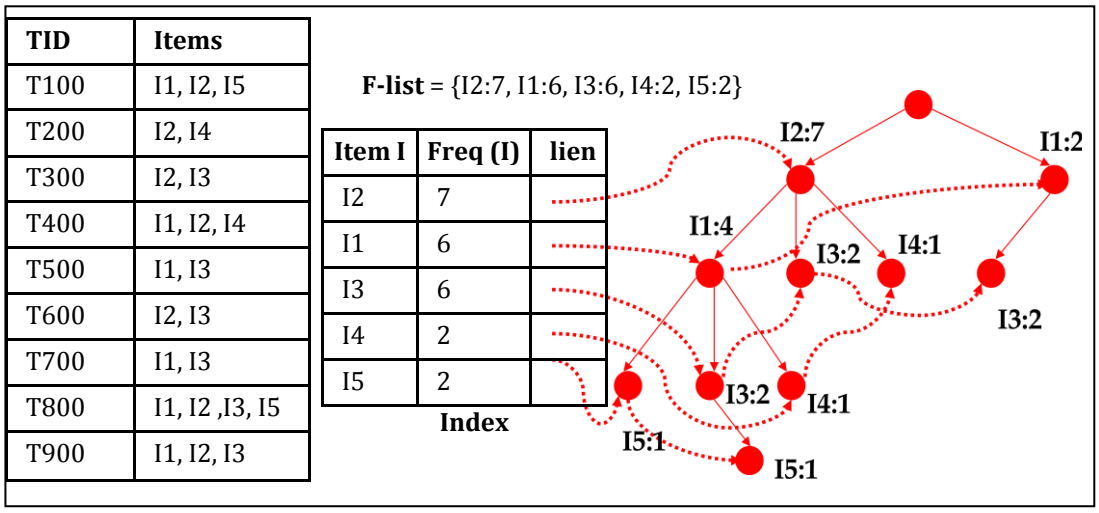


Figure 2.3: Exemple d'une structure FP-tree

La construction de FP-tree nécessite deux parcours de *D* (Algorithme 2.3). Le premier parcours permet de générer les items fréquents de taille 1 qui seront triés par ordre décroissant des supports dans la structure F-list (Figure 2.2). Un second parcours de *D* est ensuite effectué, où chaque transaction est triée selon l'ordre des items dans F-list, i.e. par support décroissant des items.

Le nœud racine de l'arbre null est d'abord créé. Durant ce même parcours, une branche sera créée pour chaque transaction, mais des transactions ayant un même préfixe partageront le même début d'une branche de l'arbre. Ainsi, deux transactions identiques seront représentées par une seule et même branche. La raison pour laquelle les items sont traités du plus fréquent au moins fréquent est que les items fréquents seront proches de la racine et seront mieux partagés par les transactions. Ceci fait du FP-tree une bonne structure compacte pour représenter les bases transactionnelles. Néanmoins, cela ne garantit pas que tout le FP-tree tienne en mémoire centrale.

Algorithme ConstruireFP-tree
Entrée : *D* une base de données transactionnelle ou relationnelle, minSup
Sortie: FP-tree,

1) Scanner la BD pour extraire les 1-itemsets fréquents "*F-list*", et trier F-list par ordre décroissant des supports.
2) Créer la table des liens et le nœud racine *R* et lui affecter «null»
3) **Pour** chaque transaction *t*
4) Sélectionner les items fréquents selon l'ordre de *F-list*. Soit $t = [p|P]$, avec *p* le premier élément et *P* le reste des éléments dans t
5) *insert_tree([p|P, R)*

6) **fonction** *insert_tree([p|P, R)*
   Si (*R* possède un noeud fils *q*, tel que *q.itemName = p.itemName)*
   q.itemCount++
   Sinon
   Créer un nouveau noeud *q*;
   q.itemCount =1;
   lier *q* avec le noeud père *R ;*
   lier *q* avec la liste de même itemName par le lien horizontal ;
   Si *P* est non vide, appeller récursivement *insert_tree(P, R)*

Algorithme 2.3: Construction de la structure FP-tree

2.4.2 Extraction des itemsets fréquents

Après la compression de la base sous forme d'un arbre FP-tree, il est possible d'extraire tous les itemsets fréquents à partir de cette structure. Pour accomplir cette tâche, Han présente un ensemble de lemmes et propriétés, ainsi qu'une description de l'algorithme FP-Growth .

Un des principes de base est la notion de «base conditionnelle » : Si X et Y sont deux itemsets, le support de $X \cup Y$ dans la base de données est exactement le support de Y si on restreint la base aux transactions contenant X. Cette restriction est appelée *la base conditionnelle de X*. L'arbre construit est appelé *FP-tree conditionnel* de X. On note que pour tout itemset Y fréquent dans la base conditionnelle de X, $X \cup Y$ est un itemset fréquent dans la base d'origine

Nous illustrons ce processus par l'exemple de la figure (Figure 2.4) où on calcule les itemsets fréquents associées à l'item I3. Les bases conditionnelles de I3 représentent les chemins qui précèdent I3 dans l'arbre comme par exemple I2I1 qui apparait avec un support 2. L'arbre FP-tree conditionnelle est une structure FP-tree construite à partir de la base conditionnelle.

Lorsque FP-tree contient un seul chemin, on arrête la récursivité et on déduit les itemsets fréquents en calculant toutes les combinaisons possibles des items dans le chemin avec l'item en question. Les étapes principales de F-Growth sont données dans (Algorithme 2.4).

Algorithme FP-Growth
Entrée: Fp-tree, noeud (l'itemset considéré), minSupp
Sortie: itemsets fréquents

Si l'arbre contient un seul chemin P **Alors**
  Pour chaque combinaison β des noeuds dans P
   Générer l'itemset $(\beta\ U\ noeud)$ avec support = support min des noeuds dans β
Sinon
  Pour chaque item ai dans la table index **faire**
   Générer $\beta = ai \cup nœud$ avec $support = support(ai)$
   Construire la base conditionnelle de β
   Construire le FP-tree conditionnelle de β, β_tree
   si B_tree est non vide **alors** FP-Growth(β _tree, β);
 finPour

Algorithme 2.4: Algorithme FP-Growth

Item I3:6
- Base Conditionnelle {(I2 I1: 2), (I2: 2), (I1: 2)}
- F-list I2:4 , I1:4
- FP-tree Conditionnelle
- **Itemsets Fréquents**: **I3:6**

Item I1I3:4
- Base Conditionnelle {(I2:2)}
- F-list I2:2
- FP-tree Conditionnelle
- **Itemsets Fréquents: I1I3:4, I2 I1 I3: 2**

Item I2I3:4
- Base Conditionnelle ϕ
- **Itemsets Fréquents**: **I2I3: 4**

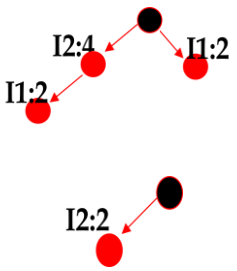


Figure 2.4: Exemple illustratif sur l'extraction des itemsets fréquents contenant l'item I3

3 La tâche de classification dans le datamining

La classification ou « Apprentissage Supervisé » est une des tâches importante du datamining. L'objectif est de construire un modèle qui permet de prédire si une instance de donnée est membre d'une classe prédéfinie. Nous décrivons dans ce qui suit les notions de base partagées par la plupart des algorithmes utilisées dans la classification comme les arbres de décision, les méthodes ensemblistes, ainsi que les méthodes associatives.

3.1 Principe général de la classification

La classification utilise un ensemble D de données appelées ensemble d'apprentissage. Chaque donnée est typiquement représentée sous forme d'un

vecteur d'attributs x $= < x1, x2, \ldots., xm, y >$ avec y un attribut de classe. L'objectif de la classification est d'entraîner un algorithme de classification A sur l'ensemble S, pour trouver une bonne approximation d'une certaine fonction f(x) = y. La fonction approximative Cl calculée est appelée classifieur. L'évaluation de la précision de Cl est faite sur un ensemble de données T indépendant de S, appelé ensemble de test. Le classifieur sera par la suite capable de prédire la valeur de classe y pour de nouvelles données d, en calculant Cl(d) (Figure 2.5).

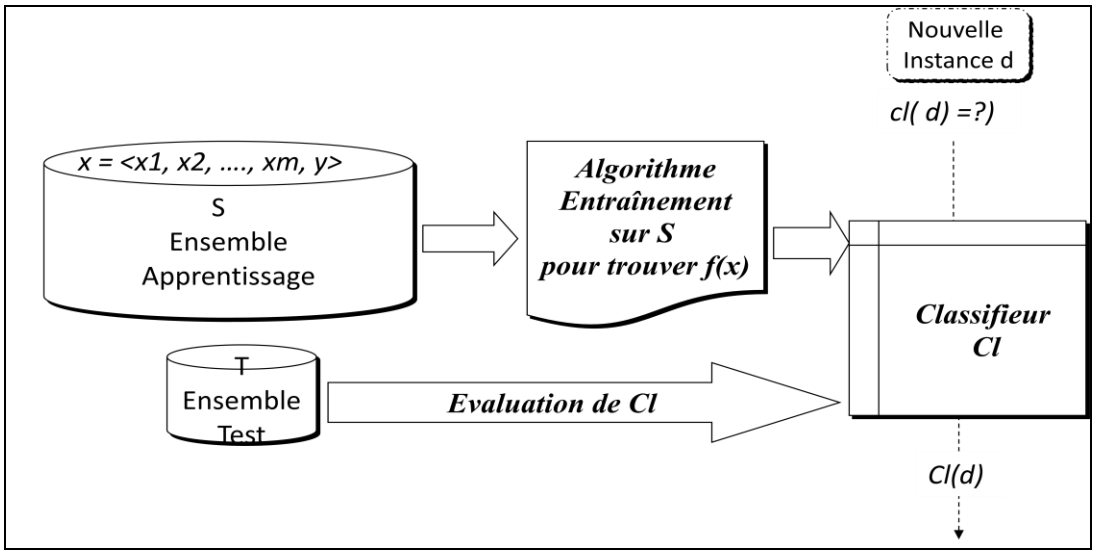


Figure 2.5: Schéma général de la tâche de classification

3.2 Evaluation de la performance de la classification

La performance de la classification est mesurée en fonction du nombre d'instances bien classées et mal classées par le modèle. Elle est évaluée généralement par « la précision » et « le taux d'erreur » :

- $Précision\ (Accuracy) = \frac{Nombre\ d'instances bien\ classées}{Nombre\ total\ d'instances}$
- $Taux\ d'erreur\ (Error\ rate) = \frac{\text{Nombre d'instances mal classées}}{Nombre\ total\ d'instances}$

Ces valeurs sont mesurées sur l'ensemble de test qui peut être construit par différentes méthodes :

- **La méthode 2/3 Apprentissage, 1/3 Test**

L'apprentissage est réalisé sur 2/3 de l'ensemble de données, tandis que le 1/3 est réservé pour le test.

- **La méthode de validation croisée**

C'est une méthode qui consiste à diviser l'ensemble D en k parties disjointes $\{D1, D2, \dots Dk\}$. Le processus suivant est répété k fois (Figure 2.6): construction d'un classifieur Cli avec l'ensemble D privé de Di $(D - Di)$, ensuite évaluer la précision de Cli en utilisant les données Di. La précision globale est obtenue par la moyenne. Les ensembles construits de cette manière sont appelés « cross validated commitees ».

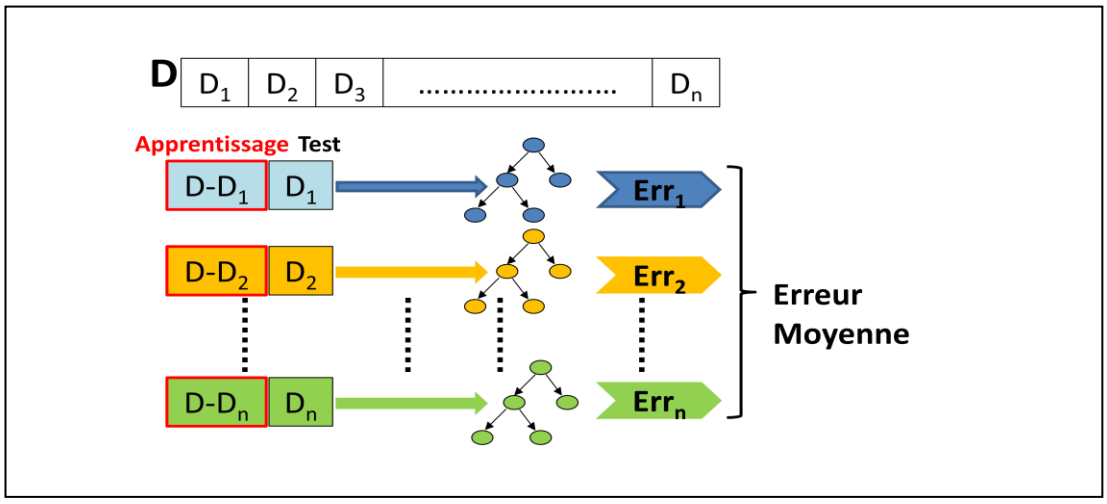


Figure 2.6: La méthode de validation croisée

3.3 Les arbres de décision

Le principe d'un arbre de décision se base sur une division récursive et le plus efficacement possible des instances de l'ensemble d'apprentissage, déjà classés, par des tests définis à l'aide des attributs, jusqu'à ce que l'on obtienne des sous ensembles d'instances ne contenant -presque- que des instances appartenant tous à une même classe.

3.3.1 La phase de construction de l'arbre

Etant donné un ensemble d'apprentissage D constitué de n tuples, et m attributs, C étant un attribut qualitatif (variable de classe) ; l'objectif est de construire un arbre de décision permettant de faire la classification des données selon les valeurs de la classe C. Chaque noeud créé représente une condition appliquée à un attribut sélectionné. Le choix de l'attribut est basé sur la mesure du « Gain » d'informations.

3.3.2 Mesures de sélection d'un attribut

La quantité d'information nécessaire pour déterminer si une instance prise au hasard fait partie d'une des classes peut être calculée par différentes mesures utilisées dans la communauté du datamining comme le Gini Index dans l'algorithme ID3 et le gainRation dans C4.5. On teste successivement tous les attributs pour trouver celui qui donne la quantité maximale d'informations possible sur la valeur de la classe.

Issue de la théorie d'informations, l'entropie est parmi les mesures les plus utilisées pour calculer le gain d'informations. Etant donné un ensemble de données contenant n tuples, si la variable de classe peut prendre k modalités C_1, … C_k, avec n_i le nombre de tuples prenant la valeur C_i pour la classe C, on a donc les fréquences suivantes : $P_1 = n_1/n, \ldots, P_k = n_k/n$. L'entropie correspond à une variable de classe C et se calcule en fonction du nombre de tuples prenant la valeur C_i pour la classe C : $I(C) = -\sum_{i=1}^{k} P_i \times log_2(P_i)$, avec P_i le nombre de tuples prenant la valeur C_i.

Pour décider du point de division des données, on calcule l'entropie de chaque attribut Ai qui peut prendre d modalités $a_1, \ldots, a_d$. L'information que l'attribut Ai donne pour classer un individu par rapport à une variable de classe C est : $I(A_i, C) = -\sum_{i=1}^{d} P_i \times I_{ai}(C)$, avec $Iai\ (C)$ la quantité d'information lorsqu'on restreint la population aux individus prenant la modalité ai. On remarque que plus l'entropie $I(A_i, C)$ est faible, plus la partition en sous-ensemble est pure (c.-à-d. que chaque sous ensemble tend vers une unique modalité).

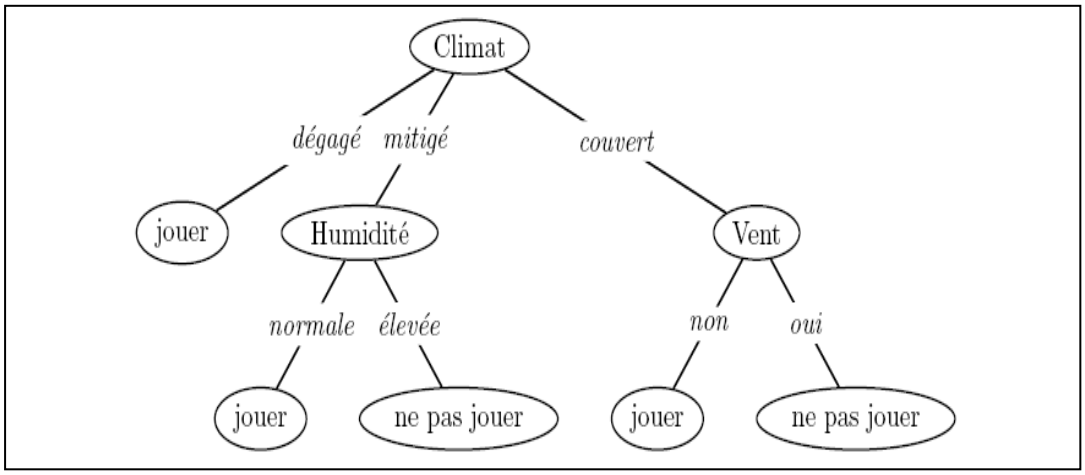


Figure 2.7: Exemple d'un arbre de décision construit à partir d'une base de données sur la décision (classe) "jouer ou non" selon certains attributs sur le climat

Le Gain d'Information donné par un attribut A_i est calculé par: $Gain(C) = I(C) - I(A_i, C)$. Il faut pour chaque nœud de l'arbre choisir l'attribut qui a le gain le plus élevé pour la population du nœud. Un exemple d'arbre de décision est présenté dans (Figure 2.7).

3.3.3 La phase d'élagage

Les arbres de décision peuvent souffrir du phénomène typique du sur-apprentissage (over-fitting) qui se manifeste par une augmentation du taux de précision sur la base d'apprentissage, et une baisse significative sur les objets de l'ensemble de test.

Elaguer un arbre de décision signifie la suppression de certains sous arbres dans le but d'obtenir un arbre ayant un meilleur pouvoir de généralisation (améliorer l'aspect prédictif de l'arbre, et réduire sa complexité). Une fois l'arbre est construit, on cherche à améliorer celui-ci en élaguant certains de ses arbres. L'algorithme C4.5 par exemple se base sur le calcul de « l'erreur pessimiste» dans la phase d'élagage.

3.4 L'algorithme C4.5

L'algorithme C4.5 est l'un des algorithmes classiques de la classification par les arbres de décision. Il se présente comme une amélioration des versions antérieures ID3 en y ajoutant de nouvelles fonctionnalités, parmi elles :

- La discrétisation des variables quantitatives.
- Le groupement des valeurs discrètes d'un attribut.
- La génération des règles à partir de l'arbre.

3.4.1 Phase d'expansion

L'algorithme C4.5 utilise la fonction entropie plus une mesure de répartition de données appelée $SplitInfo$. La fonction $Gain$ ainsi définie présente le défaut de privilégier les attributs ayant un grand nombre de modalités. La valeur du $Gain$ est normalisée en la divisant par la mesure $SplitInfo$, afin de réduire le $Gain$ des attributs ayant beaucoup de modalités :

$SplitInfo(A) = -\sum_{i=1}^{i=m}(ni/n)\, log(ni/n)$ avec m : le nombre de modalités de l'attribut A, ni : le nombre des éléments prenant la ième valeur de A.

La mesure de sélection des attributs dans l'algorithme $C4.5$ est définie par le $GainRation$:

$GainRation = Gain/SplitInfo$.

3.4.2 Phase d'élagage

Pour élaguer l'arbre de décision, l'algorithme $C4.5$ calcule l'erreur «pessimiste» (err). Pour chaque nœud v de l'arbre, on calcule l'erreur au niveau du nœud v, et au niveau du sous arbre T enraciné dans v. Le sous arbre T est élagué si $err(T) >= err(v)$. Plus de détails sur cette opération est dans.

3.4.3 Traitement des attributs

La fonction $GainRation$ permet d'éviter de privilégier les attributs de type discret. $C4.5$ donne aussi la possibilité de regrouper les valeurs des attributs de ce type. Par exemple, si on dispose d'un attribut A prenant les valeurs $a1, a2, a3$, et $a4$, le nombre de tests peut être réduit à deux tests binaires "$A \in \{a1, a2\}$" et "$A \in \{a3, a4\}$" ou à trois tests : "$A = a1$" , "$A = a3$" et "$A \in \{a2, a4\}$" .

$C4.5$ convertit les variables continues en variables discrètes en utilisant un seuil. Les valeurs sont d'abord triées, les tests utilisés sont généralement binaires de la forme $ai < v$, où v est la valeur médiane entre chaque deux valeurs consécutives. Par exemple, supposons que l'attribut A prenne les valeurs $\{0, 3, 6, 10, 12\}$, les tests considérés seront alors : $A > 1{,}5$; $A > 4{,}5$; $A > 8$ et $A > 11$.

3.5 Les méthodes ensemblistes

3.5.1 Principe général

Dans le cas de la classification par les méthodes ensemblistes, N classifieurs de base Cli sont construits, à partir de N ensembles de données Di (Figure 2.8). Le classement d'une nouvelle donnée se fait par la combinaison des prédictions des N classifieurs de base, par un vote majoritaire par exemple. Malgré la simplicité de cette idée intuitive «l'union fait la force», elle repose sur une théorie statistique renforcée par plusieurs études empiriques. Ces études ont montré dans différents travaux de recherche que la précision d'un algorithme d'apprentissage peut être améliorée d'une façon significative en appliquant le principe de perturbation et combinaison. Les algorithmes les plus appropriés à l'application de cette approche sont ceux considérés comme «instables», c.-à-d. que des petites modifications dans les données d'apprentissage pourrait induire à un grand changement dans la fonction Cl estimée. Les arbres de décision par exemple sont considérés comme de bons candidats.

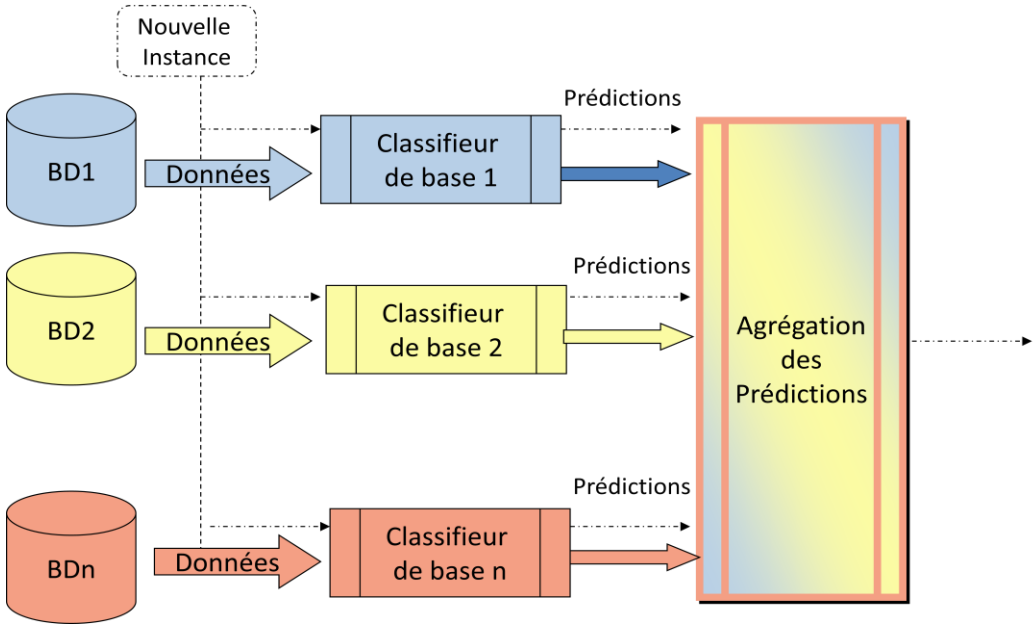


Figure 2.8: Schéma illustratif du processus des méthodes ensemblistes

Cette perturbation permet de générer plusieurs ensembles d'apprentissage, à partir d'un ensemble de base, comme dans les techniques de *bagging* et de *boosting*. Elle peut aussi être appliquée sur les algorithmes de construction des classifieurs, en utilisant plusieurs algorithmes différents, ou en modifiant certains paramètres. Les résultats expérimentaux montrent que 50 répliques sont en générale suffisantes, mais le temps de calcul est encore un champ d'investigation.

3.5.2 Exemple d'algorithme ensembliste: Adaboost

L'idée de base est de construire un nouveau classifieur, selon la performance d'une série de classifieurs précédents, dans un processus séquentiel. L'ensemble d'apprentissage d'origine est renforcé par des poids qui seront ajustés à chaque étape, dans l'objectif «d'amplifier» (boost) les exemples mal classés. Les modèles sont combinés par vote à majorité pondéré où la pondération est déterminée par la précision de prédiction de chaque classifieur. *Adaboost* est un exemple d'algorithme utilisant cette technique.

Algorithme Boosting

Entrée:

D, ensemble de n tuples d'apprentissage (enregistrements) étiquetés par des valeurs de classe: $\{(x_i, y_i)_{\ i=1..n}\}$ avec $X = \{x_i\}$ ensemble d'attributs, $y_i \in Y = \{c_1, \dots, c_d\}$ ensemble de valeurs de classe.

$WeakLearn$, algorithme d'apprentissage faible.

T, nombre d'itérations.

Sortie: Le classifieur final obtenu par un vote des T classifieurs

1) //Initialiser les poids des tuples pour obtenir la première distribution D_1
2) **Pour** $i = 1, 2, \dots, n$ **faire** $D_1(i) = \frac{1}{n}$
3) **Fin Pour**
4) //Construction séquentielle de T classifieurs
5) **Pour** $t = 1, 2, \dots, T$ **faire**
6) //Appliquer $WeakLearn$ sur D_t
7) //Construire le classifieur $Cl_t: X \to Y$
8) //Calculer l'erreur de Cl_t en fonction des poids mal classés
9) $\epsilon_t = \sum_{i:Cl_t(x_i) \neq y_i} D_t\ (i)$
10) $\beta_t = \epsilon_t / (1 - \epsilon_t)$
11) //Mise à jour des poids des tuples i correctement classé
12) **Si** $Cl_t(x_i) = y_i$ **alors** $D_{t+1}(i) = D_t(i) \cdot \beta_t$
13) //Normaliser les poids en utilisant une constante de normalisation Z_t
14) $D_{t+1}(i) = D_t(i) / Z_t$
15) //Calculer le poids du classifieur Cl_t
16) $w_t = \log(1/\beta_t)$
17) **Fin pour**
18) //Construction du classifieur final pour classer les tuples x
19) $Cl_{fin}(x) = \arg\max \sum_{t:Cl_t(x)=y} w_t$

Algorithme 2.5: Algorithme Adaboost

Yoav Freund et Robert Schapire ont introduit l'algorithme AdaBoost (Adaptive Boosting) pour lequel ils ont eu le prix Gödel[1] en 2003. AdaBoost, le plus utilisé pour implémenter le boosting, cherche à produire des classifieurs "forts" (très précis) en combinant des instances "faibles" d'un classifieur donné. Le boosting permet ainsi d'utiliser des classifieurs de "moindre qualité" mais qui peuvent avoir de bonnes propriétés notamment en termes de vitesse de calcul.

L'algorithme AdaBoost prend en entrée un ensemble D d'apprentissage de n tuples (enregistrements) étiquetés par des valeurs de classe: $\{(x_i, y_i)_{\ i=1..n}\}$ avec $X = \{x_i\}$ ensemble d'attributs, $y_i \in Y = \{c_1, \dots, c_k\}$ ensemble de valeurs de

[1] Distinction offerte par European Association for Theoretical Computer Sciences pour honorer des travaux remarquables d'informatique théorique, http://eatcs.org/index.php/goedel-prize.

classe (Algorithme 2.5). L'algorithme fait appel T fois (T: nombre d'itérations) à une méthode d'apprentissage comme les arbres de décision, dénotée $WeakLearn$.

A chaque itération t, on associe des poids aux tuples d'apprentissage et on obtient ainsi une distribution D_t. Le but de l'algorithme $WeakLearn$ est de calculer un classifieur $Cl_t: X \rightarrow Y$ qui minimise l'erreur « d'apprentissage »

La distribution initiale D_1 est uniforme sur D, $D_1(i) = \frac{1}{n}$ pour tous les tuples i. Pour calculer la distribution D_{t+1} à partir de D_t et le dernier classifieur Cl_t, le poids du tuple correctement classé est multiplié par un nombre $\beta_t \in [0,1]$ calculé en fonction de l'erreur ϵ_t (ligne 11), tandis que le poids des tuples mal classés restent inchangés. Ainsi, les poids des exemples «faciles» qui sont correctement classés diminueront par rapport aux autres.

Le poids d'un classifieur Cl_t est défini par $\log(1/\beta_t)$ (ligne 16) ce qui permet de donner plus de poids aux classifieurs dont l'erreur est petite. Le classifieur global est un vote pondéré entre les t classifieurs construits. C'est à dire, pour un tuple x, $Cl_{fin}(x)$ produit la classe y qui maximise la somme des poids des classifieurs ayant donnés cette valeur de classe (ligne 19).

4 Le clustering (Segmentation)

4.1 Principe général

Le clustering ou «classification non supervisée» consiste à partitionner les données en groupes « clusters », tels que les données d'un même cluster soient similaires et les données de deux clusters distincts soient différentes. Cette tâche peut être utile dans plusieurs domaines tels que :

- Biologie: trouver les groupes de gènes qui ont des fonctions similaires
- Médias sociaux : identifier les groupes d'utilisateurs similaires en terme de contenue des tweets

Formellement : Soit X un ensemble de N données décrites chacune par leurs P attributs, la segmentation consiste a créer une décomposition de cet ensemble en groupes telle que les données appartenant au même groupe se ressemblent, tandis que les données appartenant a deux groupes différents soient peu ressemblantes.

Le problème de segmentation optimale en k groupes est NP-complet. Il faut donc chercher des algorithmes calculant une bonne décomposition, sans espérer être sûr de trouver la meilleure. On distingue essentiellement deux

types de méthodes «non hiérarchiques» ou «hiérarchques» (cf. 4. 3, 4.4). L'outils de base dans cette tâche est la mesure de similarité entre les données.

4.2 Mesures de similarité

La méthode de calcul de distance entre deux objets dépend essentiellement de la nature des données à segmenter (Euclidienne, Jaccard, Hamming, Menhatten, ...). La figure (2.9) présente un exemple de calcul de la distance euclidienne entre deux objets.

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk}\right)^2}$$

Objet	a1	a2	a3	a4
A	0	3	4	5
B	7	6	3	-1

$$d_{BA} = \sqrt{(0-7)^2 + (3-6)^2 + (4-3)^2 + (5+1)^2}$$
$$= \sqrt{49+9+1+36} = 9.747$$

Figure 2.9 : Exemple de calcul de la distance euclidienne

4.3 Approche non hiérarchique

L'ensemble d'objets est décomposé en *k* groupes. L'algorithme le plus répandu dans cette approche est «k-means» (Algorithme 2.6). La méthode du choix de la valeur de k (nombre de centroides initial) est un champs d'investigation. Par défaut, k est choisi aléatoirement. Le critère d'arrêt peut être soit la stabilisation des objets, ou bien d'autre mesures comme l'inertie.

1) Déterminer le nombre de Clusters K
2) Prendre K objets arbitraires « centroides »
3) Calculer les distances entre les objets et les centroides
4) Grouper les objets selon la distance minimale
5) Si aucun objet ne change de groupes, alors fin
6) Sinon calculer les nouveaux centroides, aller à 4

Algorithme 2.6: Principe général de l'algorithme k-means

La figure (figure 2.10 a et b) présente un exemple d'application du principe k-means en utilisant la distance euclidienne.

Itération 0

Données à segmenter

Objet	X	Y
A	1	1
B	2	1
C	4	3
D	5	4

1. **Valeurs Initiales des centroides:**
   - K = 2;
   - A: C1(1,1) (Groupe 1), B: C2(2,1) (Groupe 2)
2. **Calcul des Distances**
   - D^m : matrice des distances à l'itération m.
   - $D^m[i,j]$ = distance entre l'objet j et le centroide i
   - Exp. Distance Euclidienne
     - d (C, A) = $\sqrt{(4-1)^2+(3-1)^2} = 3.61$
     - d (C, B) = $\sqrt{(4-2)^2+(3-1)^2} = 2.83$

$$D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix}$$

3. **Groupement des objets**

$$G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{matrix} group-1 \\ group-2 \end{matrix}$$
$$A \quad B \quad C \quad D$$

Itération 1

1. **Déterminer les nouveaux centroides:**
   - Groupe 1: Un seul membre ⇒ le centroide reste le même C1(1,1)
   - Groupe 2: Calculer la moyenne des coordonnées des trois membres

$$c_2 = (\frac{2+4+5}{3}, \frac{1+3+4}{3}) = (\frac{11}{3}, \frac{8}{3})$$

2. **Calcul des Distances**

$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix}$$

3. **Groupement des objets**

$$G^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} group-1 \\ group-2 \end{matrix}$$
$$A \quad B \quad C \quad D$$

Figure 2.10 (a): Exemple d'application de k-means (Itération 0 et 1)

Itération 2

1. **Déterminer les nouveaux centroides:**
   - Groupe 1:

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1\tfrac{1}{2}, 1)$$

   - Groupe 2:

$$c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4\tfrac{1}{2}, 3\tfrac{1}{2})$$

1. **Calcul des Distances**

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix}$$

2. **Groupement des objets**

$$G^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \begin{matrix} group-1 \\ group-2 \end{matrix}$$

$$A \quad B \quad C \quad D$$

$G^2 = G^1$ ⇒ *Fin des itérations*

Objet	X	Y	Groupe
A	1	1	1
B	2	1	1
C	4	3	2
D	5	4	2

Figure 2.10 (b): Exemple d'application de k-means (Itération 2)

4.4 Approche hiérarchique

On décompose l'ensemble d'objets en une arborescence de groupes (dendogramme). Le principe consiste à débuter de N groupes, que l'on réduit à N-1, puis a N-2.... etc. On passe donc de k+1 a k groupes en regroupant deux groupes (Algorithme 2.7). On obtient ainsi une hiérarchie sous forme d'un «dendogramme» (figure 2.11) dans lequel on peut facilement retrouver k groupes en le coupant à un certain niveau. Le principe de clustering hiérarchique est illustré dans l'exemple (figure 2.11) en utilisant la distance euclidienne.

1) Calculer la matrice des distances
2) Chaque objet représente un Cluster
3) Regrouper les deux clusters les plus proches
4) Mettre à jour la matrice des distances
5) Si le nombre de Clusters = 1, fin Sinon, aller à 3

Algorithme 2.7 : Principe de base du clustering hiérarchique

1) Données à segmenter

	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

2) Matrice des distances

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

3) Grouper les clusters D, F

Nouvelle Matrice de Distances

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

La Distance entre deux clusters est définie par la distance minimale entre les objets des deux clusters

$$d_{(D,F)\mapsto A} = \min\left(d_{DA}, d_{FA}\right) = \min\left(3.61, 3.20\right) = 3.20$$

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

4) Grouper les clusters A, B

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

$$d_{C\to(A,B)} = \min\left(d_{CA}, d_{CB}\right) = \min\left(5.66, 4.95\right) = 4.95$$

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	4.95	0	2.24	1.41
(D, F)	2.50	2.24	0	1.00
E	3.54	1.41	1.00	0

5) Grouper ((D, F), E)

Dist	(A,B)	C	(D, F), E
(A,B)	0.00	4.95	4.95
C	4.95	0.00	1.41
(D, F), E	2.50	1.41	0.00

6) Grouper ((D, F), E avec C)

Dist	(A,B)	(D, F), E),C
(A,B)	0.00	2.50
((D, F), E),C	2.50	0.00

7) Fusionner les deux clusters qui restent: Fin du calcul

Figure 2.11: Exemple d'application du clustering hiérarchique

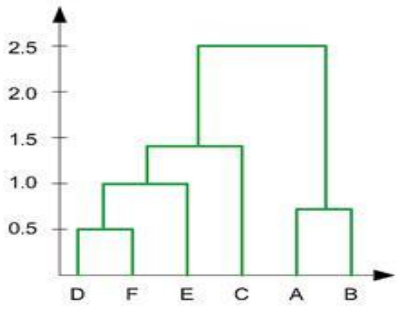


Figure 2.12 : Dendogramme correspondant à l'exemple de la figure 2.10

Exercices

1) Expliquer les différences principales entre :

   a) C4.5 et k-means

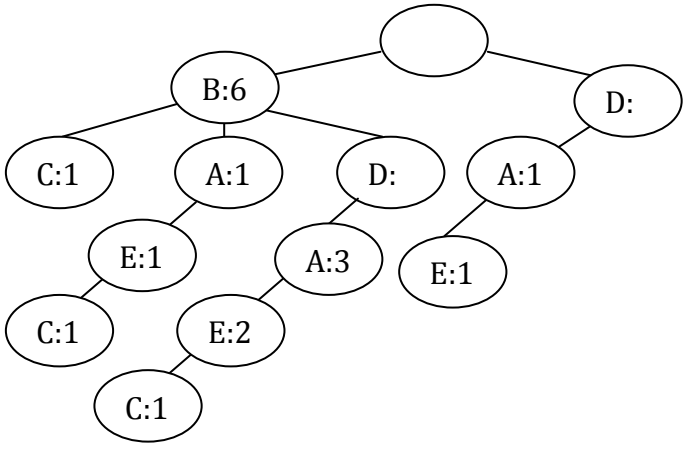
   b) Segmentation hiérarchique et non hiérarchique (k-means) en terme de nombre initial de clusters, dans le cas de n objets.

2) Démonter la propriété de la non monotonie utilisée dans Apriori : « Si un itemset est non fréquent, alors tous ses sur-ensembles sont non fréquents »

3) Est ce que tout itemset maximal fréquent est fermé fréquent ? pourquoi ?

4) Quel est l'intérêt d'extraire les itemsets fréquents fermés ou maximaux au lieu des fréquents?

5) Soit l'arbre FP-tree suivant :



   a) En déduire la valeur *minSupp*.
   b) Montrer les étapes d'exécution de l'algorithme FP-growth dans l'extraction des itemsets fréquents contenant l'item E.
   c) Quelles sont les règles extraites de l'itemset *ABE*
   d) Quels sont les avantages de cet algorithme par rapport à l'algorithme Apriori ?
   e) Quel est l'inconvénient majeur de l'algorithme FP-growth ?

6) Dans le cas de l'algorithme FP-growth :

Transactions
C A B E
B A D
F A D B
D A C E B

a) Schématiser les étapes d'exécution de l'algorithme sur l'exemple ($minSupp$ = 0,6).
b) Parmi les itemsets fréquents trouvés, quels sont les itemsets fermés et maximaux (en même temps).
c) Dans le cas où il n'existe pas de parties communes entre les transactions, quelle est le problème qui peut se poser.

7) On voudrait voir l'effet de l'augmentation de la valeur $minSupp$ sur le nombre d'itemsets fréquents générées en appliquant l'algorithme Apriori ou FP-growth. Faire cette expérimentation et tracer la courbe correspondante sur la base « T10I4D100K ».

8) On dispose d'un échantillon de données (X1, X2, .. X6) dont la valeur de classe est connue (P/Q) (voir figure)

ID	Att1	Att2	.....	Classe
X1	.....	.....	.....	P
X2	.....	.....	.....	Q
X3	.....	.....	.....	Q
X4	.....	.....	.....	P
X5	.....	.....	.....	Q
X6	.....	.....	.....	P

a) En vue d'appliquer une classification locale par arbres de décision, calculer l'entropie de cet échantillon.
b) Quelle est le rôle de la méthode «validation croisée» ? Expliquer comment peut-on l'appliquer sur ces données avec k = 3.

9) On dispose d'une base d'apprentissage avec des attributs binaires X, Y, Z et un attribut classe W qui peut avoir deux valeurs «$True$ » ou «$False$ », avec un effectif de 3 et 2 respectivement.

a) Calculer l'entropie associée à la classe W.
b) A quoi sert le calcul du Gain d'information à chaque étape de la construction de l'arbre?

	Gain d'Information			Nbre de feuilles pour l'attribut de division
	X	Y	Z	
Etape 1	0.02	0.42	0.17	1 (True)
Etape 2	0,25	-	0,91	2

c) Déduire l'arbre de décision d'après les données d'exécution de l'algorithme (voir tableau)

10) Comment peut on adapter les règles d'association afin de construire un modèle de classification supervisée ?

11) On désire appliquer l'algorithme K-means (K=3) sur les données suivantes (Nombre de cluster = 3, Distance de similarité : Euclidienne). Déterminer les nouveaux clusters et leurs centres après la première itération.

Objet	X	Y
A	2	10
B	2	5
C	8	4
D	5	8
E	7	5
F	6	4
G	1	2
H	4	9

Chapitre 3

Fouille de Données dans les Médias Sociaux

1 Introduction

Les médias sociaux représentent une mine de connaissances cachées à travers un ensemble de données de plus en plus volumineuses, hétérogènes, et distribuées. L'application des outils du datamining sur de telles données ou bien « Social Media Mining» vise la représentation, l'analyse et l'extraction de modèles significatifs, à partir des données des médias sociaux.

Ce chapitre présente les aspects principaux pouvant aider à analyser ces données. On aborde essentiellement certaines propriétés statistiques ainsi que des mesures permettant de comprendre la structure du réseau. L'assimilation de certaines parties du chapitre nécessite une connaissance préalable des notions de base en théorie des graphes.

Plusieurs techniques du datamining peuvent être utilisées et adaptées à la fouille des médias sociaux comme les règles d'associations, les arbres de décision et la segmentation (clustering). On présente aussi dans ce chapitre trois tâches importantes dans le domaine du «social media mining» : la détection des communautés, la classification collective, et les systèmes de recommandation.

2 Les médias sociaux comme données du datamining

Le rôle principale d'un média social est de permettre aux utilisateurs un partage mutuelle d'un certain contenu. Le réseau Geocities (1994) est considéré comme le premier media social dans ce sens ; il permet aux utilisateurs de créer leur propre page web. Plusieurs références considèrent que le premier site au sens «réseau social» est SixDegree.com (1997). Les médias sociaux sont actuellement omniprésents sur internet, on cite parmi les plus répandus: Facebook et Twitter.

2.1 Définition du média social

Un média social est un groupe d'applications Internet (services en ligne) conçues sur les fondations du Web 2.0 (simplicité, sociabilité, intelligence collective,..), et qui permettent à chaque utilisateur de générer (créer) un certain contenu et de le partager avec les autres utilisateurs.

2.2 Taxonomie des médias sociaux

Une classification des médias sociaux peut être basée sur les fonctionnalités offertes par le média :

- Réseaux sociaux (Facebook, LinkedIn)
- Microblogging (Twitter)
- Partage de Photos (Flickr, Photobucket, or Picasa)
- Partage de Vidéos (YouTube, MetaCafe)
- livecasting (Transmission d'un flux continu d'événements dans la vie d'une personne à travers les médias numériques: Ustream, Justin.TV)
- Jeux Sociaux (World of Warcraft)
- Messagerie instantanée (Google Talk, Skype, Yahoo! messenger)

2.3 Représentation des données issues des médias sociaux: Graphe social

Les données issues des médias sociaux peuvent être modélisées par un graphe $G(V, E)$, V : ensemble de n sommets et E : ensemble de m liens. On distingue plusieurs types de graphes : graphe orienté/ non orienté/ pondéré/ biparti... etc. ... Par exemple, le réseau Facebook peut être modélisé par un graphe non orienté $G(V, E)$, V: Ensemble d'individus, E: liens d'amitié. Par contre, le média social Twitter peut être représenté par un graphe orienté $G(V, E)$ orienté, V: Ensemble d'individus, E: liens de «Follower».

Les réseaux complexes tels que les médias sociaux en ligne se caractérisent par un nombre très élevé de nœuds et de liens, ce qui rend la représentation en mémoire centrale de ces données plus complexe. Comme exemple, on peut visiter le site consacré aux statistiques sur le réseau Facebook : http://www.facebook.com/press/info.php?statistics.

3 Propriétés et modèles des médias sociaux

Selon des études en sciences sociales, un ensemble de caractéristiques observées dans les réseaux sociaux réels, ont aussi été trouvés dans beaucoup de médias sociaux (études empiriques statistiques). On présente trois propriétés parmi les plus citées :

- Distribution des Degrés selon une loi de puissance
- Coefficient de clustering élevé
- Effet petit monde

3.1 Distribution des Degrés selon une loi de puissance

Le degré d'un nœud est défini comme le nombre de liens connectés à ce nœud. La distribution des degrés est la description des fréquences relatives des nœuds de degrés différents d. Cette distribution est décrite par une fonction $f(P_d) = d$ avec $P_d = n_d/n, n_d$: le nombre de noeuds ayant le degré d, et n le nombre de nœuds dans le graphe. Comme exemple, dans la figure 2.1, $P_2 = 4/7$. La distribution des degrés dans les médias sociaux suit généralement une loi de puissance, c.-à-d. la fraction des nœuds ayant i liens = $1/i^{\theta}$ (θ>0).

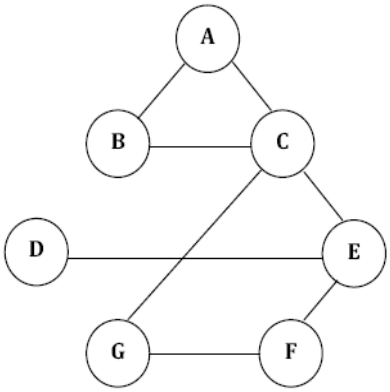


Figure 2. 1: Exemple d'un graphe non orienté où $P_2 = 4/7$

Comme interprétation, on peut dire que dans un média social beaucoup de nœuds ont des degrés faibles, et très peu de nœuds ont des degrés très grands (figure 2.2).

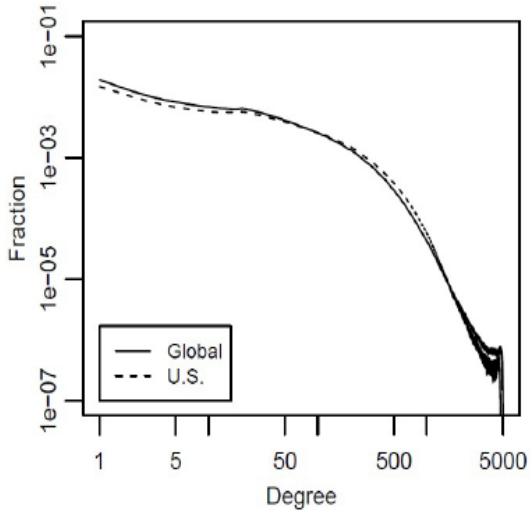


Figure 2. 2: Distribution des degrés dans Facebook, Mai 2012,

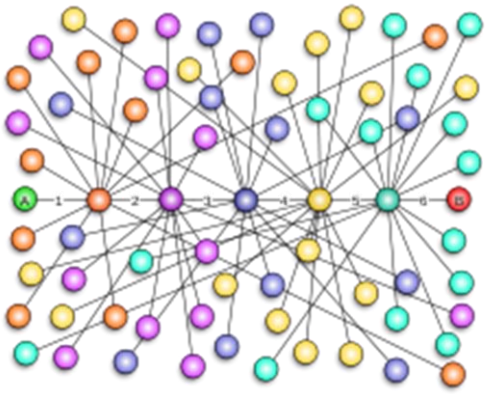
3.2 Coefficient de clustering élevé

Le coefficient de clustering est une mesure qui permet d'évaluer à quel point les liens dans un réseau sont de nature transitive. En pratique dans un média social,

ce coefficient mesure à quel point “les amis de mes amis sont mes amis”. Si cette valeur est élevée, on aura une tendance à former des groupes de nœuds plus fortement connectés entre eux qu’avec les autres, ou “Structure de communautés ”.

3.3 Effet petit monde

Dans les réseaux réels, des études montrent que chaque deux éléments du réseau sont généralement connectés via des chemins courts. Des expériences de Stanley Milgram en 1960 visaient à calculer le nombre moyen de nœuds (longueur du chemin) qui relient deux personnes. Il a fait transiter une lettre depuis Omaha jusqu’à Boston, de telle sorte que chacune des personnes qui reçoit la lettre doit la remettre en mains propres à une personne de son choix, en permettant de rapprocher la lettre de l’objectif.



Longueur moyenne des chemins (Mai 2011)

Web	Facebook	Flickr	LiveJournal	Orkut	YouTube
16.12	4.7	5.67	5.88	4.25	5.10

Figure 2.3 : Effet petit monde observé dans certains médias sociaux

Ces expériences ont révélé que les personnes à travers le monde sont connectés entre eux via un chemin d’une longueur maximale égale à 6. Des valeurs rapprochées ont été aussi observées dans certains médias sociaux (figure 2.3). Des expériences similaires effectuées au niveau mondial en utilisant le courrier électronique, ont révélé une longueur moyenne des chemins égale à 5.

3.4 Modèles de réseaux

La complexité des réseaux d’entités créés dans les médias sociaux (des millions de nœuds connectés par des milliards de liens) rend difficile la tâche de fouille de données. Plusieurs théories mathématiques tentent de concevoir des modèles qui simulent les propriétés observées dans les réseaux réels. Ces modèles peuvent aider à l’étude et l’analyse des médias sociaux pour plusieurs raisons :

- Générer dans une échelle plus petite, des graphes similaires aux réseaux réels ;
- Une meilleure compréhension des phénomènes observés dans le monde réel, en apportant des arguments mathématiques rigoureuses ;
- Plus de contrôle sur les expériences effectuées sur des réseaux synthétiques

Les principaux modèles sont : modèle aléatoire (Modèles Erdős-Rényi), modèle petit monde (Modèle Watts-Strogatz), modèle scale-free (Modèle Barbaris-Albert).

a) **Modèle aléatoire (Modèles Erdős-Rényi) :** Ce modèle suppose que les liens entre les nœuds sont générés aléatoirement. On distingue deux types de modèles de graphes aléatoire :
   - Modèle *gnp* : La présence de chacun des liens est décidée aléatoirement selon une probabilité p.
   - Modèle gnm : m liens sont choisis uniformément parmi les liens possibles du graphe.

b) **Modèle petit monde (Modèle Watts-Strogatz) :** Ce modèle se base sur le phénomène de « six-degré de séparation».

c) **Modèle scale-free (Barabási-Albert) :** Ce modèle suppose que Lorsque de nouveaux nœuds sont ajoutés à un réseau, ils sont plus susceptibles de se connecter à des nœuds qui ont déjà beaucoup de liens (tendance à choisir les nœuds de degré supérieur).

4 Mesures et métriques sur les réseaux

L'application de différentes mesures sur les données issues des médias sociaux permet d'extraire de nouvelles connaissances sur ces réseaux. Ces mesures peuvent aider à répondre à plusieurs questions, comme par exemple :

- Quels sont les éléments influents dans le réseau ?
- Quels sont les nœuds similaires ?
- Quels sont les modèles d'interactions communs entre les éléments du réseau ?

4.1 Mesures de centralité

La centralité mesure l'importance ou l'influence d'un nœud à l'intérieur d'un réseau. Cette « importance » peut être mesurée par différentes méthodes.

4.1.1 Centralité de degré

Les personnalités ayant beaucoup de relations sont souvent considérées comme « importantes ». Cette notion peut être mesurée par le degré d'un nœud (nombre de liens adjacents). La centralité de degré mesure l'importance relative d'un nœud dans un graphe en fonction du nombre de ses liens.

Graphe non orienté :

$C_d(v_i) = d_i$ avec d_i le degré du nœud v_i

Graphe orienté :

On peut utiliser le degré supérieur ou inférieur ou bien une combinaison des deux:

$C_d(v_i) = d_i^{in}$ (mesure la popularité du nœud)

$C_d(v_i) = d_i^{out}$ (mesure la sociabilité du nœud)

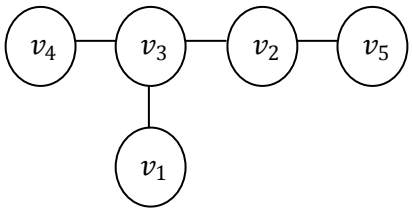
$C_d(v_i) = d_i^{in} + d_i^{out}$ (la direction est ignorée)

La centralité de degré peut être mesurée pour un groupe de nœuds. Soit S un ensemble de nœuds dont on voudrait calculer la centralité, $V - S$ dénote l'ensemble des nœuds n'appartenant pas au groupe :

$C_d^{group}(S) = \left|\{v_i \in V - S \; tel \; que \;\; v_i \text{est liée au nœud } v_j \in S\}\right|$

Exemple :

$C_d(v_3) = 3$; $C_d^{S=\{v2,v3\}} = 3$



4.1.2 Centralité de vecteur propre (spectrale) (eigenvector centrality)

Cette mesure repose sur l'idée qu'il ne suffit pas d'avoir beaucoup de relations pour être considéré comme «important», mais le fait d'avoir plus «d'importantes relations » est un argument plus fort. Dans un réseau social, cela correspond à l'idée qu'un acteur est d'autant plus important qu'il est connecté à des acteurs qui sont eux même importants.

Soit G un graphe non orienté de n nœuds, A la matrice d'adjacence correspondante. La centralité d'un nœud v_i « $C_e(v_i)$ » est proportionnelle à la

somme des valeurs de centralités de ses voisins « $C_e(v_j)$ ». Bonacich (1972) définit la centralité spectrale d'un nœud v_i comme multiple positif de la somme des valeurs de centralité de ses voisins :

$\lambda C_e(v_i) = \sum_{j=1}^{n} C_e(v_j) \forall i$ avec λ une constante positive

$\lambda C_e(v_i) = \sum_{j=1}^{n} A_{ji} C_e(v_j) \ \forall i$ (1)

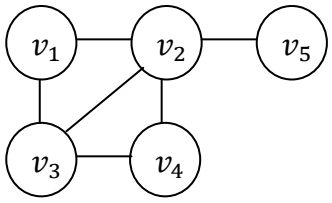
$$C_e(v_i) = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ji} C_e(v_j) \ \forall i$$

Si on considère C_e un vecteur dont les valeurs sont les centralités spectrales de tous les nœuds v_k, $k = 1..n : C_e = (C_e(v_1), C_e(v_2), \dots, C_e(v_n))$, l'équation (1) peut être réécrite:

$\lambda C_e = A C_e$

C_e est le vecteur propre de la matrice d'adjacence A, λ est la valeur propre correspondante. Donc le calcul des valeurs de centralités des n nœuds consiste à trouver le vecteur propre et la valeur propre correspondante pour la matrice d'adjacence qui représente le graphe. Il existe pour cela plusieurs algorithmes basés sur des théorie mathématique comme la méthode de «la puissance itérée».

Exemple



$\lambda C_e(v_1) = C_e(v_2) + C_e(v_3)$

$\lambda C_e(v_2) = C_e(v_1) + C_e(v_3) + C_e(v_4) + C_e(v_5)$

$\lambda C_e(v_3) = C_e(v_1) + C_e(v_2) + C_e(v_4)$

$\lambda C_e(v_4) = C_e(v_2) + C_e(v_3)$

$\lambda C_e(v_5) = C_e(v_2)$

$$\lambda \begin{bmatrix} C_e(v_1) \\ C_e(v_2) \\ C_e(v_3) \\ C_e(v_4) \\ C_e(v_5) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} C_e(v_1) \\ C_e(v_2) \\ C_e(v_3) \\ C_e(v_4) \\ C_e(v_5) \end{bmatrix} \qquad C_e = \begin{bmatrix} 0.412 \\ 0.583 \\ 0.217 \\ 0.524 \\ 0.412 \end{bmatrix}$$

4.1.3 Centralité d'intermédiarité

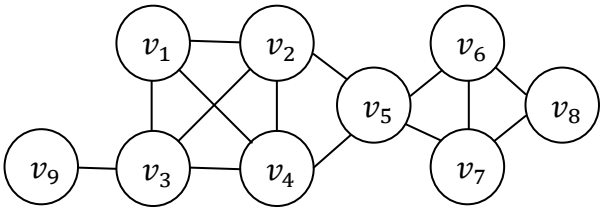
La centralité d'intermédiarité (Betweenness centrality) est basée sur l'hypothèse que les nœuds ne communiquent ou interagissent entre eux qu'à travers les chemins les plus courts. Etant donné un nœud v, cette mesure) $C_B(v)$ est définie par le nombre de plus courts chemins entre deux autres nœuds x et y qui passent par le nœud v noté $\sigma_{xy}(v)$, divisé par le nombre total de plus courts chemins allant de x à y $\left(\sigma_{xy}(v)\right)$.

$$C_B(v) = \sum_{x \neq y \neq v} \frac{\sigma_{xy}(v)}{\sigma_{xy}}$$

Pour calculer $C_B(v)$, on calcule les plus courts chemins entre toutes les différentes paires de nœuds x et y dans le graphe.

Exemple :

$C_B(v_1) = 0$
$C_B(v_2) = 6$
$C_B(v_3) = 7$
$C_B(v_4) = 6$
$C_B(v_5) = 15$
$C_B(v_6) = 3$
$C_B(v_7) = 3$
$C_B(v_8) = 0$
$C_B(v_9) = 0$



4.1.4 Centralité de fermeture (Closeness Centrality)

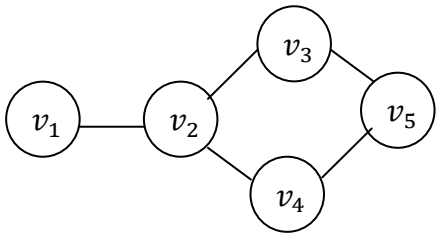
Intuitivement, un nœud est considéré comme le plus central s'il peut atteindre les autres nœuds le plus rapidement. Formellement, ce nœud est lié aux autres nœuds par un chemin de longueur moyenne la plus courte. La centralité de fermeture est défini par :

$$C_c(v_i) = \frac{1}{\bar{l}_{vi}}$$

avec $\bar{l}_{vi} = \frac{1}{n-1} \sum_{vj \neq vi} l_{i,j}$ est la longueur moyenne des plus courts chemins à partir de v_i vers les autres nœuds.

La centralité de fermeture peut être mesurée pour un groupe de nœuds. Soit S un ensemble de nœuds dont on voudrait calculer la centralité, $V - S$ dénote l'ensemble des nœuds n'appartenant pas au groupe :

Exemple :



$$C_c(v_2) = \frac{1}{(1+1+1+2)/4} = 0{,}8$$

4.2 Mesure de Transitivité

La transitivité permet d'analyser le comportement des liens à l'intérieur d'un média social. La mesure la plus utilisée est le coefficient de clustering qui mesure à quel point le voisinage d'un sommet est connecté. Il est défini globalement pour tout le réseau, ou bien localement pour un nœud.

4.2.1 Coefficient de clustering global

Le coefficient de clustering global analyse la transitivité dans un graphe non orienté. Le Coefficient de clustering global est basé sur la notion de triplet de nœud. Un triplet indique un ensemble ordonné de 3 nœuds connectés par 2 liens (triplet ouvert) ou 3 liens (triplet fermé ou triangle). On calcule le rapport entre le nombre de triplets fermés et le nombre total de triplets (ouverts et fermés) :

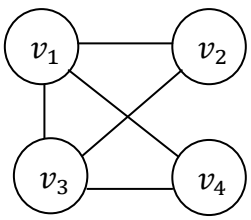
$$CCG = \frac{Nombre\ de\ triplets\ fermés}{Nombre\ total\ de\ triplets}$$

On note que chaque triangle dans le graphe (sou graphe possédant trois sommets et trois liens) consiste en 3 triplets fermés, chacun commençant par un des trois nœuds, donc on peut écrire :

$$CCG = \frac{(nombre\ de\ triangles) \times 3}{Nombre\ total\ de\ triplets}$$

$$CCG = \frac{(nombre\ de\ triangles) \times 3}{Nombres\ de\ triplets\ de\ noeuds\ connectés}$$

Exemple :



$$CCG = \frac{2 \times 3}{2 \times 3 + 2} = 0.75$$

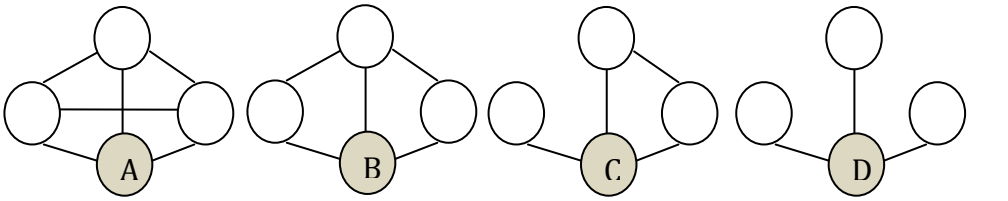
4.2.2 Coefficient de clustering local (CCL)

Le coefficient de clustering local d'un noeud x est défini par la fraction entre le nombre de liens qui relient les voisins de x entre eux, et le nombre maximum de liens qui peuvent exister entre ces voisins. Il est généralement défini pour les réseaux non orientés :

$$CCL(v_i) = \frac{Nombre\ de\ liens\ qui\ relient\ les\ voisins\ de\ v_i\ entre\ eux}{Nombre\ \ maximal\ de\ liens\ qui\ peuvent\ exister\ entre\ ses\ voisins}$$

En pratique dans un média social, ce coefficient mesure à quel point "les amis de mes amis sont mes amis". Il estime la force des liaisons entre les voisins d'un nœud. Si cette valeur est élevée, on aura une tendance à former des groupes de nœuds plus fortement connectés entre eux qu'avec les autres, ou "Structure de communautés ".

Exemple :



$CCL(A) = 1$ $CCL(B) = 2/3$ $CCL(C) = 1/3$ $CCL(D) = 0$

4.3 Mesures de similarité

Pour estimer à quel point deux nœuds dans un réseau sont similaires, on fait appel à des mesures de similarité. Cette similarité peut être calculée en se basant soit sur des informations sur les nœuds et les liens qui les relient (similarité structurelle), ou bien sur le contenu généré par les nœuds en question (similarité de contenu).

4.3.1 Similarité structurelle

Une idée simple pour évaluer la similarité structurelle entre deux nœuds consiste à calculer le nombre de voisinages en communs. Par exemple dans un réseau social comme facebook, deux frères ont tendance à avoir tous les deux comme amis d'autres membres de la même famille. Soit $N(v_i)$ et $N(v_j)$

l'ensemble de voisins des nœuds v_i et v_j respectivement. La similarité $\sigma(v_i, v_j)$ est définie par :

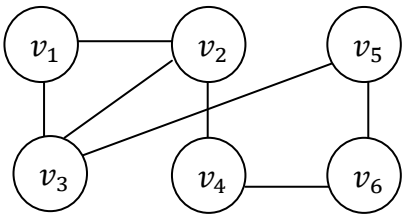
$$\sigma(v_i, v_j) = |N(v_i) \cap N(v_j)|$$

Pour éviter d'avoir de grandes valeurs de σ, particulièrement dans les grands réseaux, on la limite dans l'intervalle [0,1]. Comme exemple de normalisation, on cite : la similarité de Jaccard et la similarité Cosinus :

$$\sigma_{Jaccard}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$$

$$\sigma_{Cosinus}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{\sqrt{|N(v_i)| \parallel |N(v_j)|}}$$

Exemple :



$\sigma_{Jaccard}(v_2, v_5) = 0.25$ $\sigma_{Cosinus}(v_2, v_5) = 0.40$

4.3.2 Similarité de contenu

La similarité de contenu dans un média social est basée sur le profil de l'utilisateur (Age, Pays, Points d'intérêts, ...) ainsi que son comportement dans sa page sociale. Cette mesure est aussi traitée dans le cadre de la classification non supervisée, comme dans le cas de l'algorithme k-means (c.f. Chapitre 3).

5 Application des techniques du Datamining dans les Médias Sociaux

Les médias sociaux représentent une mine de connaissances cachées à travers un ensemble de données de plus en plus volumineuses, hétérogènes, et distribuées. L'application des outils du datamining sur de telles données ou bien « Social Media Mining» vise la représentation, l'analyse et l'extraction de modèles significatifs, à partir des données des médias sociaux.

Plusieurs techniques du datamining peuvent être utilisées et adaptées à la fouille des médias sociaux comme les règles d'associations, les arbres de décision et la segmentation (clustering). On présente dans cette partie trois tâches importantes dans le domaine du «social media mining» : la détection des communautés, la classification collective, et les systèmes de recommandation.

5.1 Détection de communautés

5.1.1 Définition d'une communauté

Etant donné un graphe *G(V, E), u*ne structure de communautés dans G est l'ensemble S = {C1,C2,…Ck} avec V = C1 ∪ C2 ∪ …∪ CK. Chaque élément Ci vérifie la définition de communauté considérée. Une communauté est donc un ensemble de sommets ayant beaucoup de liens entre eux et peu de liens avec les autres nœuds du graphe (figure 3.1).

5.1.2 Objectifs de la détection des communautés

On vise à trouver la structure (Partition des sommets) S = {C1,C2,…Ck} en minimisant k et en maximisant la qualité de la partition. Si Ci ∩ Cj ≠ ϕ, la structure de communautés est dite « avec recouvrement ou chevauchement », sinon la structure est aussi appelée partition.

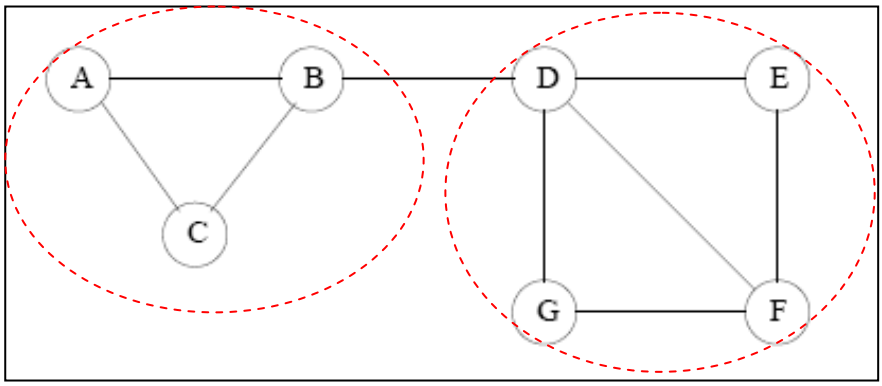


Figure 3.1: Un graphe simple avec deux communautés

La détection des communautés dans des données issues des médias sociaux permet de comprendre la structure du réseau. Une communauté peut correspondre, à un groupe d'amis dans facebook, ou un groupe de chercheurs travaillant sur des mêmes thèmes dans Linkedln, .. etc. Les résultats trouvés peuvent induire à une connaissance «utile» comme par exemple : cibler un groupe spécifique d'utilisateurs par des recommandations, détecter des communautés « dangereuses », ... etc.

5.1.3 Techniques de détection de communautés

Approches classiques

Il s'avère que les techniques du clustering classique sont généralement inadaptés au problème de segmentation des graphes représentant les médias sociaux. Ceci est dû essentiellement à la nature spécifique de ces données où les liens entre entités jouent un rôle important dans la structure du réseau. L'application directe de l'algorithme des centres mobiles k-means pose plusieurs problèmes, entre autres :

- Le choix aléatoire des centres peut donner des nœuds d'une même communauté, comme dans l'exemple (figure 3.1) où on peut choisir A et B comme centres initiaux.
- La définition de la distance entre objets est liée essentiellement à la structure des liens entre ces objets. Dans l'exemple ci-dessus (figure 3.1), si B et F sont choisis comme centres initiaux, F apparait «intuitivement» comme membre du cluster F, malgré que D est à une même distance des deux nœuds B et F. D'autres part, si la distance est définie par la moyenne des distances entre D et tous les autres nœuds du cluster, le nœud D sera affecté à la communauté de F, mais ceci pose un problème dans l'ordre de traitement des nœuds.

Approches dédiées aux structures de graphes

De nouvelles approches basées sur la structure des graphes sont apparues afin de palier aux inconvénients des méthodes classiques. Une des approches les plus répandues est l'algorithme de Girvan et Newman (GN). Ce dernier consiste à chercher les liens représentant des «ponts» qui relient les différents clusters. Ceci va aider à trouver les parties qui séparent les communautés et obtenir ainsi un partitionnement du réseau.

L'algorithme GN prend en entrée un graphe non orienté non pondéré, et divise le graphe en plusieurs communautés en retirant progressivement les arêtes reliant les communautés distinctes. La recherche de ce type de liens est basée sur une mesure appelée «Edge Betweenness Centrality EBC» ou « Centralité d'Intermédiarité». D'autres part, la qualité de la partition est évaluée à l'aide d'une mesure appelée «modularité».

5.1.4 Mesure de Centralité d'Intermédiarité (Edge Betweenness Centrality EBC)

La mesure EBC correspondant à un lien *e* évalue la proportion de plus courts chemins passant par ce lien (figure 3.2). Ceci peut être expliqué par l'intuition

suivante : puisqu'il existe peu d'arrêtes reliant les différentes communautés, les plus courts chemins entre deux sommets de deux communautés différentes ont de fortes chances de passer par ces arrêtes :

$$EBC(lien) = \frac{Nbre\ total\ de\ plus\ courts\ chemins\ passant\ par\ ce\ lien}{Nbre\ total\ de\ plus\ courts\ chemins\ dans\ le\ graphe}$$

$$EBC(lien) = \frac{\sigma_{ij}(lien)}{\sigma_{ij}}$$

avec σ_{ij} : Nombre de plus courts chemins entre i et j ; $\sigma_{ij}(lien)$: Nombre de plus courts chemins entre i et j passant par ce lien .

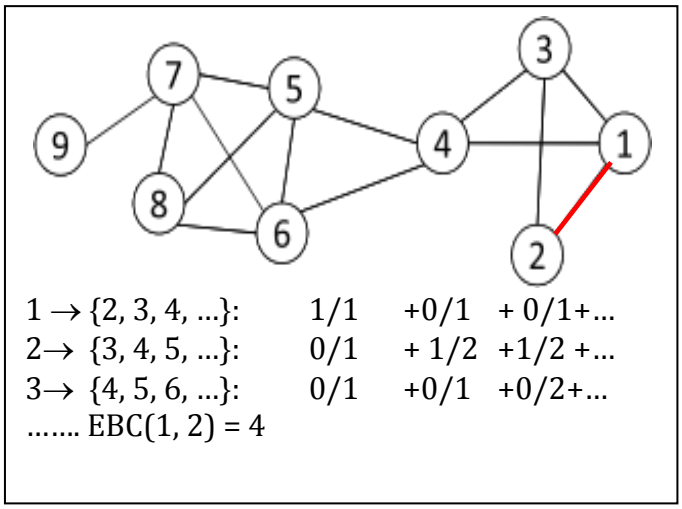


Figure 3.2: Exemple de calcul de la mesure EBC

Le calcul des valeurs EBC est basé sur le parcours en largeur BFS (breadth-first search) (voir les slides du cours).

5.1.5 Algorithme de principe : Girvan et Newman

La méthode de détection des clusters est un algorithme en deux passes :

- Etape1 : Déconnecter les clusters en calculant les EBC des liens
  - Calculer les valeurs EBC pour chaque arête du graphe.
  - Supprimer celle possédant la plus grande valeur.
  - Recalculer les EBC pour toutes les arêtes du graphe résultant jusqu'à ce que tous les arêtes aient été enlevées.
- Etape2 : Construction des clusters
  - Au début, chaque nœud représente un cluster
  - Les arêtes entre clusters sont ajoutées dans l'ordre inverse de celui dans lequel elles ont été supprimées dans la première étape.
  - Si les deux nœuds groupés par l'arête font partie de deux clusters différents, Alors tous les nœuds des deux clusters sont groupés dans un seul cluster. Sinon l'ensemble des clusters reste inchangé.

5.1.6 Evaluation de la qualité de la partition : Modularité

Il faut déterminer quand la phase de construction des clusters doit s'arrêter pour maximiser la pertinence de la clusterisation. Newman et Girvan introduisent une mesure appelée « modularité ».

La modularité est définie par La proportion de liens qui connectent deux nœuds appartenant à la même communauté (rapport entre le nombre d'arêtes internes et d'arêtes externes aux communautés) :

Soit G un graphe non orienté de n nœuds et m arêtes représenté par une matrice d'adjacence A. L'élément A_{ij} de la matrice:

$$A_{ij} = \begin{cases} 1 & si\ une\ arête\ lie\ les\ noeud\ i\ et\ j \\ 0 & sinon \end{cases}$$

On suppose que les sommets sont divisés en communautés tel que le nœud $i \in C_i$ avec C_i une communauté. La fraction d'arêtes qui appartiennent à des communautés (c.-à-d. qui connectent deux nœuds appartenant à la même communauté) est définie par :

$$Q_1 = \frac{\sum_{ij} A_{ij}\delta(C_i, C_j)}{\sum_{ij} A_{ij}} = \frac{1}{2m}\sum_{ij} A_{ij}\delta(C_i, C_j)$$

D'après cette première définition, on constate que plus la valeur est élevée, plus on se rapproche d'une «bonne» partition. Néanmoins, si tous les nœuds font partie d'une même communauté, la valeur Q1 est maximale (figure 3.3). Ceci a conduit à une soustraction de « Q2» qui représente la probabilité d'avoir un lien entre deux nœuds i, j dans le cas d'un graphe aléatoire :

$Q2 = \frac{d_i d_j}{2m}$ Avec di et dj: degrés des nœuds i et j

La modularité mesure le nombre d'arêtes à l'intérieur des communautés moins ce même nombre obtenu sur un graphe aléatoire (c.a.d. sans structure) de même taille mais gardant exactement la même distribution de degrés.

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	0	1
3	1	0	0	1	0
4	0	0	1	0	1
5	0	1	0	1	0

C1 = (1, 4, 3) C2 = (2, 5) $\boldsymbol{Q_1 = 0,4}$

C = (1, 4, 3, 2, 5) $\boldsymbol{Q_1 = 1}$

Figure 3.3 : Exemple de calcul de la modularité

L'idée est que dans un graphe aléatoire, certains liens ont naturellement une forte chance d'exister (notamment entre 2 nœuds de très fort degrés). L'existence de ce lien dans le graphe réel ne sera donc pas un argument pertinent pour considérer que ces 2 nœuds sont effectivement dans la même communauté. Par contre, l'existence d'un lien entre 2 nœuds ayant peu de chances d'être liés dans le graphe aléatoire sera un argument fort pour les regrouper. En fin, la modularité est définie par :

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{d_i d_j}{2m}\right]\delta(C_i, C_j) \qquad (Q \in [-1,1])$$

Des études statistiques suggèrent qu'une « bonne » structure de communauté possède une modularité >0,3.

5.2 Classification collective

La construction du modèle de classification dans le processus d'extraction des connaissances est basée essentiellement sur les attributs des objets, indépendamment des relations qui peuvent exister entre ces objets. Les méthodes classiques de la classification doivent être adaptées à la nature des

données issues des médias sociaux, où la notion de « liens » entre entités joue un rôle important dans la connaissance qui peut être extraite.

5.2.1 Définition et Notations

Etant donné un graphe G=(V, E, X, Y) avec :

- V : ensemble de n nœuds,
- E : ensemble de m liens,
- XL : ensemble de vecteurs d'attributs locaux. Chaque nœud v_i est décrit par un vecteur d'attributs $xl_i \in XL$.
- XR : ensemble de vecteurs d'attributs relationnel. Chaque nœud v_i est décrit par un vecteur d'attributs $xr_i \in XR$.
- Y : ensemble des étiquettes de classe. Chaque nœud v_i est décrit par une valeur de classe $y_i \in Y$.

L'objectif est de trouver une fonction qui permet de prédire la classe des nœuds dont la valeur est inconnue.

Pour un nœud v_i, sa valeur de classe y_i ne dépend pas seulement des attributs x_i, mais elle peut dépendre aussi des valeurs de classe y_j de tous les utilisateurs voisins v_j , ainsi que de leurs attributs x_j. La classification collective utilise les attributs et les valeurs de classe des voisins.

5.2.2 Principe de base : Classification itérative

Un des premières approches proposées est l'algorithme ICA (Iterative Classification Algorithm). Le principe de base est illustré dans l'exemple (figure 3.2) où on suppose avoir un graphe qui représente quatre comptes sur le média social twitter. Les arcs dans le graphe modélisent les relations de « followers».

Chaque entité $v_i$dans le graphe est caractérisée par deux vecteurs d'attributs : xl_i pour les attributs locaux (âge, sexe, endroit, ...), et xr_i pour les attributs relationnels. Une méthode simple pour évaluer le vecteur xr_i est d'estimer le nombre de voisins (successeurs) correspondant à chacune des valeurs de classe.

Si on dispose par exemple de deux valeurs de classe A et B avec $xr_i[0] = 2$ et $xr_i[1] = 3$, le nœud v_i possède une relation avec deux nœuds de classe A et trois nœuds de classe B.

L'étape (a) montre un état initial où seule le nœud v_3 a une valeur de classe connue «y_2». A l'étape (b), une estimation initiale des valeurs de classe inconnues est effectuée par la méthode de classification locale classique, en

utilisant seulement les attributs locaux (arbres de décision, SVM, ...). Ceci permet de calculer les valeurs initiales des attributs relationnels (étape c).

Une classification collective est effectuée par la suite en se basant sur les attributs locaux et relationnels (étape d). A l'étape suivante (e), on recalcule les valeurs des attributs relationnels puisque ces derniers peuvent éventuellement changer à cause de l'opération «re-classification» de l'étape (d).

Les étapes (d) et (e) sont répétées jusqu'à vérification d'un certain critère d'arrêt (nombre d'itérations, stabilisation des valeurs de classe, ...). Le principe général de l'algorithme ICA est résumé dans l'algorithme 3.1.

1) Un classifieur calcule les valeurs de classe initiales correspondant aux nœuds inconnus, en utilisant les attributs locaux

2) **Pour** chaque nœud dont la classe est connue
   Construire les attributs relationnels
   Fin Pour

3) **Construire** un Classifieur ***Cl*** à partir des nœuds dont la classe est connue (*Cl* peut être par exemple les arbres de décision)

4) **Répéter**
   Générer un ordre *O* dans l'ensemble total des nœuds dont la classe est inconnue
   Pour chaque nœud dans l'ensemble ordonné O
      Calculer les attributs relationnels
      Classer le nœud avec le classifieur **Cl**
   Jusqu'à vérification du critère d'arrêt

Algorithme 3.1: Principe de base de l'approche ICA

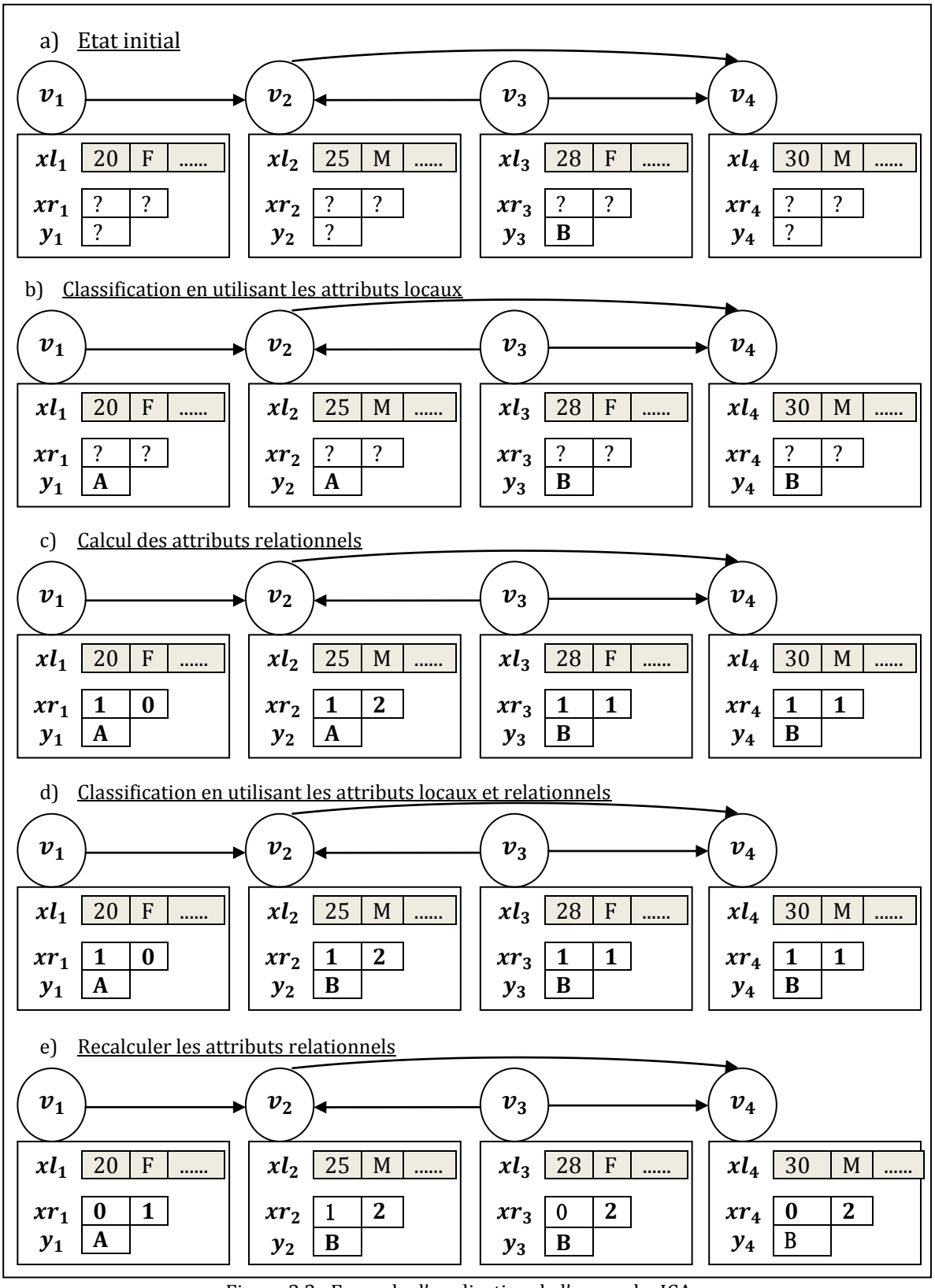


Figure 3.2 : Exemple d'application de l'approche ICA

5.3 Systèmes de recommandation

5.3.1 Définition

Un système de recommandation est une application web qui permet de faire des propositions et suggestions aux utilisateurs (films, vidéos, amis, articles, ...) en se rapprochant le plus de leurs «intérêts». L'objectif est de prédire l'avis de l'utilisateur sur un objet donné.

5.3.2 Classification des techniques des systèmes de recommandation

Les techniques des systèmes de recommandation se basent sur la comparaison du profil de l'utilisateur à certaines propriétés de référence. On cite deux classes d'approches :

- Les approches basées sur le contenu examinent les propriétés des objets recommandés et proposent des objets similaires.
- Les approches de filtrage collaboratif recommandent à l'utilisateur les objets préférés par d'autres utilisateurs similaires.

5.3.3 Systèmes à base de contenu

Les systèmes à base de contenu se focalisent sur les propriétés des objets. La recommandation d'un objet se fait en mesurant la similarité entre les objets selon leurs propriétés. Les caractéristiques des objets à recommander sont analysées, puis regroupées. Par la suite, le système va suggérer aux utilisateurs ayant utilisé un produit quelconque par le passé, les objets estimés similaires. Les éléments essentiels de tels systèmes sont résumés dans les paragraphes suivants :

- **Construction du profil pour chaque objet**

Le profil de l'item est représenté par un vecteur qui contient les caractéristiques principales. Comme exemple, on peut citer la recommandation des films aux utilisateurs. Le profil de l'item dans ce cas correspond aux attributs suivants : Type du film, année de production, acteurs principaux, réalisateur, ..etc. Dans le cas de recommandation de documents, le profil peut représenter des « poids » des mots dans le document (nombre d'occurrence, poids TFIDF, ...).

- **Représentation du profil de l'utilisateur**

Le profil de l'utilisateur est aussi représenté par un vecteur de mêmes composants que les vecteurs des objets. Il décrit les préférences de l'utilisateur.

- **Recommandation d'objets aux utilisateurs en se basant sur le contenu**

Le degré auquel un utilisateur préfèrerait un objet peut être estimé en calculant la distance cosinus entre les vecteurs de l'utilisateur et de l'objet. La mesure de Cosinus quantifie la similarité entre deux vecteurs comme le cosinus de l'angle entre les deux vecteurs.

5.3.4 Recommandation à base de filtrage collaboratif

Cette approche est basée sur l'hypothèse que les utilisateurs qui ont préféré des objets similaires par le passé ont un goût similaire et vont donc apprécier les mêmes objets dans le futur. Un des exemples les plus connus est « Amazon.com ». La similarité des objets est déterminée par la similarité des évaluations de ces objets par les utilisateurs qui les ont évalués.

5.3.5 Autres techniques de systèmes de recommandation

Un système de recommandation peut être construit par un modèle de classification (arbres de décision ou autres) dont le rôle est de prédire l'opinion de l'utilisateur sur un objet donné. La technique du clustering (k-means, hiérarchique, ...) est aussi utilisée dans plusieurs travaux de recherche sur la recommandation.

Exercices

1) Dans le cas du réseau Twitter, donner une interprétation de chacune des propriétés observées généralement dans les médias sociaux:
   a) Distribution des Degrés selon une loi de puissance
   b) Coefficient de clustering élevé
   c) Phénomène «petit monde ou six-degré de séparation »

2) Quel est l'intérêt de la mesure de centralité dans un média social ?

3) Quel est l'intérêt de mesurer le coefficient de clustering dans un média social ?

4) Quel est l'intérêt de mesurer la similarité dans un média social ?

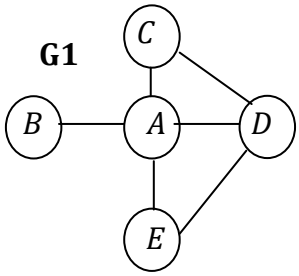
5) Soit le graphe $G1$. Ordonner les nœuds selon leur importance en terme de centralité de proximité. Interpréter le résultat.

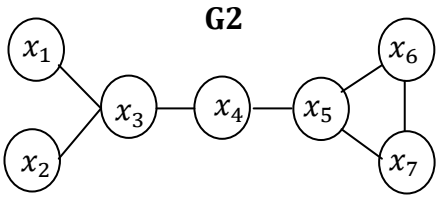
6) Pour le graphe G2, calculer les mesures de centralité : degré, proximité, intermédiarité

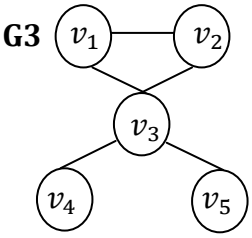
7) Calculer les coefficients de clustering local et global pour le graphe G3. Quel est le nœud dont la force des liaisons entre ses voisins est la plus forte ?

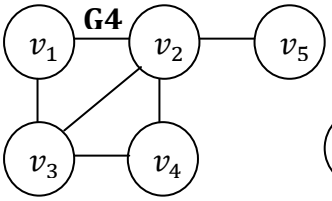
8) Calculer en langage R les centralités spectrales pour le graphe $G4$

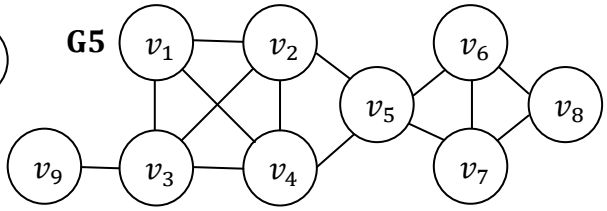
9) Vérifier en langage R les valeurs de centralité d'intermédiarité pour le graphe $G5$ (cf. 4.1.3). Comparer entre les ordres donnés pour chacune des mesures présentées dans ce polycopié.











10) On voudrait vérifier les trois propriétés observées généralement dans les médias sociaux. Effectuer cette analyse en langage R (ou Python) sur les graphes suivants :
   a) Un graphe généré par le modèle scale-free
   b) Un graphe du dataset réel: https://snap.stanford.edu/data/ego-Facebook.html

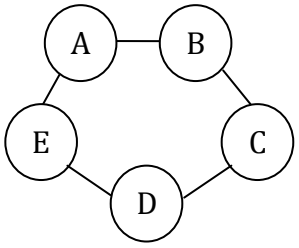
11) Extraire à partir de Facebook https://developers.facebook.com/ votre réseau d'amis et ordonner les nœuds selon leur ordre d'influence.

12) On dispose d'un ensemble de données issues d'un média social où les utilisateurs U_i peuvent effectuer plusieurs actions A_J. L'objectif est de découvrir des groupes d'utilisateurs effectuant des opérations similaires (2 opérations ou plus).
   a) Proposer une méthodologie afin d'extraire ce type de connaissances (Préparation des données, Méthode du datamining appliquée)
   b) Donner et discuter les résultats trouvés en précisant l'algorithme choisi.

Utilisateur	Action
U1	A1
U1	A2
U1	A5
U2	A1
U2	A3
U2	A5
U2	A4
U3	A1
U7	A2
U4	A6
U5	A4
U5	A3
U6	A2
U6	A5
U8	A3
U9	A1

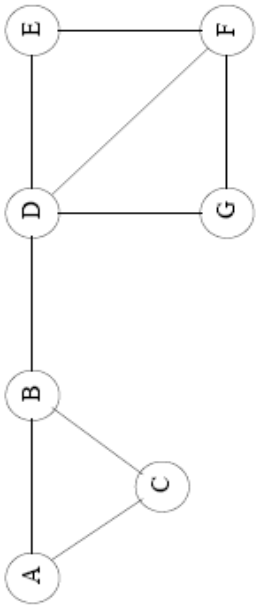
c) Peut on considérer les groupes d'utilisateurs découverts comme des « communautés » ? Pourquoi ?

13) En appliquant l'algorithme de Girvan et Newman (GN) sur le graphe suivant, les arêtes sont supprimées dans cet ordre: AB – CD – DE – BC – EA



a) Schématiser la hiérarchie de construction des communautés
b) Donner la partition obtenue juste avant l'itération qui regroupe tous les nœuds dans une seule communauté puis évaluer sa qualité.
c) Pourquoi l'algorithme GN est couteux en temps d'exécution pour les grands graphes ?

14) On veut découvrir les communautés qui existent dans le graphe suivant en appliquant l'algorithme de Girvan et Newman (GN).



a) Visualiser les communautés découvertes
b) Donner l'ordre de suppression des liens
c) Vérifier la qualité de la partition (A, B, C, D) (E, F, G).

15) Une approche alternative pour détecter les communautés consiste à utiliser le clustering hiérarchique. Pour mesurer la distance entre deux nœuds i et j on utilise la mesure suivante :

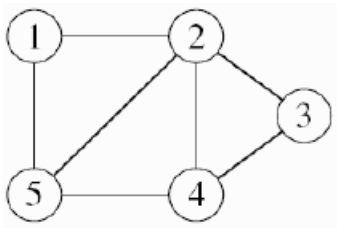
$$d_{ij} = degre_i + degre_j - 2 * \times nombre\ de\ noeuds\ voisins\ communs\ entre\ les\ noeuds\ i\ et\ j$$

a) Ecrire un script en R qui permet de construire la matrice de similarité. L'application de votre code sur l'exemple du graphe G devrait donner la matrice D (voir schéma).

b) Donner le dendogramme correspondant au graphe G.

c) Comparer la qualité de la partition trouvée avec le cas où on utilise l'algorithme GN

G ***D***



$$\begin{bmatrix} 0 & 4 & 2 & 1 & 3 \\ 4 & 0 & 4 & 3 & 3 \\ 2 & 4 & 0 & 3 & 1 \\ 1 & 3 & 3 & 0 & 4 \\ 3 & 3 & 1 & 4 & 0 \end{bmatrix}$$

d) Effectuer une étude expérimentale comparative entre l'algorithme GN et cette approche en terme de temps d'exécution et qualité de la partition. Utiliser des données synthétiques et/ou des données de test disponibles sur le net: karate-club Zachary, ego_facebbok, gemsec-Deezer ...

16) On suppose que ces données sont issues d'un média social et représentées sous forme d'un graphe non orienté (voir figure). Les nœuds (X7, X8, X9) ont la valeur de classe inconnue. En appliquant une méthode de classification locale utilisant les attributs locaux, on suppose obtenir les valeurs de classe suivantes (X7 : P, X8 : Q, X9 : Q).

	X1	X2	X3	X4	X5	X6	X7	X8	X9
X1	...	1	1	...	...	...		1	
X2	...	...	...	...	1	...	1		1
X3	...	...	...	1	...	...			
X4	...	...	...	...	1	...	1		1
X5	...	...	...	...	...	...	1	1	1
X6	...	...	...	...	...	...		1	
X7	...	...	...	...	...	...			
X8	...	...	...	...	...	...			1
X9	...	...	...	...	...	...	1		

a) Donner le contenu des vecteurs des attributs relationnels pour les nœuds X7, X8, X9 dans le cas de la première itération de l'approche ICA.
b) Quelle est le critère d'arrêt de l'approche ICA ?

Références

[1] R. Agrawal, R. Srikant. Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases, Morgan Kaufmann, Santiago, Chile, pp. 487–499, 1994.

[2] P. Bonacich. Factoring and weighting approaches to status scores and clique identification. Journal of Mathematical Sociology 2, 113-120. 1972.

[3] U. Fayyad. Knowledge discovery in databases: An overview. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), Springer, Berlin, Heidelberg, vol. 1297, 1997.

[4] J. Han, J. Pei, Y. Yin. Mining frequent patterns without candidate generation. Proceedings of the ACM SIGMOD International Conference on Management of Data. Dallas, TX: ACM Press, pp. 1–12, 2000.

[5] A. Kaplan and M. Haenlein. Users of the world, unite! the challenges and opportunities of social media. Business Horizons, 53(1):59–68, Jan 2009.

[6] D. Mokeddem, H. Belbachir. A survey of Distributed Classification Based Ensemble Data Mining Methods. Journal of Applied Sciences 9(20) : 3739-3745, ISSN 1812-5654, 2009.

[7] D. Mokeddem. Datamining Distribué: Contribution à l'Amélioration des Performances. Thèse de doctorat en sciences, Département d'Informatique, USTOMB, 2017.

[8] J. Quinlan. Induction of decision trees. Machine Learning, pp. 81-106, 1986.

[9] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad. Collective classification in network data. Artificial Intelligence Magazine 29(3):93–106, 2008.

[10] R. Zafarani, M. A. Abbasi, H. Liu. Social media mining: An Introduction. Cambridge University Press, 2014. http://dmml.asu.edu/smm/SMM.pdf